Imperial College London

Multi-Precision Policy Enforced Training (MuPPET)

A precision-switching strategy for quantised fixed-point training of CNNs

Aditya Rajagopal, Diederik Adriaan Vink
Stylianos I. Venieris, Christos-Savvas Bouganis

intelligent Digital Systems Lab
Dept. of Electrical and Electronic Engineering

www.imperial.ac.uk/idsl

SAMSUNG AI Center
- Cambridge

Training of Convolutional Neural Networks (CNNs)

Typical Datasets

- **CIFAR10**
  - 10 categories
  - 60000 images
- **CIFAR100**
  - 100 categories
  - 60000 images
- **ImageNet Dataset**
  - 1000 categories
  - 1.2 million images

Typical Networks

Architecture	# Parameters (million)	ILSVRC12 Top-1 Accuracy (%)
AlexNet[1]	60.0	59.3
GoogLeNet[2]	6.80	64.0
ResNet18[3]	11.0	69.5
NASNet-A[4]	88.9	82.7
AmoebaNet-A[5]	469	83.9

Motivation

Training Time

- **Enable wider experimentation** with training e.g. Neural Architecture Search
- **Increase productivity** of deep learning practitioners

Power Consumption

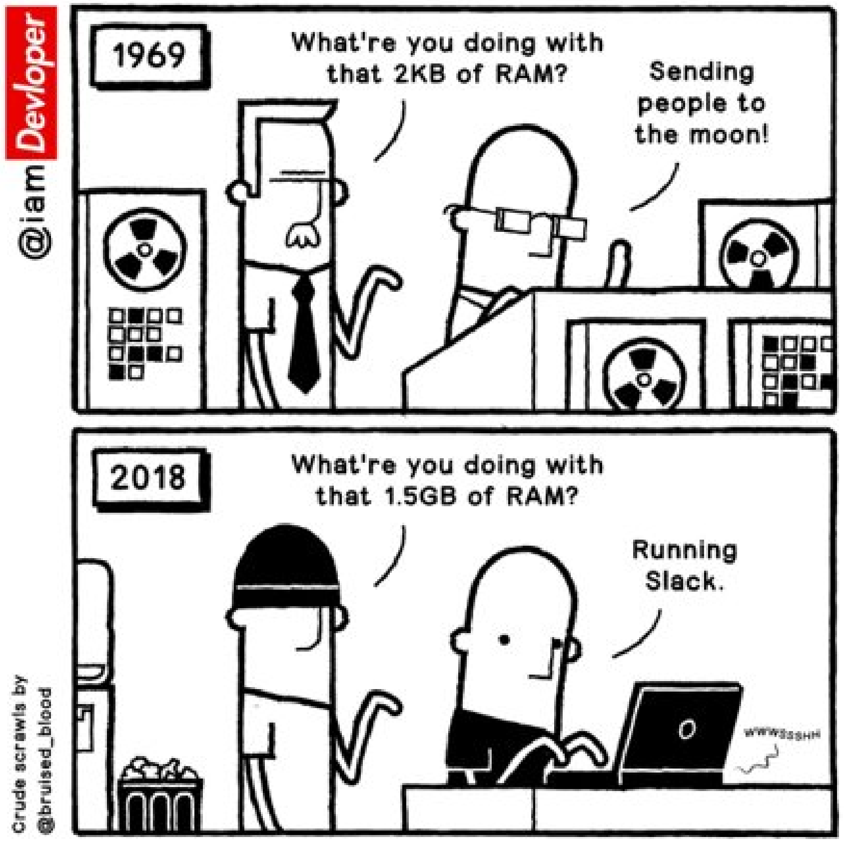
- **Reduce cost** of training in large data centers
- Perform training on **edge devices**

Exploit low-precision hardware capabilities

- NVIDIA Turing Architecture **(GPU)**
- Microsoft Brainwave **(FPGA)**
- Google TPU **(ASIC)**

Goal

Perform **quantised training** of CNNs while **maintaining FP32 accuracy** and producing a model that performs **inference at FP32**



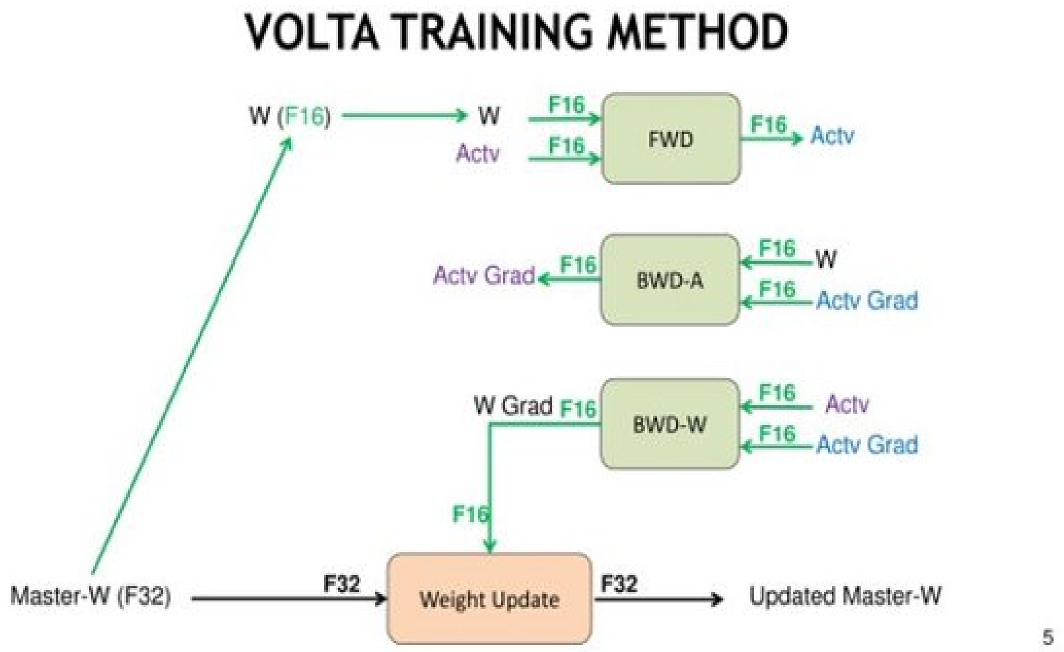
Contributions of this paper

- **Generalisable policy** that decides at **run time** appropriate points to increase the precision of the training process **without impacting** final test accuracy
  - **Datasets**: CIFAR10, CIFAR100, ImageNet
  - **Networks**: AlexNet, ResNet, GoogLeNet
  - **Up to 1.84x** training time improvement with **negligible loss** in accuracy
- **Extending** training to bit-widths **as low as 8-bit** to leverage the low-precision capabilities of modern processing systems
- **Open source PyTorch** implementation of the MuPPET framework with emulated quantised computations

Background: Mixed Precision Training

- **Current state-of-the-art:** Mixed-precision training (Micikevicius et al., 2018)
  - Maintains **master copy** of the weights at FP32
  - Quantises **weights and activations** to FP16 for all computations
  - Accumulates **FP16 gradients** into FP32 master copy of the weights
- Incurs **accuracy drop** if precision **below FP16** is utilised

[6]



[6] Micikevicius, P. et.al.. Mixed Precision Training. In International Conference on Learning Representations (ICLR), 2018

Multilevel optimisation formulation

- Hierarchical formulation that **progressively increases** precision of computations

$$\min_{w^{(q^N)} \in \mathbb{R}^D} Loss\left(f\left(w^{(q^N)}\right)\right)$$

FP32 Master Copy of Weights

$$\min_{w^{(q^{N-1})} \in \mathbb{R}^D} Loss\left(f\left(w^{(q^{N-1})}\right)\right)$$

FP32 Master Copy of Weights

$$\min_{w^{(q^{N-2})} \in \mathbb{R}^D} Loss\left(f\left(w^{(q^{N-2})}\right)\right)$$

$$\vdots$$

$$\min_{w^{FP32} \in \mathbb{R}^D} Loss\left(f\left(w^{FP32}\right)\right)$$

Proposed policy decides at **run time** the epochs at which these changes need to be made

Background: Gradient Diversity

- Yin et al. 2018 computes diversity between minibatches **within an epoch**

Gradient of weights for **minibatch i**

$$\Delta_S(w) = \frac{\sum_{i=1}^{n} \|\nabla f_i(w)\|_2^2}{\left\|\sum_{i=1}^{n} \nabla f_i(w)\right\|_2^2} = \frac{\sum_{i=1}^{n} \|\nabla f_i(w)\|_2^2}{\sum_{i=1}^{n} \|\nabla f_i(w)\|_2^2 + \sum_{i \neq j} \langle \nabla f_i(w), \nabla f_j(w) \rangle}$$

- Modified for MuPPET to compute diversity between minibatches **across epochs**

Gradient of **last** minibatch for **layer l** in **epoch k**

$$\Delta_S(w)^j = \frac{1}{|\mathcal{L}|} \sum_{\forall l \in \mathcal{L}} \frac{\sum_{k=j-r}^{j} \|\nabla f_l^k(w)\|_2^2}{\left\|\sum_{k=j-r}^{j} \nabla f_l^k(w)\right\|_2^2}$$

Resolution of **r** epochs ≤ **epoch j**

Average gradient diversity across all layers from last **r** epochs

Precision Switching Policy: Methodology

- Every **r** epochs:
  - The **inter-epoch gradient diversity** $\Delta_S(w)^j$ is calculated
  - Given an epoch **e** when the precision switched from level q^{n-1} to q^n, and current epoch **j**

$$S(j) = \{\Delta_S(w)^i \; \forall e \le i \le j\}$$

$$p = \frac{\max S(j)}{\Delta_S(w)^j}$$

- Empirically chosen **decaying threshold** placed on **p:** $T = \alpha + \beta e^{-\lambda j}$

- If **p** violates **T** more than γ times, a precision switch is triggered and $S(j) = \emptyset$

Precision Switching Policy: Hypotheses

- **Intuition**
  - Low gradient diversity increases value of **p**
  - The likelihood of observing **r** gradients across **r** epochs that have **low diversity** at **early stages** of training is **low**
  - If this happens, may imply that information is being lost due to quantisation (**high p value**)

- **Generalisability**

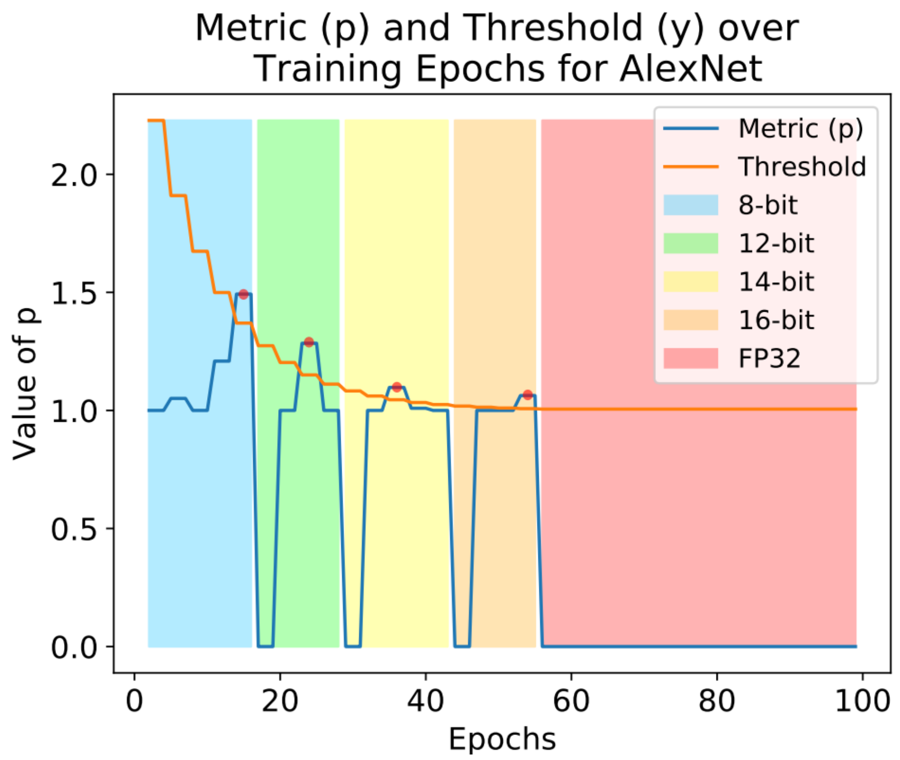
Generalisability across epochs

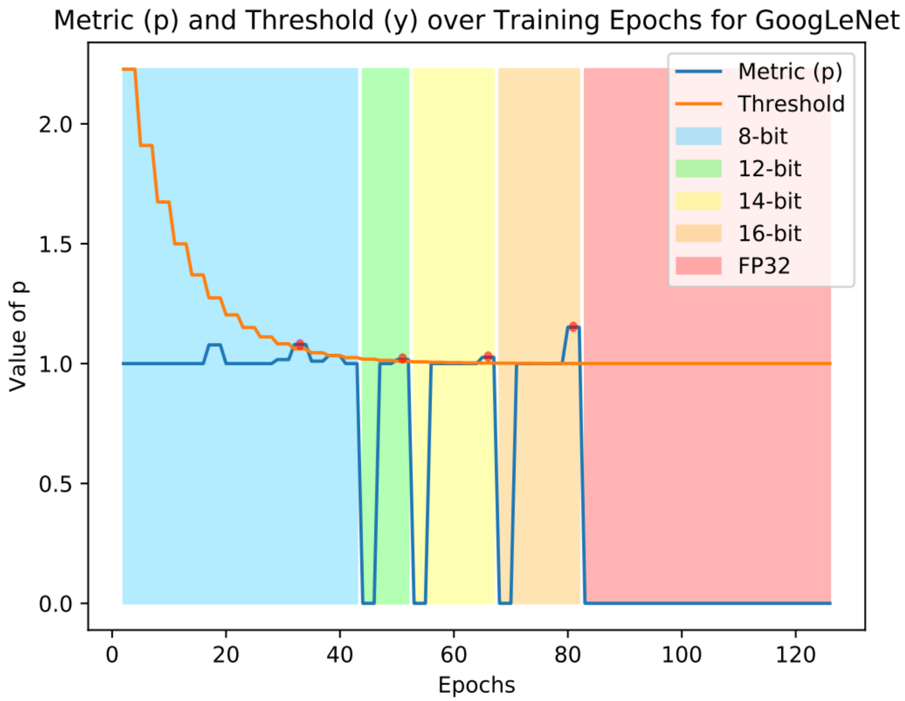
$$p = \frac{\max S(j)}{\Delta_s(\boldsymbol{w})^j}$$

Generalisability across networks and datasets

Precision Switching Policy: Generalisability

- **Similar values** across various networks and datasets
- Decaying threshold accounts for **volatility** in early stages of training

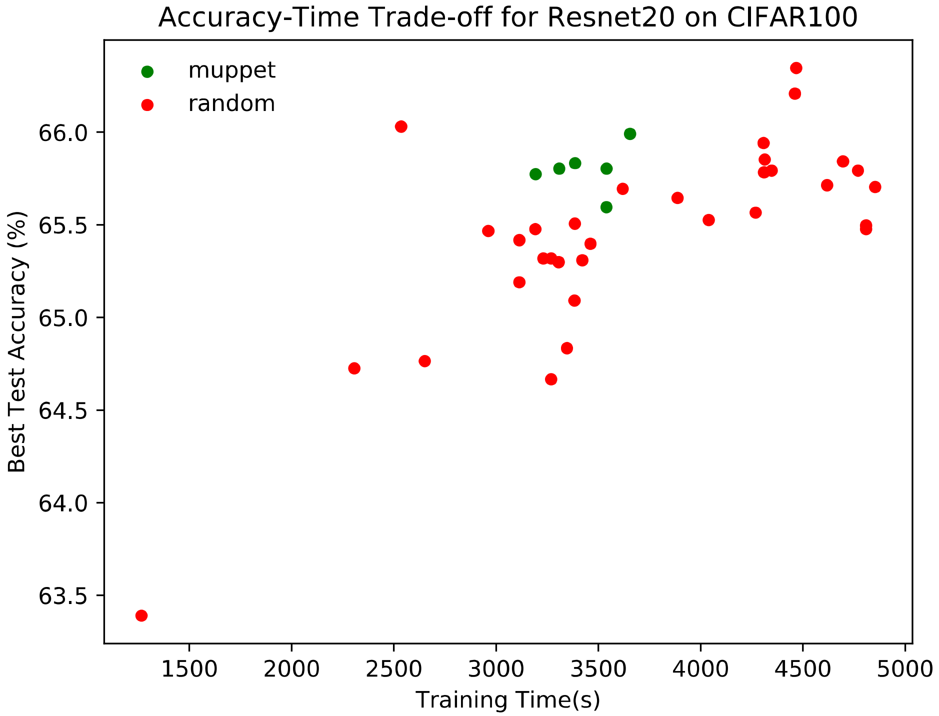




Imperial College London

Precision Switching Policy: Adaptability

- Is it better than randomly switching?



- Does it tailor to network and dataset?

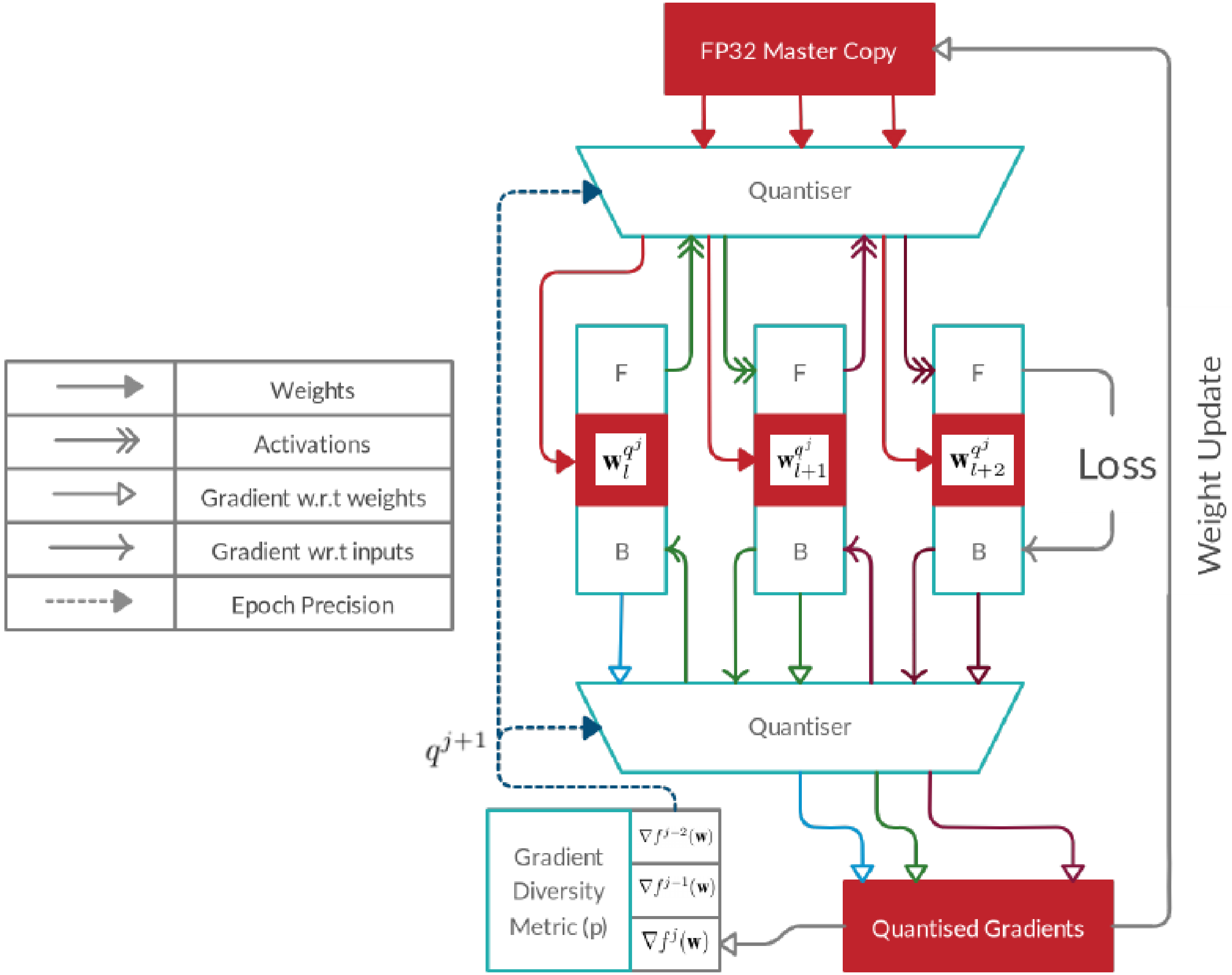
	ResNet20		GoogLeNet	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
ResNet20	65.01	**65.80**	-	65.0
GoogLeNet	-	64.00	64.70	**65.70**

Precision Switching Policy: Performance (Accuracy)

- **Nets**
  - AlexNet, ResNet18/20, GoogLeNet
- **Datasets**
  - CIFAR10, CIFAR100 (Hyperparameter Tuning), ImageNet (Application)
- **Precisions**
  - 8-, 12-, 14-, 16-bit **Dynamic Fixed**-Point (Emulated) and 32-bit **Floating**-Point
- Training with MuPPET **matches accuracy** of standard FP32 training when trained with **identical SGD hyperparameters**

	CIFAR-10			**CIFAR-100**			**ImageNet**		
	FP32	MuPPET	Diff (pp)	FP32	MuPPET	Diff (pp)	FP32	MuPPET	Diff (pp)
AlexNet	75.45	74.49	-0.96	39.20	38.19	-0.99	56.21	55.33	-0.88
ResNet	90.08	90.86	0.78	64.60	65.80	1.20	69.48	69.09	-0.39
GoogLeNet	89.23	89.47	0.24	62.90	65.70	2.80	59.15	63.70	4.55

Quantised Training



Quantisation

Quantised signed INT

$$x_{quant}^{\{weights,act\}} = \lfloor x^{\{weights,act\}} \cdot ¿¿$$

Original value

Quantisation

Imperial College London

$$x_{quant}^{\{weights,act\}} = \lfloor x^{\{weights,act\}} \cdot ¿¿$$

Quantised signed INT

Original value

Quantisation

Quantised signed INT

$$x_{quant}^{\{weights,act\}} = \lfloor x^{\{weights,act\}} \cdot 2^{s^{\{weights,act\}}} + Unif(-0.5, 0.5) \rceil$$

Original value

$$s^{\{weights,act\}} = \left\lfloor \log_2 \left(\min \left(\frac{UB + 0.5}{X_{max}^{\{weights,act\}}}, \frac{LB - 0.5}{X_{min}^{\{weights,act\}}} \right) \right) \right\rfloor$$

Activations

Weights

Scale factor

Quantisation

Quantised signed INT

$$x_{quant}^{\{weights,act\}} = \lfloor x^{\{weights,act\}} \cdot ¿¿$$

Original value

Wordlength lower bound

Wordlength upper bound

$$s^{\{weights,act\}} = \left\lfloor \log_2 \left(\min \left(\frac{UB+0.5}{X_{max}^{\{weights,act\}}}, \frac{LB-0.5}{X_{min}^{\{weights,act\}}} \right) \right) \right\rfloor$$

Activations

Scale factor

Weights

Quantisation

Quantised signed INT

$$x_{quant}^{\{weights,act\}} = \lfloor x^{\{weights,act\}} \cdot ¿¿$$

Original value

Wordlength lower bound

Wordlength upper bound

$$s^{\{weights,act\}} = \left\lfloor \log_2\left(\min\left(\frac{UB+0.5}{X_{max}^{\{weights,act\}}}, \frac{LB-0.5}{X_{min}^{\{weights,act\}}}\right)\right)\right\rfloor$$

Activations

Weights

Scale factor

Quantisation configuration

$q_l^i = < WL^{net}, s_l^{weights}, s_l^{act} >^i \quad \forall l \in \mathcal{L}$ and $q^i = \langle q_l^i \mid \forall l \in \mathcal{L} \rangle$

Quantisation

Quantised signed INT

$$x_{quant}^{\{weights,act\}} = \lfloor x^{\{weights,act\}} \cdot 2^{s^{\{weights,act\}}} + Unif(-0.5, 0.5) \rceil$$

Original value

Wordlength lower bound

Wordlength upper bound

$$s^{\{weights,act\}} = \left\lfloor \log_2 \left(\min \left(\frac{UB+0.5}{X_{max}^{\{weights,act\}}}, \frac{LB-0.5}{X_{min}^{\{weights,act\}}} \right) \right) \right\rfloor$$

Activations

Weights

Scale factor

Quantisation configuration

Network layers

Optimisation level

$$q_l^i = < WL^{net}, s_l^{weights}, s_l^{act} >^i \quad \forall l \in \mathcal{L} \text{ and } q^i = \langle q_l^i | \forall l \in \mathcal{L} \rangle$$

Network word length

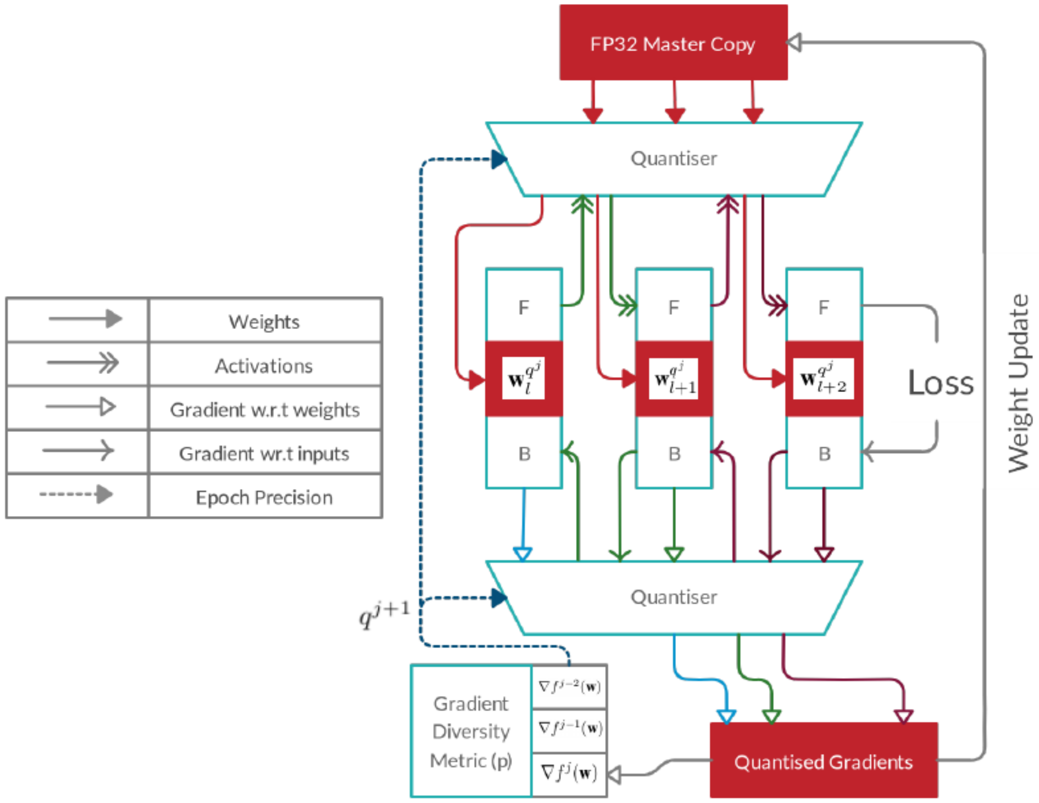
Current layer

Quantisation Emulation

- **No ML framework support** for reduced precision hardware
  - e.g. NVIDIA Turing architecture
- GEMM profiled using **NVIDIA's CUTLASS** Library
- Training profiled through PyTorch
  - Quantisation of weights, activations and gradients
  - All gradient diversity calculations
- 12- and 14-bit fixed profiled as 16-bit fixed point
  - Included for **future** custom precision hardware

Performance (Wall-clock time)

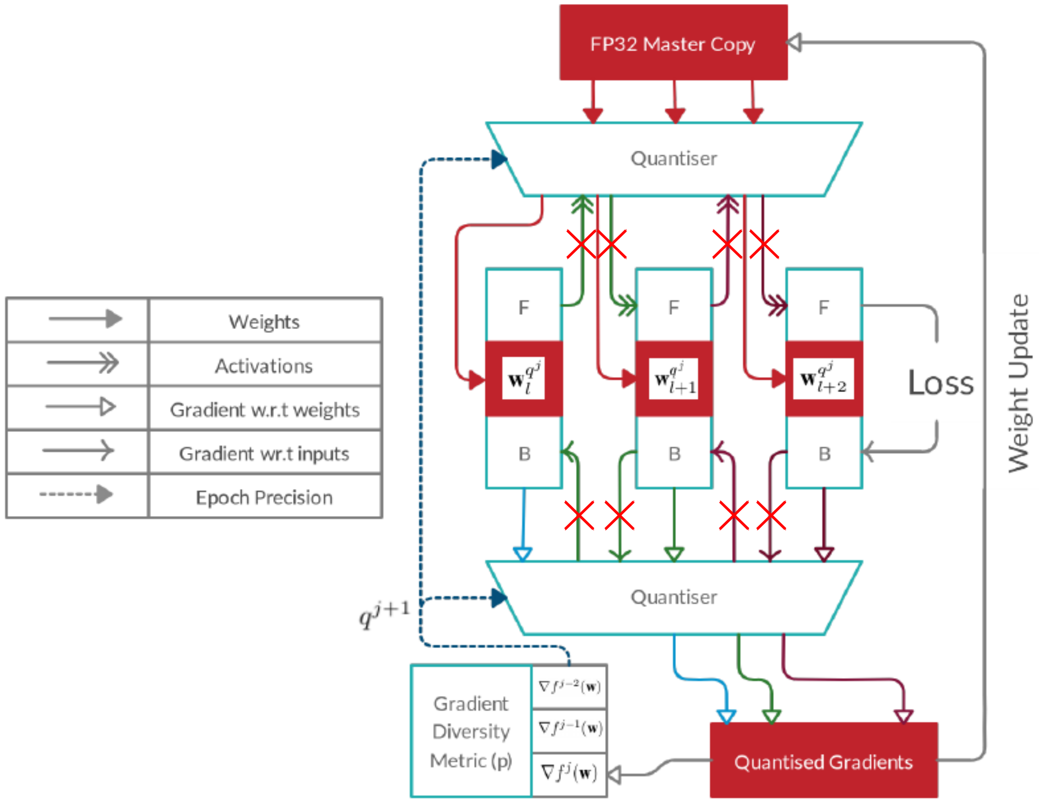
Current Implementation



	FP32 (Baseline)	Mixed Prec (Micikevicius et al., 2018)	MuPPET (Current Impl.)	MuPPET (Ideal)
AlexNet	30:13 (1×)	29.20 (1.03×)	23:52 (1.27×)	20:25 (1.48×)
ResNet18	132:46 (1×)	97:25 (1.36×)	100:19 (1.32×)	92:43 (1.43×)
GoogLeNet	152:28 (1×)	122:51 (1.24×)	122:13 (1.25×)	82:38 (1.84×)

Performance (Wall-clock time)

Ideal



	FP32 (Baseline)	Mixed Prec (Micikevicius et al., 2018)	MuPPET (Current Impl.)	MuPPET (Ideal)
AlexNet	30:13 (1×)	29.20 (1.03×)	23:52 (1.27×)	20:25 (1.48×)
ResNet18	132:46 (1×)	97:25 (1.36×)	100:19 (1.32×)	92:43 (1.43×)
GoogLeNet	152:28 (1×)	122:51 (1.24×)	122:13 (1.25×)	82:38 (1.84×)