

CS 498: Machine Learning System Spring 2025

Minjia Zhang

The Grainger College of Engineering



Mixed Precision Training





- Mixed Precision Hardware
- What is Mixed Precision Training?
- Considerations for Mixed Precision
- Mixed Precision Software

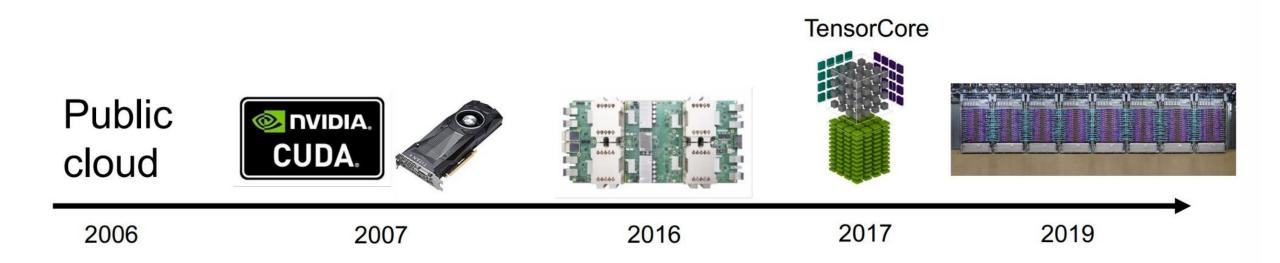




Mixed Precision Hardware





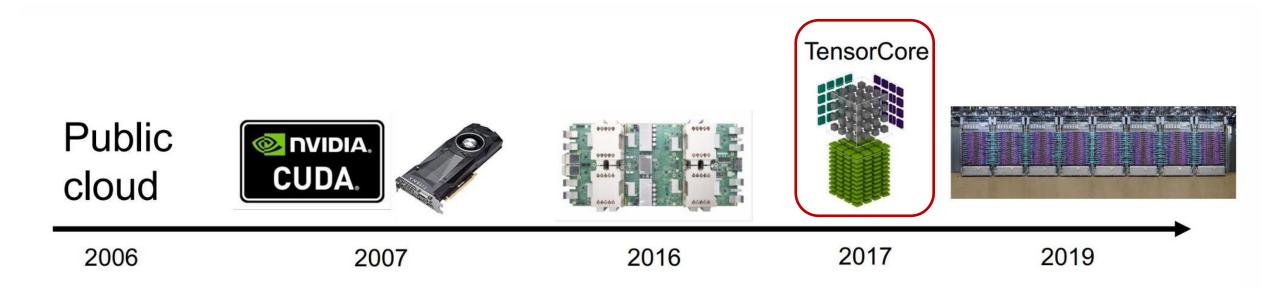


Compute scaling



COMPUTER SCIENCE





Compute scaling

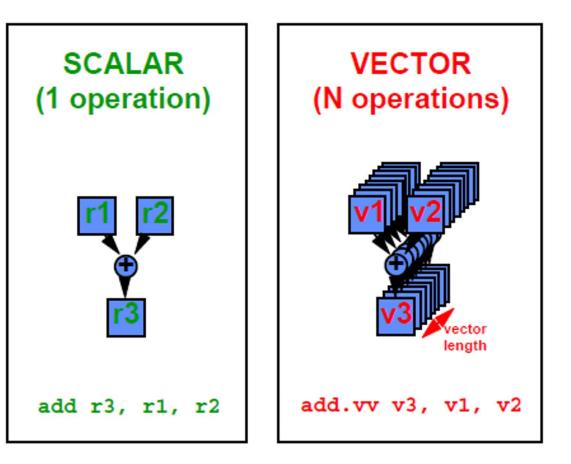


COMPUTER SCIENCE

Vector Processing Unit

Ι

- A processing unit that operates on an entire vector in one instruction (SIMD)
- The operand to the instructions are vectors instead of a scalar
- Work done in parallel



What are Tensor Cores? Matrix Multiplication Units

				L1Imtrust	ion Cache				
	L0.	nstruction C	acho:		-	L	Instruction	Recto	
Warp Scheduler (32 thread/c9/)				Warp Schedular (32 threadicle)					
	Dispate	th Unit (32 sh	readicik)		<u> </u>	Disper	ch Unit (321	hread/clk)	_
	Register File (16,384 x 32-bit)					Registe	r File (16,3	84 x 32-bit)	
FP64	NT NT	6932-69			FP64	INT IN	FP32 FP3		
FP64	INT INT	193210			FP64	INT: IN	FR02 FP0		
FP64	INT INT	EP32 EP			FP64	INT IN	FP32 FP3		
FP64	INT INT	FROD FR	TENSOR	TENSOR	FP64	INT IN	F 1932 FP3	TENSOR	TENSOR
FPS4	INT INT	FP32-FP	CORE	CORE	FP64	INT: INT	F FP32 FP3	CORE	CORE
FP64	INT INT	FP32 FP			FP64	INT IN	FROM FROM		
1154	INT INT	F#32 FF			1764	INT INT	FIRST FIRST		
FP54	INT INT	11932 11 2			FP64	INT IN	FRSZ FPS		
101 101 87 87	LDI LDV ST ST	10/ 10/ 81 11	LDF LDF	SFU	LOV LOY BT BT	LDY LD ST ST		LD' LD' ST ST	SFU
	60	nstruction C	ache:			010	Instruction	Riberta I	
	A REAL PROPERTY AND ADDRESS OF	heduler (321	the state of the s			Warp S	cheduler (32	mreed/city)	-
	Dispate	h Unit (32 th	readicik)			Dispat	ch Unit (32 t	hread/clk)	
	Register	File (16,38	4 x 32-bit)			Registe	r File (16,3	64 x 32-bit)	
	INT INT	FP32 FP32			FP64	INT IN	1012 102		
FP64							and the second second		
FP64	INT INT	1932 1932			FP64	INT IN	C FROM FROM		
	INT INT	FP32 FP32 FP32 FP32			FP64 FP64	INT INT			
7994			TENSOR	TENSOR			FP32 FP3	TENSOR	TENSOR
FP64	WT INT	FP32 FP32	TENSOR CORE	TENSOR CORE	FP64	1MT 1N	r FP32 FP3 r FP32 FP3		TENSOR
FP64 FP64	INT INT INT INT	FP32 FP32 FP32 FP32			FP64 FP64	INT IN	r 6932 693 6 6932 693 7 6932 693	TENSOR	
FP94 FP94 FP94 FP94	THE THE INT DAT INT THE	FP32 FP32 FP32 FP32 FP32 FP32			FP64 FP64 FP64	INT IN INT IN INT IN	r FP32 FP3 FP32 FP3 FP32 FP3 FP32 FP3 FP32 FP3	TENSOR CORE	
FP94 FP94 FP94 FP94 FP94	THE THE THE THE THE THE THE THE	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32			FP64 FP64 FP64 FP64	INT IN INT IN INT IN INT IN	FP32 FP3 FP32 FP3 FP32 FP3 FP32 FP3 FP32 FP3 FP32 FP3	TENSOR CORE	
FP94 FP94 FP94 FP94 FP94 FP94	THE THE THE THE THE THE THE THE THE THE	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32			FP64 FP64 FP64 FP64 FP64	INT IN INT IN INT IN INT IN INT IN	r P32 P93 r P32 P93	TENSOR CORE	
FP64 FP64 FP64 FP64 FP64 FP64 FP64 ED ED	THE	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	CORE LOT LOT ST ST	CORE	FP64 FP64 FP64 FP64 FP64 FP64 LDV LDV ST ST	Brt Brt Brt Brt	FP32 FP32 FP32 FP32	TENSOR CORE	CORE

VOLTA ARCHITECTURE

• Tensor cores are:

- Built to accelerate deep learning
- Special hardware execution units
- Execute matrix multiply operations (fused multiply-add on small matrices) in one instruction cycle

						-							
L0 instruction Cache Warp Scheduler (32 thread/c9)					L8 Instruction Ceche Warp Schedular (32 threedicle)				_				
Dispatch Unit (32 threadiclk)						Dispetch Unit (32 thread/clk)			-				
	Reg	ister	File (16	384	x 32-bit)			Reg	pister	File (16,38	4 x 32-bit)	
FP64	-017	мт	FP32 F	1			FP64	197	INT	EP34	FIPSIa		
FP64	INT	INT	1032.0	1			FP64	INT:	нт	1632	FIFSE		
FPG	187	INT	1992 F	1			FP64	INT	INT	1933	FP33		
FP54	нт	INT	FP32 F	1	TENSOR	TENSOR	FP64	INT	INT	6603	ress	TENSOR	TENSOR
FP54	INT	мт	F#32 F	1	CORE	CORE	1764	INT	INT	T POZ	FIPSS	CORE	CORE
FP64	INT	INT	FP32 F				EP64	INT	вт	FR33	FPOD		
FPSA	INT	INT	FP32 F	1			1964	шт	INT	1100	F#32		
FP64	INT	INT	FP-12	ł			FP64	INT	INT	1912	FP32		
1 LD/	101 87	LDV ST	UDF 8T	101	LDF LDF	SFU	LDV LDV BT BT	1.07 \$1	LD/ aT	1.07 \$1	LDY ST	LDF LDF	SFU
	1000			_	iche				_	_	tion C		-
_	Di	spatch	h Unit (J	12 11 12 thr	eadicik)			Di	n gi Sch spatici	h Unit	(32 th	nneedich) ireadicik)	_
	Di Reg	spatch	File (10	12 the	read/clk)			Di Reg	ep Sch spatc) pistor	h Unit File ((32 th (32 th	nreed/clk)	
FP64	Di Reg	spatch lister INT	File (10	12 th 12 th 1, 184 P32	eadicik)		F994	Reg	n <mark>g Sch</mark> spatic jistor INT	File ((32 th (32 th 16,38 FP34	nneedich) ireadicik)	
1994	Di Rog art art	spatch lister JMT JMT	File (10 File (10 FP32 F	12 194 12 194 1, 3, 8, 4 1932 1932	eadicik)		1964	Di Reg INT	epato pistor INT INT	File (File (FP32	(32 b) (32 b) 16,38 7932	nneedich) ireadicik)	
F7954	D) Rog art art art	ister Inter Int Int	File (10 File (10 File 2 File 2 File 2	12 (b) 12 (b) 13 (b) 13 (b) 14	veedick) eadick) x 32-bit)		1964 1964	Reg INT INT	int int int int int int	File (File (FP32 FP32 FP32	(32 th (32 th 16,38 7932 7932 7933	Preedicite) readicite) 4 x 32-bit)	
FP54 FP54 FP54	Rog Rog RT RT RT RT	inter Inter Int Int Int Int	File (10 File (10 FP32 F FP32 F FP32 F	12 De 12 De 1,384 1932 1932 1932	eadicik)	TENSOR	1964 1964 1964	Di Reg INT INT INT INT	istor INT INT INT	File (File (FP32 FP32 FP32 FP32	(32 th (32 th 16,38 FP32 FP32 FP32 FP32	nneedich) ireadicik)	TENSOR
FP64	Rog StT StT StT StT StT	ister Inter Int Int	File (10 File (10)File (10 File (10)File (10)Fil	12 (b) 12 (b) 13 (b) 13 (b) 14	readicik) adicik) x 32-bit) TENSOR		1964 1964	Reg INT INT	int int int int int int	File (File (File) File (File) File (File)	(32 b) (32 b) 16,36 7932 7932 7933 7933	need(cib) reed(cib) 4 x 32-bit) TENSOR	
FP64 FP64 FP64 FP64	Rog Rog RT RT RT RT	ister ister int int int int	File (10 File (10)File (10 File (10)File (10)Fil	92 0 92 0 932 0 932 932 932 932 932	readicik) adicik) x 32-bit) TENSOR		1964 1964 1964 1964	Di Reg INT INT INT INT	pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater pater	File (File (FP32 FP32 FP32 FP32	(32 th (32 th 16,38 FP32 FP32 FP33 FP33 FP33	need(cib) reed(cib) 4 x 32-bit) TENSOR	
FP54 FP54 FP54 FP54	Rog Bit Bit Bit Bit Bit Bit Bit	ant ant ant ant ant ant ant ant	File (10 File (10)File (10 File (10)File (10)Fil	92 0 12 994 1,384 1932 1932 1932 1932 1932	readicik) adicik) x 32-bit) TENSOR		1964 1964 1964 1964 1964	Reg INT INT INT INT	int int int int int int int	File (File (FP32 FP32 FP32 FP32 FP32 FP32	(32 th (32 th 16,38 FP32 FP32 FP33 FP33 FP33	need(cib) reed(cib) 4 x 32-bit) TENSOR	
7794 7994 7994 7994 7994	Rog art art art art art art art	spatid ister MT MT MT MT MT MT	File (10 File (10)File (10)Fil	92 0 12 0 1,384 1,384 1932 1932 1932 1932	readicik) adicik) x 32-bit) TENSOR		1944 1944 1944 1944 1944	DA Reg INT INT INT INT INT	ep Sch spatch pistor NT NT NT NT	File (File (FP32 FP32 FP32 FP32 FP32 FP32 FP32	(122 th (12 th 16,30 (1932 (1932 (1932 (1932 (1932 (1932	need(cib) reed(cib) 4 x 32-bit) TENSOR	

VOLTA ARCHITECTURE

• Tensor cores are:

- Built to accelerate deep learning
- Special hardware execution units
- Execute matrix multiply operations (fused multiply-add on small matrices) in one instruction cycle

Different flavors

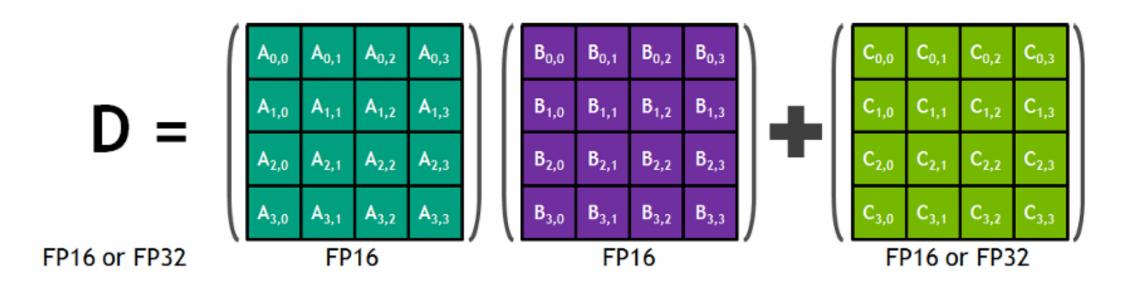
- (V100) Volta Tensor Cores FP16
- (T4) Turing Tensor Cores FP16/INT8/INT4

	_			1000	Literatives						-	_
			netrureti o			L6 Instruction Cache						
	I DOWN	1000	No. of Concession, Name	2 threadicile) threadicile)		Warp Schedular (32 thread/clk) Dispatch Unit (32 thread/clk)						
Register File (16,384 x 32-bit)						Register File (16.384 x 32-bit)						
				1CTTT	10111		-	-				
FP64	-err	MT	FP32 FI			FP64	INT	INT	EP32			
FP64	INT	INT	1035.01			F964	BIT:	INT	1605	FIFTE		
FPG4	INT	INT	F#32 FF			FP64	INT	INT	1911	FP33		
FP64	нт	INT	FIRST RI	TENSO		FP64	INT	INT	F 1932	F(P)32	TENSOR	
FP64	INT	мт	F#22 FF	CORE	CORE	FP64	INT	INT	(POZ	FP32	CORE	CORE
FP64	INT	INT	FP22 FF			FP64	INT	INT	FR32	6932		
F754	INT	INT	F#32 FF			1964	шт	INT	1102	11132		
FP64	INT	INT	FP-12 F1			FP64	INT	INT	1912	F#32		
101 LUN 87 87	LOI ST	1.0V 81	101 I 81	LD LD	SFU	LDV LDV ST ST	1.07		1.07 ST	LOY ST	LD' LD' ST ST	SFU
			Josef L A	210000200								
			netruccio	n Cache					na bracci	tion G	actes	
				The other states and the state of the			1.0		A COLUMN TWO IS NOT	1122	A REAL PROPERTY.	-
	-	and so the local division of the	1000	12 thread/clk) t threadiclk)			-	and some first the		-	nreedicik) read/cik)	
	Di	spate	h Unit (36	the survey of the second			Di	spatc	h Unit	(32 th	and the second second second	
FP64	Di	spate	h Unit (36	t threadiclk) 384 x 32-bit		7964	Di	spatc	h Unit	(32 th	read/clk)	
FP64 FP64	Di Reg	spatic	File (16,	t threadiclk) 384 x 32-bit		FP64 FP64	Di Reg	epatic pistor	File (1	(32 th	read/clk)	
	Di Reg	spatch lister INT	File (16, File (16, Fraz Fi	2 threadiclk) 364 x 32-bit			Reg	istor INT	Filo (1 Filo (1	(32 th (6,36) 7932	read/clk)	
FP64	Di Rog art art	spatel lister INT INT	File (16, File (16, Frizz Fi Frizz Fi Frizz Fi	1 Dreadictk) 384 x 32-bit 92 92		1964	Di Reg INT	istor INT INT	Filo (1 Filo (1 FP32	(32 th 16,384 7932	readicits) 4 x 32-bit)	TENSOR
FP64	Di Rog art art art	spatel Inter INT INT	- Unit (3) File (16, 1932 FF 1932 FF 1932 FF	1 threadictk) 384 x 32-bit 92 92		7964 7964	Di Reg INT INT	istor INT INT INT	File (1 File (1 FP32 FP32 FP32	(32 th 16,384 FP32 FP32 FP33	read/clk)	TENSOR
FP64 FP64 FP64	Rog Set Set Set Set	inter Inter INT INT INT INT	H Unit (3) File (16, 1932 FF 1932 FF 1932 FF 1932 FF	204 x 32-bit 92 92 92 92 92 92 92 92 92 92 92 92 92	TENSOR	1964 1964 1964	Di Reg INT INT INT	int Int Int Int	Filo (1 Filo (1 FF32 FF32 FF32	(32 (h) (6,36) 7932 7932 7932 7932	read/cik) 4 x 32-bit) TENSOR	
FP64 FP64 FP64 FP64	Rog Rog art art art art art	inter Inter Int Int Int Int	File (16, File (16, File 2 Fi File 2 Fi File 2 Fi File 2 Fi File 2 Fi File 3 Fi	2 Shreadicik) 384 x 32-bit 92 92 92 92 93 93 12ENSO 60RE	TENSOR	194 194 194 194	Di Reg INT INT INT INT	istor INT INT INT INT INT	Filo (1 Filo (1 FF32 FF32 FF32 FF32	(32 th (6,384 7932 7932 7932 7932	read/cik) 4 x 32-bit) TENSOR	
FP64 FP64 FP64 FP64 FP64	Rog arr arr arr arr arr	ANT ANT ANT ANT ANT ANT ANT	H Unit (2) File (16, 1932 FF 1932 FF 1932 FF 1932 FF 1932 FF 1932 FF	2 Shreadicik) 354 x 32-bit 92 92 92 92 92 92 92 92 92 92	TENSOR	1744 1744 1744 1744 1744	Di Reg INT INT INT INT INT	9507 947 947 947 947 947	File (1 File (1 FP32 FP32 FP32 FP32 FP32	(32 th (6,38) (932 (932 (932 (933 (933 (933	read/cik) 4 x 32-bit) TENSOR	
FP64 FP64 FP64 FP64 FP64 FP64	Di Rog arr arr arr arr arr arr arr arr	spatic inter NT NT NT NT NT NT	 Unit (2) File (16, re32 Fi 	204 x 32-bit 384 x 32-bit 92 92 92 92 92 92 92 92 92 92 92 93 92 93 93 93 93	TENSOR	1794 1794 1794 1794 1794 1794 1794	DA Reg INT INT INT INT INT INT	anata astor ast ast ast ast ast ast ast ast ast	h Unit Filo (1 1932 1932 1932 1932 1932 1932 1932	(12 th) (6,304 (932) (932) (932) (932) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (933) (93	read/cb) (x 32-bit) TENSOR CORE	
FP64 FP64 FP64 FP64 FP64 FP64	Di Reg art art art art art art art	spatici ister ist ist ist ist ist ist ist	 Unit (2) File (16, re32 Fi 	204 x 32-bit 384 x 32-bit 92 92 92 92 92 92 92 92 92 92	TENSOR	294 294 294 294 294 294 294 294 295	Di Reg INT INT INT INT INT INT INT INT INT	940 940 947 947 947 947 947 947 947 947 947	File (1 File (1 FP32 FP32 FP32 FP32 FP32 FP32 FP32	(12 th (6,384 (932) (932) (932) (932) (932) (932) (932) (932)	read/cik) 4 x 32-bit) TENSOR	CORE

VOLTA ARCHITECTURE

•

GRAINGER ENGINEERING

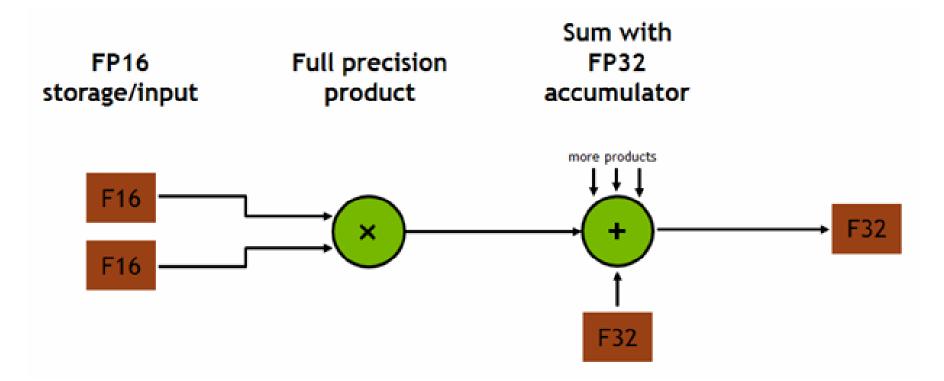


D = AB + C



COMPUTER SCIENCE





- Accelerate matrix multiplications and convolutions
- Tensor core optimized libraries: cuDNN, cuBLAS, CUTLASS

https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/





What is Mixed Precision Training?



• Number can be represented by $(-1)^{S} * (1, M) * 2^{(E - Bias)}$

(sign)	(fraction or mantissa)

(exponent)

Half Precision (Floating Point 16)	1 bit	5 bit	10 bit
Single Precision (Floating Point 32)	1 bit	8 bit	23 bit
Double Precision (Floating Point 64)	1 bit	11 bit	52 bit
Quadruple Precision (Floating Point 128)	1 bit	15 bit	113 bit

COMPUTER SCIENCE

GRAINGER ENGINEERING



• Number can be represented by $(-1)^{S} * (1, M) * 2^{(E - Bias)}$

(sign)	(fraction or mantissa)	0 10000001 101000000000000000000000000
	Í	Assume blas is 127

(exponent)

Half Precision (Floating Point 16)	1 bit	5 bit	10 bit
Single Precision (Floating Point 32)	1 bit	8 bit	23 bit
Double Precision (Floating Point 64)	1 bit	11 bit	52 bit
Quadruple Precision (Floating Point 128)	1 bit	15 bit	113 bit



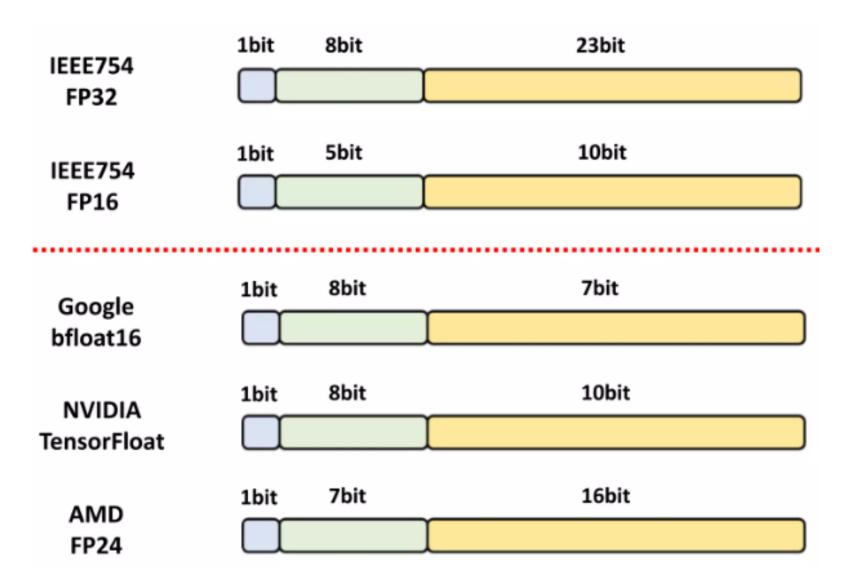
• Number can be represented by $(-1)^{S} * (1, M) * 2^{(E - Bias)}$

(sign)	(fraction or mantissa)	0 10000001 10100000000000000000000
		Assume bias is 127 (-1)^0 * (1.625) * 2^(129 - 127) = 1.625 * 4
	·	= 6.5

(exponent)

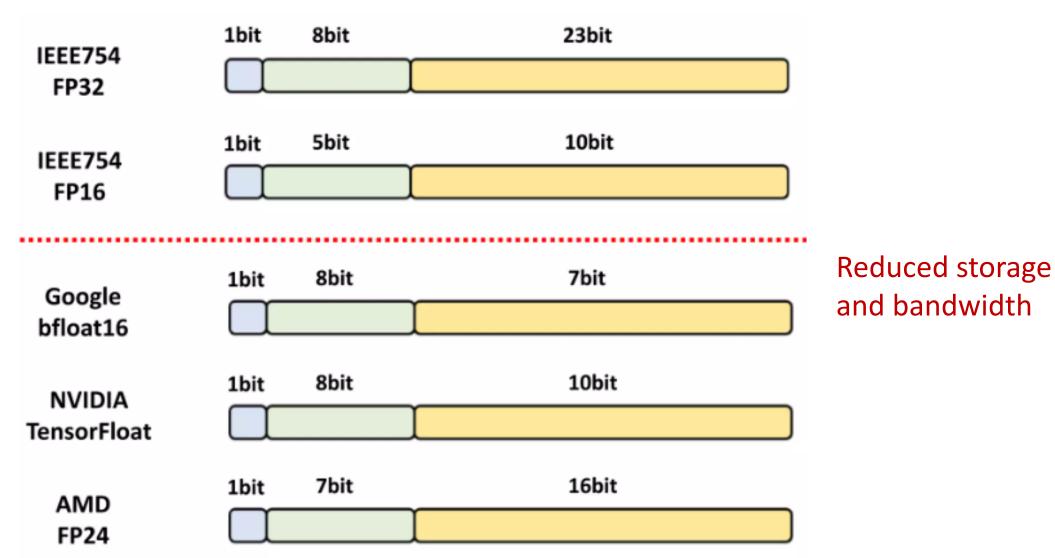
Half Precision (Floating Point 16)	1 bit	5 bit	10 bit
Single Precision (Floating Point 32)	1 bit	8 bit	23 bit
Double Precision (Floating Point 64)	1 bit	11 bit	52 bit
Quadruple Precision (Floating Point 128)	1 bit	15 bit	113 bit

New Floating-Point Format



GRAINGER ENGINEERING

New Floating-Point Format



COMPUTER SCIENCE

GRAINGER ENGINEERING

What is Mixed Precision Training? In a Nutshell

- Idea that you can train deep neural networks in multiple precisions:
 - Make precision decisions per layer or operation
 - Full precision (Fp32) where needed to *maintain task-specific accuracy*
 - Reduced precision (Fp16) everywhere else for speed and scale

What is Mixed Precision Training? In a Nutshell

- Idea that you can train deep neural networks in multiple precisions:
 - Make precision decisions per layer or operation
 - Full precision (Fp32) where needed to *maintain task-specific accuracy*
 - Reduced precision (Fp16) everywhere else for speed and scale
- By using *multiple* precisions, we can have the best of both worlds: speed and accuracy
- Goal: accelerate deep neural network training with mixed precision under the constraints of matching accuracy of full precision training and no changes to how model is trained

• Accelerates speed

• Tensor cores are 8x faster than FP32



• Accelerates speed

• Tensor cores are 8x faster than FP32

• Reduces memory bandwidth pressure

• FP16 halves memory traffic compared to FP32

• Accelerates speed

• Tensor cores are 8x faster than FP32

• Reduces memory bandwidth pressure

• FP16 halves memory traffic compared to FP32

• Reduces memory consumption

- FP16 halves the size of activation and gradient tensors
- Enables larger models, mini batches or inputs

Ι

- Models cannot always converge properly in pure FP16
 - FP16 has a narrower dynamic range than FP32
 - May cause underflow/overflow issues and other arithmetic issues



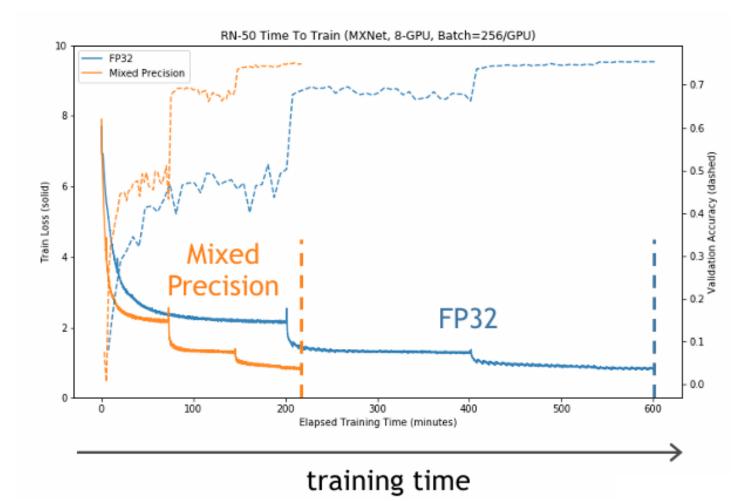
- Models cannot always converge properly in pure FP16
 - FP16 has a narrower dynamic range than FP32
 - May cause underflow/overflow issues and other arithmetic issues
- Weight updates
 - Optimizer takes very small increments when search narrows into a solution
 - Late updates often cannot be represented in FP16, but can be crucial for accuracy

- Models cannot always converge properly in pure FP16
 - FP16 has a narrower dynamic range than FP32
 - May cause underflow/overflow issues and other arithmetic issues
- Weight updates
 - Optimizer takes very small increments when search narrows into a solution
 - Late updates often cannot be represented in FP16, but can be crucial for accuracy
- Reductions
 - Large sums of values, e.g., in linear layers and convolutions, can be **too big** for FP16
 - Adding small values to a large sum can lead to **rounding errors**



Mixed Precision Training Example

- ResNet-50 training for ImageNet classification
 - 8 GPUs on DGX-1
- Comparing to FP32 training
 - ~3x speedup
 - Equal accuracy
 - No hyperparameters changed



 Works across a wide range of tasks, problem domains, deep neural network architectures

Image Classification	Detection / Segmentation
AlexNet	DeepLab
DenseNet	Faster R-CNN
Inception	Mask R-CNN
MobileNet	SSD
NASNet	NVIDIA Automotive
ResNet	RetinaNet
ResNeXt	UNET
ShuffleNet	
SqueezeNet	Recommendation
VGG	DeepRecommender
Xception	NCF

Generative Models (Images)	Language Modeling
DLSS	BERT
GauGAN	BigLSTM
Partial Image Inpainting	Gated Convolutions
Progress GAN	mLSTM
Pix2Pix	RoBERTa
	Transformer XL
Speech	
Speech Deep Speech 2	Translation
	Translation Convolutional Seq2Seq
Deep Speech 2	
Deep Speech 2 Jasper	Convolutional Seq2Seq
Deep Speech 2 Jasper Tacotron	Convolutional Seq2Seq Dynamic Convolutions

COMPUTER SCIENCE



Task	Model	Speedup
	ResNet-50	3.6x
Image Classification	DenseNet 201	2.2x*
Classification	Xception	2.1x*
	SSD	2.5x**
Detection / Segmentation	Mask R-CNN	1.5x
beginentation	RetinaNet	2.0x
* 2x batch size	** Larger batch	n size

Weeks to days

Days to hours

Hours to minutes

Task	Model	Speedup
Translation	GNMT	2.3x
	Transformer	2.9x 4.9x**
	Convolutional Seq2Seq	2.5x*
Speech	Deep Speech 2	4.5x**
	Wav2letter	3.0x*
	WaveGlow	1.9x
Language Modeling	mLSTM	4.0x**
	BERT	3.3x

"This paper shows that reduced mixed precision and large batch training can speedup training by nearly 5x on a single 8-GPU machine with careful tuning and implementation." Scaling Neural Machine Translation, Facebook

"We train with mixed precision floating point arithmetic on DGX-1 machines [...] using 1024 V100 GPUs for approximately **one day**."

RoBERTa, Facebook

"leverages mixed precision training [...] largest transformer based language model ever trained at 24x the size of BERT and 5.6x the size of GPT-2." MegatronLM, NVIDIA



Considerations for Mixed Precision



- Goal #1: Make FP16 training general purpose, not only for limited class of applications
- Goal #2: With no changes to hyperparameters or the model architecture

Three parts:

PRECISION OF OPS

Decide which operations to compute in FP16 and FP32.

MASTER WEIGHTS

Keep an FP32 copy of the model weights.

LOSS SCALING

Scale the loss value to retain small gradients.

Matrix Multiplication linear, matmul, bmm, conv

8x performance boost from Tensor Cores



Matrix Multiplication linear, matmul, bmm, conv

8x performance boost from Tensor Cores

Pointwise relu, sigmoid, tanh, exp, log **Reductions** batch norm, layer norm, sum, softmax

Loss Functions cross entropy, 12 loss, weight decay

still get some speedup (e.g. 2x memory savings), but without sacrificing accuracy



- Operations that can use FP16
 - Matrix multiplications
 - Most element-wise operations (e.g., relu, tanh, add, sub, mul)

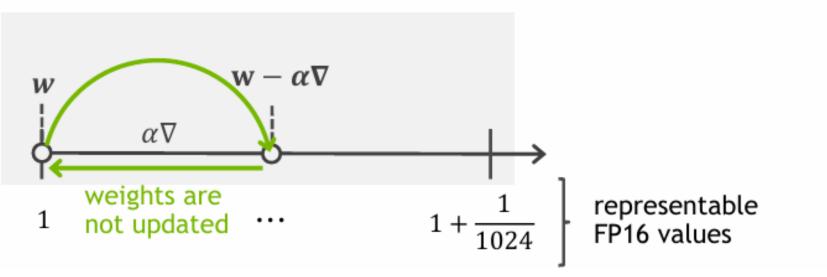
- Operations that can use FP16
 - Matrix multiplications
 - Most element-wise operations (e.g., relu, tanh, add, sub, mul)
- Operations that need FP32 mantissa
 - Reduction operations

- Operations that can use FP16
 - Matrix multiplications
 - Most element-wise operations (e.g., relu, tanh, add, sub, mul)
- Operations that need FP32 mantissa
 - Reduction operations
- Operations that need FP32 range
 - Element-wise operations where |f(x)| >> |x|, e.g., exp, log, pow
 - Loss functions

- **Problem:** in late stages of training, weight updates become too small for addition in FP16
- **Consequence:** weight update gets clipped to zero when $w \gg \alpha \nabla$

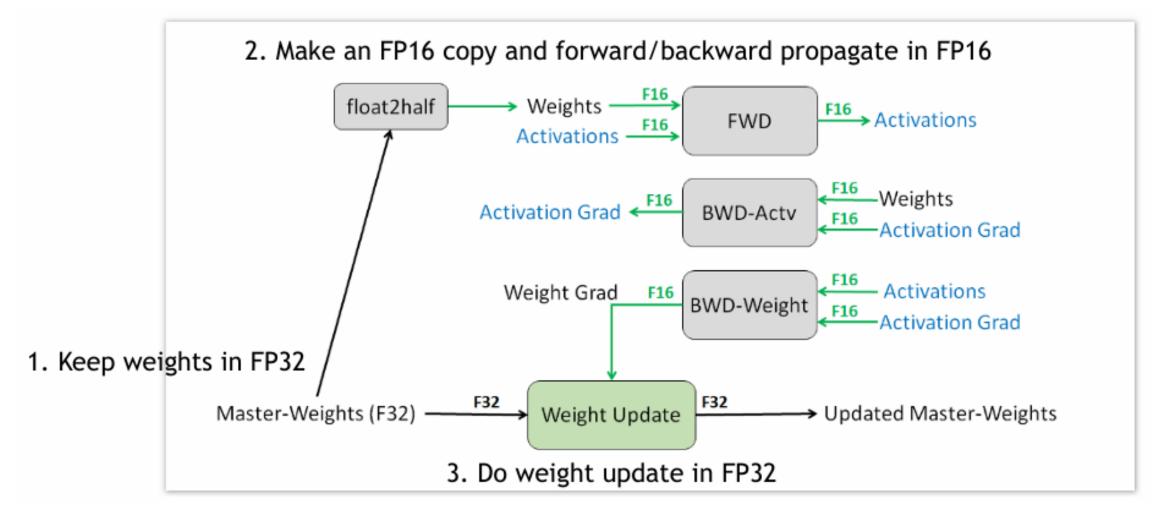


- **Problem:** in late stages of training, weight updates become too small for addition in FP16
- **Consequence:** weight update gets clipped to zero when $w \gg \alpha \nabla$



• **Conservative solution:** keep master copy of weights in FP32 so small updates can accumulate

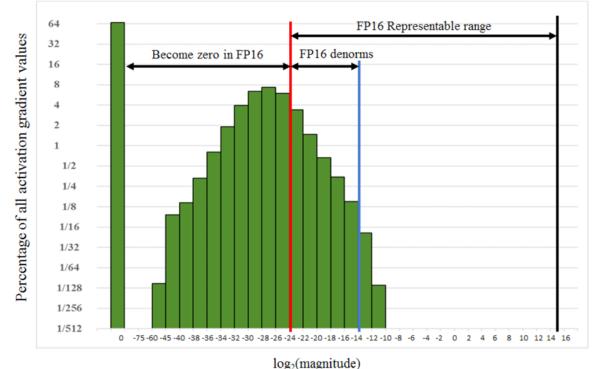




Loss Scaling: Put All Tensors in FP16 Range



- Range representable in FP16: ~40 powers of 2
- Gradients are small:
 - If cast to Fp16, some lost to zero
 - Much of fp16 representable ranges was left unused



log₂(magnitude)

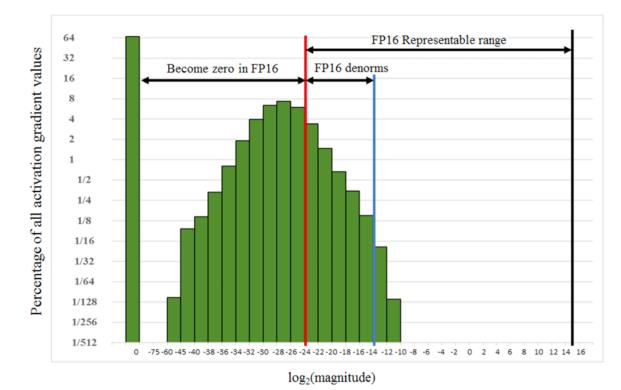
Histogram of activation gradient values

Loss Scaling: Put All Tensors in FP16 Range



- Range representable in FP16: ~40 powers of 2
- Gradients are small:
 - If cast to Fp16, some lost to zero
 - Much of fp16 representable ranges was left unused

Question: How can we make better use of FP16 representable range?



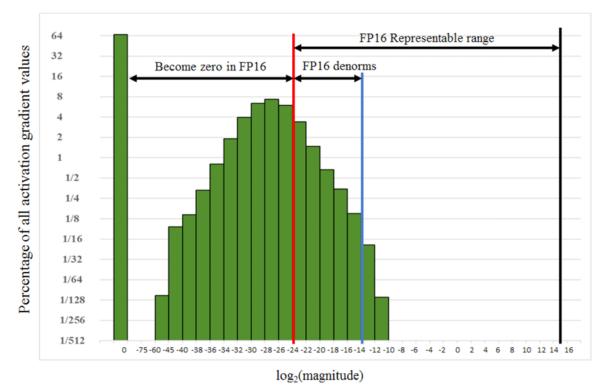
Histogram of activation gradient values

42 GRAINGER ENGINEERING

Loss Scaling: Put All Tensors in FP16 Range

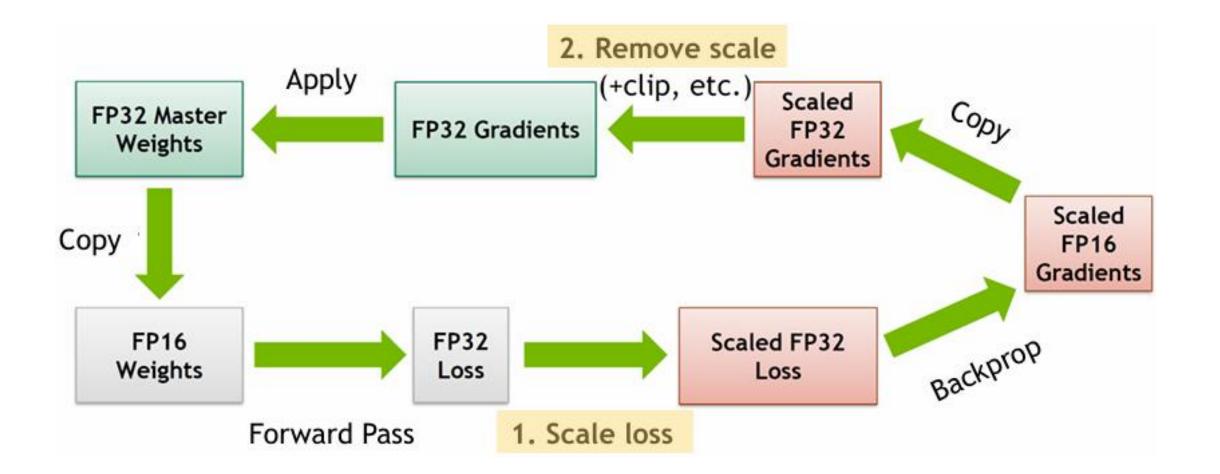


- Range representable in FP16: ~40 powers of 2
- Gradients are small:
 - If cast to Fp16, some lost to zero
 - Much of fp16 representable ranges was left unused
- Solution:
 - Move small gradients values to fp16 range
 - Scaling gradients during backpropagation to prevent underflow
 - Gradients are scaled before backpropagation begins and rescaled before updating weights



Histogram of activation gradient values







- 1. Start with very large scale factor (e.g., 2^24)
- 2. If gradients overflows (with Inf or a Nan): decrease the scale by 2 and skip the update
- 3. If no overflows have occurred for some time: increase the scale by 2 (e.g., 2000 iterations)



Train With Mixed Precision - NVIDIA Docs



Mixed Precision Software



1

- Goal: Make mixed precision training easy with minimum effort for DL practitioners
- Available for major DL frameworks

PyTorch | DeepSpeed | Megatron

- Works with all types of optimizers
 - SGD, Adam, etc
- Works with multiple models, optimizers, and losses

Enabling Mixed Precision

- Traditionally a lot of work (recap on methodology)
- Model conversion
 - Switch everything to run on FP16 values

- Cast to FP32 for loss functions, normalization, and element-wise ops that need full precision

- Master weights
 - Keep FP32 copy of model parameters
 - Make FP16 copies during forward/backward passes
- Loss scaling
 - Scale the loss value, unscale the gradients in FP32
 - Check gradients at each iteration to adjust loss scale and skip on overflow



- Automates everything from the previous slides
- Framework software to run mixed precision automatically
- Details vary by framework, but core idea is the same
- Automatic Mixed Precision (AMP) does two things:

AUTOMATIC LOSS SCALING

Wraps the optimizer in order to:

- scale loss value & unscale gradients
- adjust scale & skip on gradient overflow

AUTOMATIC CASTING

Wraps model and operations in order to:

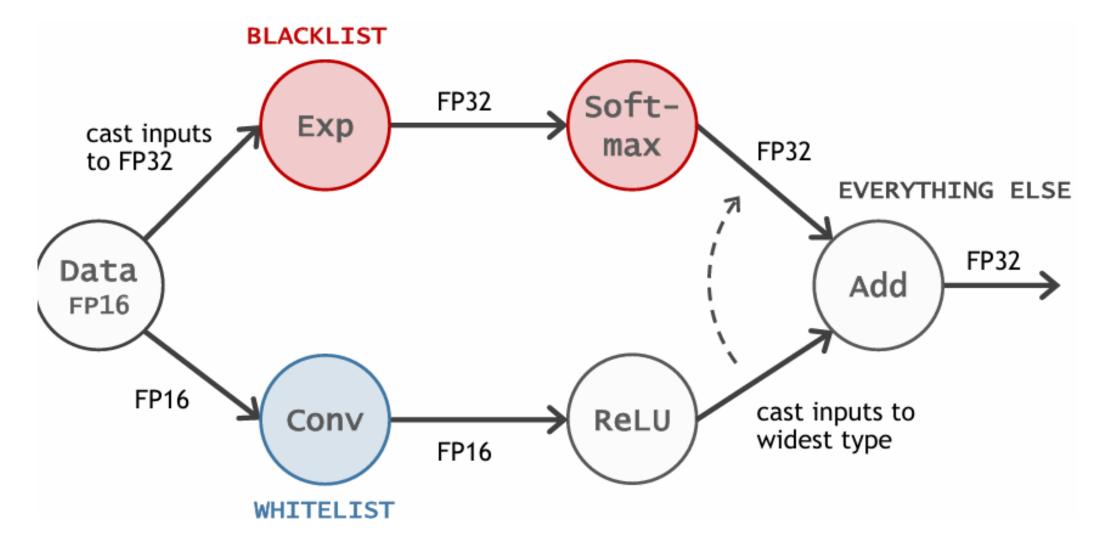
- cast data to FP16
- switch everything to run on FP16
- keep certain operations in FP32
- keep master copy of weights in FP32

- Ι
- Make type decisions for each operation a prior (static graph) or at runtime (eager execution)
- Define conservative set of rules to replace "by-hand" mixed precision

Divide the universe of operations into three kinds:

WHITELIST	BLACKLIST	EVERYTHING ELSE
FP16 enables Tensor Cores.	FP32 is needed for accuracy.	Can run in FP16, but only if inputs already in FP16.
e.g. linear, bmm, convs	e.g. loss, exp, sum, softmax	e.g. relu, add, maxpool
Rule: always run in FP16, cast if necessary.	Rule: always run in FP32, cast if necessary.	Rule: run in existing input type.







To control the operations being casted

model, optimizer = amp.initialize(model, optimizer, opt_level="01")

00	01	02
FP32 Training	Mixed Precision Training	FP16 Training
Leave everything in FP32.	FP16 whitelist and FP32 blacklist ops.	FP16 model/data with FP32 batchnorm.



Questions?

53 GRAINGER ENGINEERING

COMPUTER SCIENCE