ODTÜ METU

TÜBİTAK

EURO 4SEE

Optimizing Deep Learning Systems for Hardware

Assoc. Prof. Erdem AKAGÜNDÜZ, METU

ncc@ulakbim.gov.tr

Part IV : System-level Optimization

- PIV.a : Parallelism
- PIV.b : Mixed-Precision Training
- PIV.c : Other

Mixed-Precision Training

- Use lower-precision data types (e.g., FP16 or BF16) for parts of model training instead of full 32-bit (FP32) floats.
- Combines:
  - FP16/BF16 for compute-intensive operations
  - FP32 for numerically sensitive operations (e.g., loss scaling, weight updates)
- You need Tools/frameworks:
  - NVIDIA AMP (Automatic Mixed Precision),
  - PyTorch native amp,
  - TensorFlow mixed_precision API.

Mixed-Precision Training

- Use lower-precision data types (e.g., FP16 or BF16) for parts of model training instead of full 32-bit (FP32) floats.
- Combines:
  - FP16/BF16 for compute-intensive operations
  - FP32 for numerically sensitive operations (e.g., loss scaling, weight updates)
- You need Tools/frameworks:
  - NVIDIA AMP (Automatic Mixed Precision),
  - PyTorch native amp,
  - TensorFlow mixed_precision API.

Is it system-level really?

- When we quantized models it was model level?
- Yes. This time it is different.

Model-Level	**System-Level**
Changes the network architecture	**Keeps model architecture the same**
Requires ML domain knowledge	**Requires hardware/runtime knowledge**
Quantization Frameworks	**Mixed-Precision Frameworks**

Is it system-level really?

- Mixed-precision training does not alter the model structure or algorithm.
- It's an implementation detail handled by the training framework and hardware stack.

Model-Level	**System-Level**
Changes the network architecture	**Keeps model architecture the same**
Requires ML domain knowledge	**Requires hardware/runtime knowledge**
Quantization Frameworks	**Mixed-Precision Frameworks**

Benefits

- Faster Training Throughput
  - FP16 and BF16 operations are up to 2x–8x faster on modern GPUs (e.g., NVIDIA Tensor Cores)
- Lower Memory Footprint
  - Reduced precision cuts memory usage nearly in half
  - Enables larger batch sizes, deeper models
- Better Hardware Utilization
  - Takes advantage of specialized hardware (e.g., Tensor Cores on NVIDIA, BF16 on TPUs)

How it works:

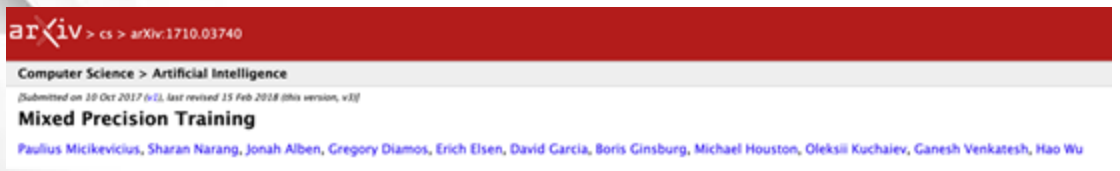
- Handled by Compiler/runtime stack
  - GPU drivers and libraries
  - Training loop wrappers — *not by the model author*


Model remains unchanged


System handles casting, scaling, kernel selection


Benefit comes from exploiting hardware and software stack

How it works:



4SEE

- This diagram illustrates how mixed-precision training works for a single layer during a training iteration. It shows:
  - Which parts of the computation use FP16 (half precision)
  - Which parts are kept in FP32 (full precision)
  - How data flows between forward pass, backward pass, and weight update steps

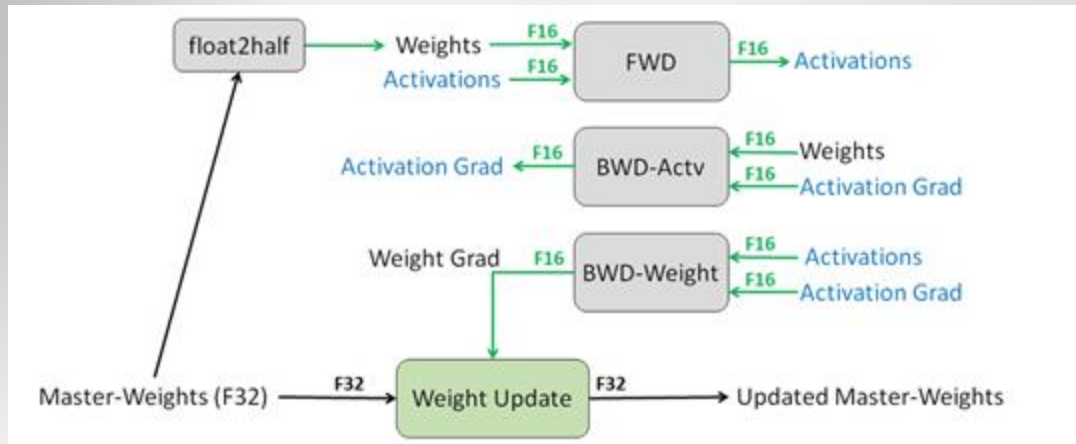
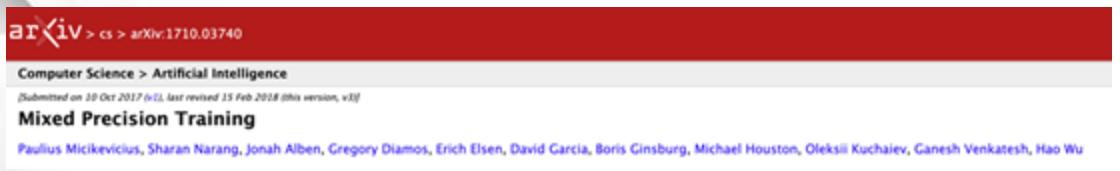


Figure 1: Mixed precision training iteration for a layer.

Components



- Master Weights (F32)
  - The main copy of the model's weights is stored in FP32.
  - These are not used directly for forward/backward passes, but only for updates.
  - They are converted to FP16 before being used for computation.

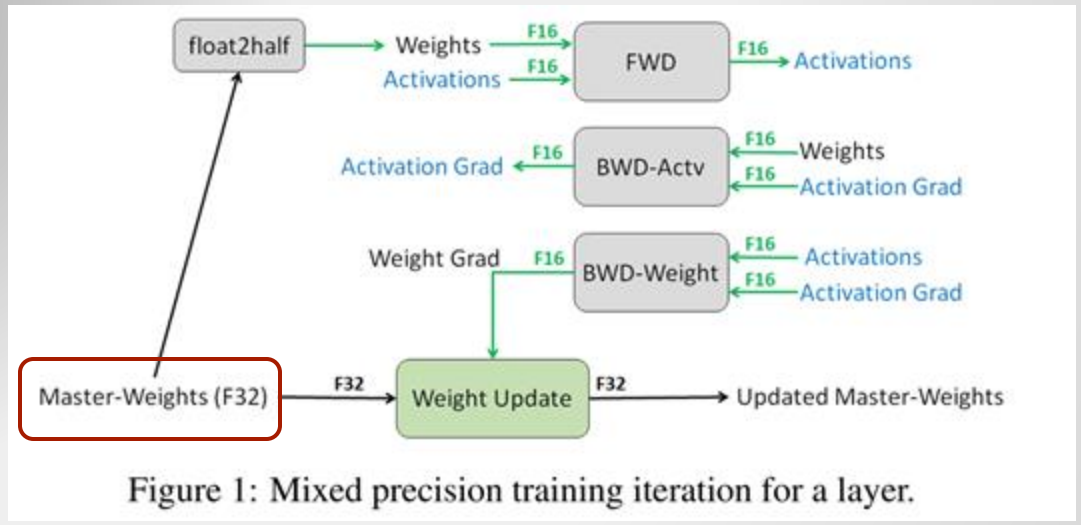
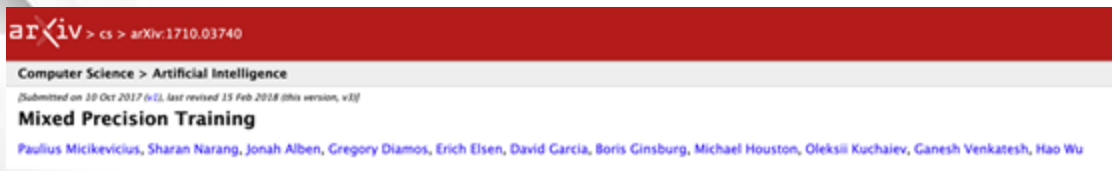


Figure 1: Mixed precision training iteration for a layer.

Components



- float2half
  - This block converts FP32 master weights to FP16 weights.
  - These FP16 weights are then used in the forward and backward passes.

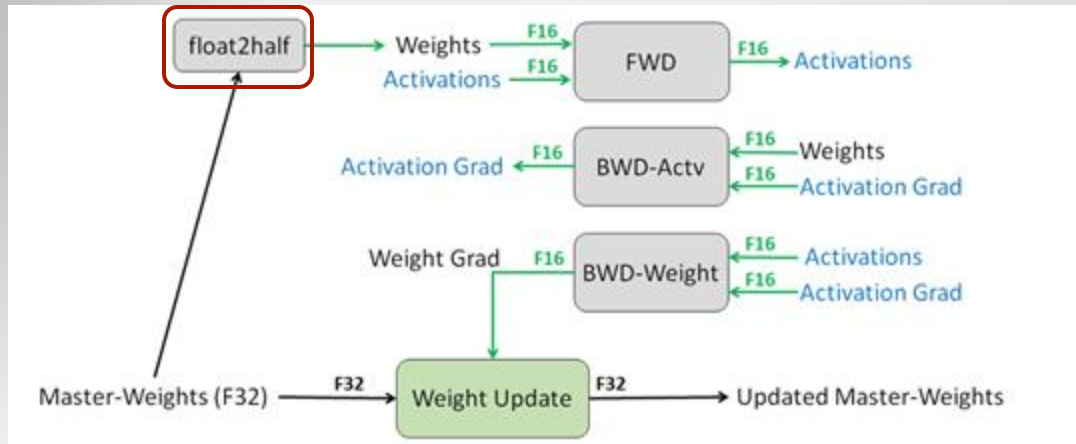
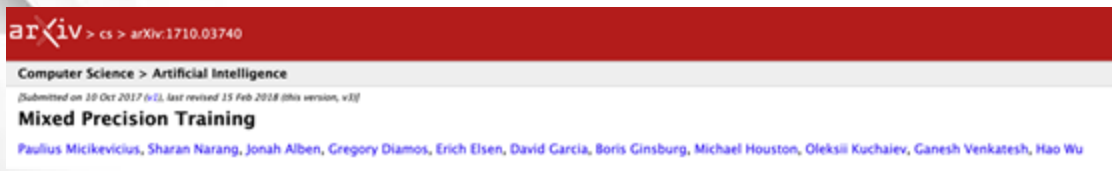


Figure 1: Mixed precision training iteration for a layer.

Components



- FWD (Forward Pass)
  - Inputs: FP16 weights and FP16 activations
  - Outputs: FP16 activations
- This is where the model computes the layer outputs from inputs.
- All done in FP16 for speed and lower memory usage.

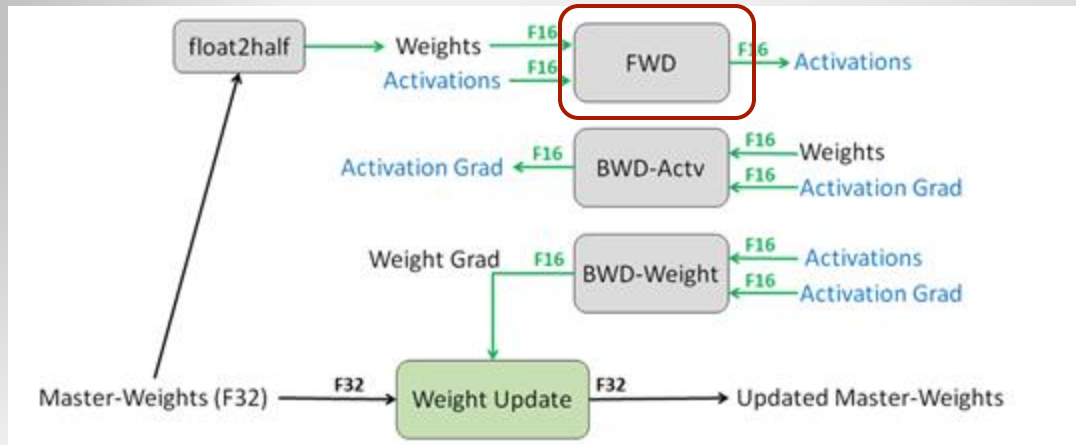
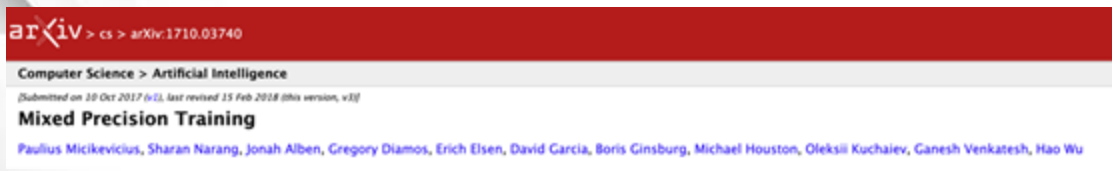


Figure 1: Mixed precision training iteration for a layer.

Components



- BWD-Actv (Backward Pass – Activation Gradient)
  - Computes gradients of the loss w.r.t. activations.
  - Takes in:
    - FP16 weights
    - FP16 gradients of outputs (activation grad)
  - Outputs:
    - FP16 gradients of inputs

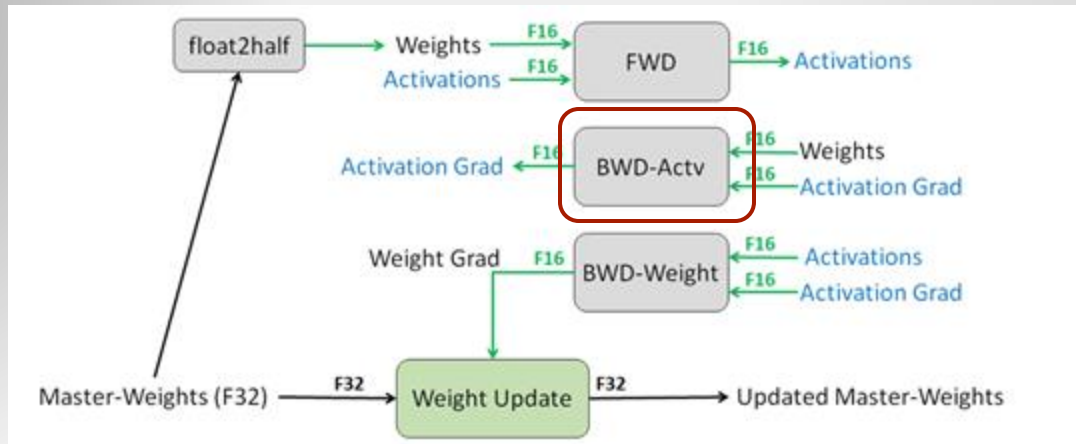
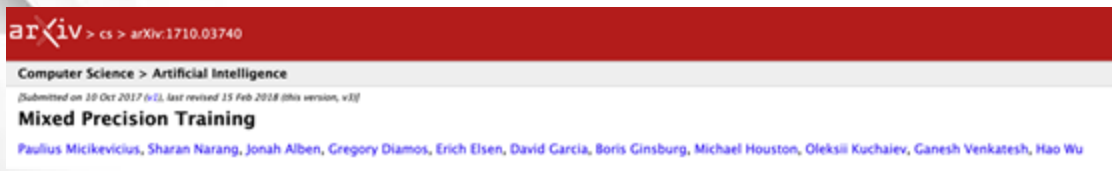


Figure 1: Mixed precision training iteration for a layer.

Components



- BWD-Weight (Backward Pass – Weight Gradient)
  - Computes gradients w.r.t. weights.
  - Takes in:
    - FP16 activations (from FWD)
    - FP16 activation gradients
  - Outputs:
    - FP16 weight gradients

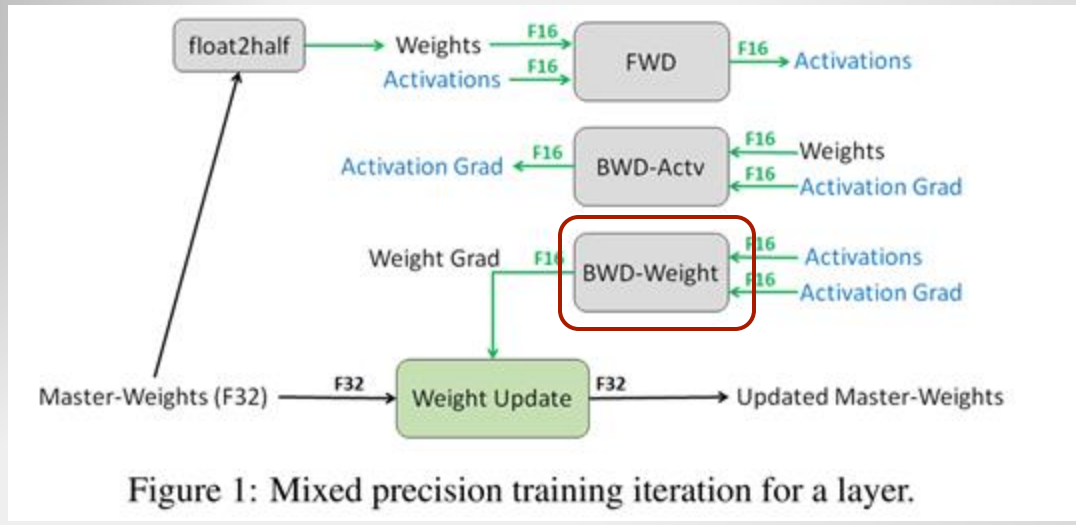
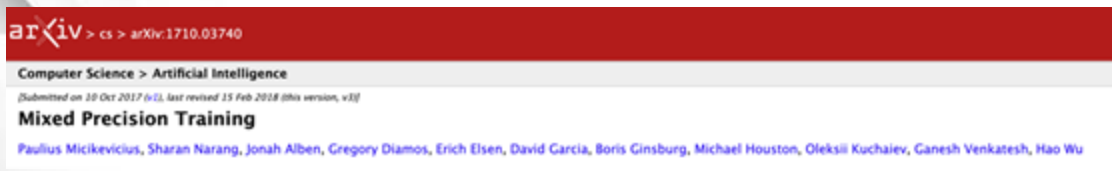


Figure 1: Mixed precision training iteration for a layer.

Components



- Weight Update
  - The FP16 weight gradients are converted to FP32 and used to update the FP32 master weights.
  - The update is performed in full precision for numerical stability.
  - The updated weights are then reused in the next iteration.

Figure 1: Mixed precision training iteration for a layer.

Components

arXiv > cs > arXiv:1710.03740

Computer Science > Artificial Intelligence

[Submitted on 10 Oct 2017 (v1), last revised 15 Feb 2018 (this version, v3)]

Mixed Precision Training

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, Hao Wu

- Task: English-to-French translation
  - Model: 3-layer LSTM with attention (3×1024 LSTM)
- Comparison between:
  - Full precision (FP32) training runs (ref1, ref2, ref3)
  - Mixed-precision training with two different loss scaling values

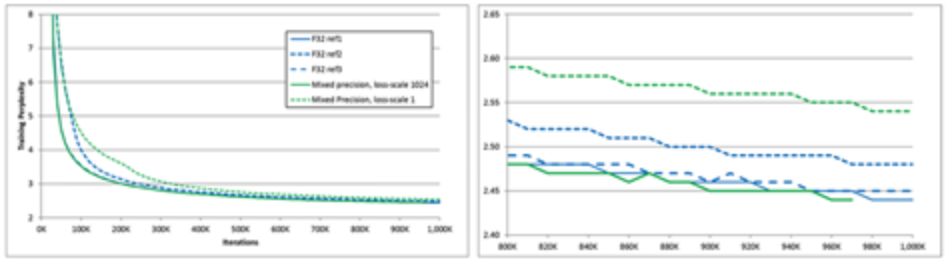
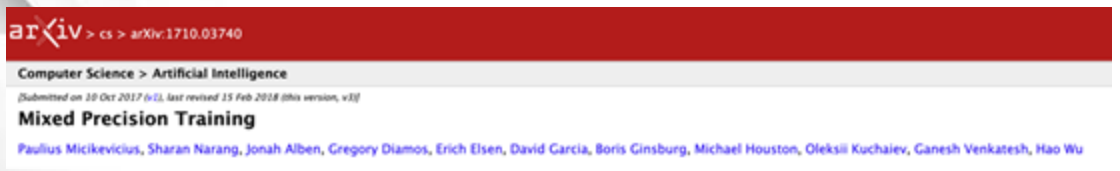


Figure 4: English to French translation network training perplexity, 3x1024 LSTM model with attention. Ref1, ref2 and ref3 represent three different FP32 training runs.

Components



- Left: Training perplexity over iterations
  - Mixed-precision (green solid line, loss scale = 1024) shows faster convergence than FP32
  - Poorer performance without proper loss scaling (green dashed line, scale = 1)
- Right Zoom-In: Final training perplexity
  - Mixed-precision (green solid) matches or slightly outperforms FP32 runs
  - Proper loss scaling is essential for stability and accuracy

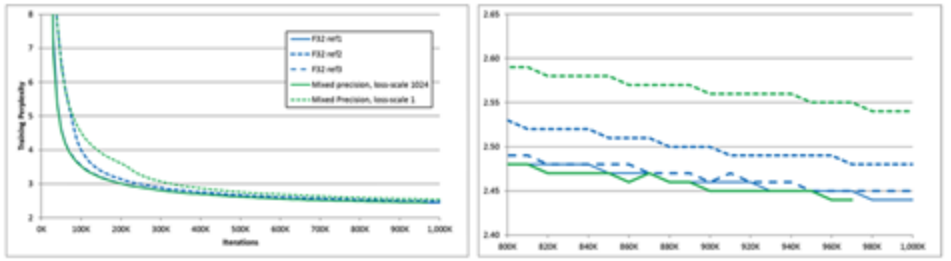


Figure 4: English to French translation network training perplexity, 3x1024 LSTM model with attention. Ref1, ref2 and ref3 represent three different FP32 training runs.

Next: Part IV.c

- PIV.a : Parallelism
- PIV.b : Mixed-Precision Training
- PIV.c : Other

EURO 4SEE

Thanks!

Co-funded by
the European Union

EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101191697. The JU receives support from the Digital Europe Programme and Germany, Türkiye, Republic of North Macedonia, Montenegro, Serbia, Bosnia and Herzegovina.