Dynamic Precision Tunability in Low-Power Training of Deep Neural Networks

Shayan Moini, Matthew Caswell, and Wayne Burleson

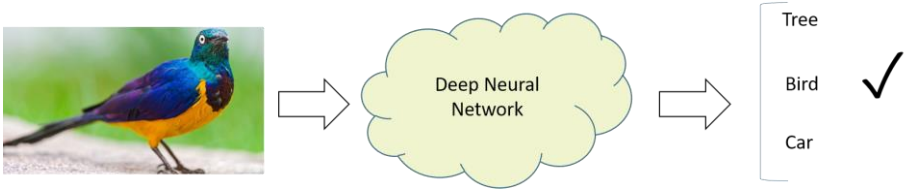
Department of Electrical and Computer Engineering

University of Massachusetts Amherst

January 2018
Partially Supported by NSF Grant # 1619558

Background

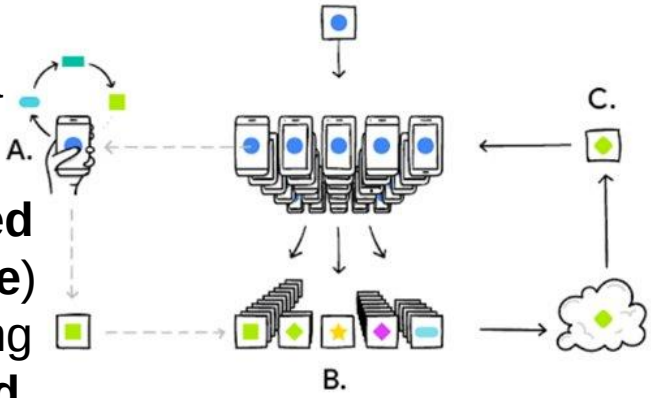
- Deep Neural Networks used in many ML applications



- DNN Training and Inference computationally heavy
  - AlexNet performs 7e^8 operations for inference per image
- High parallelism capabilities can be used in parallel platforms to accelerate DNN operations
  - Multi-CPU, GP-GPU, ASIC, FPGA
- A large community of research to accelerate inference of trained networks
  - Quantization, pruning, weight compression

Efficient Training

- Deep network training typically occurs in the cloud, offline, with a cluster of powerful GPUs
- But the recent trend of **distributed training** and **incremental** (**online**) **learning** shows a need for training algorithms suitable for **embedded systems**



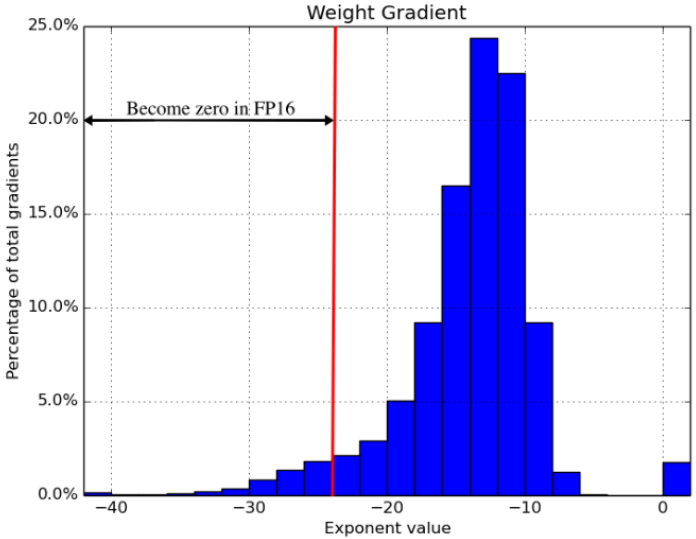
Google Federated Learning
(Source: https://bit.ly/2IHdmzw)

Beyond GPUs

- Training is orders of magnitude more complex than inference
  - Backpropagation with one million Images for AlexNet
- GPU is used for training due to high parallelism and software
- But GPU is power hungry -> Volta architecture up to 300w
- GPU traditionally supports floating point FP32
  - New Volta architecture supports FP16, INT8, etc for inference
- ASICs and FPGAs allow custom number representations

Challenges

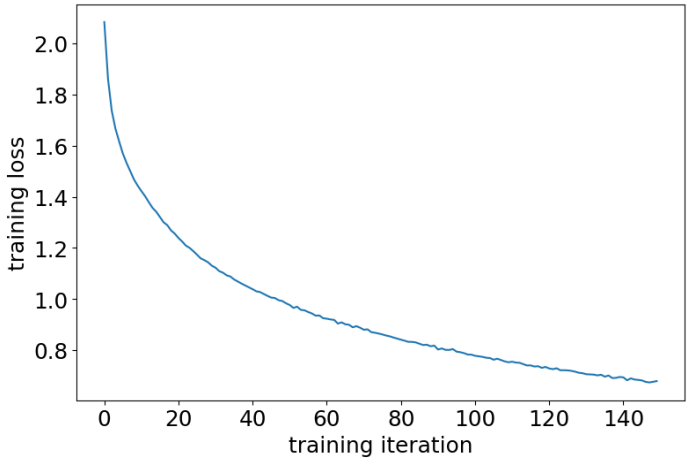
- Back propagation needs high dynamic range due to very small gradients.



Gradient Distribution in a speech recognition neural network, Baidu/Nvidia 2017[1]

Idea

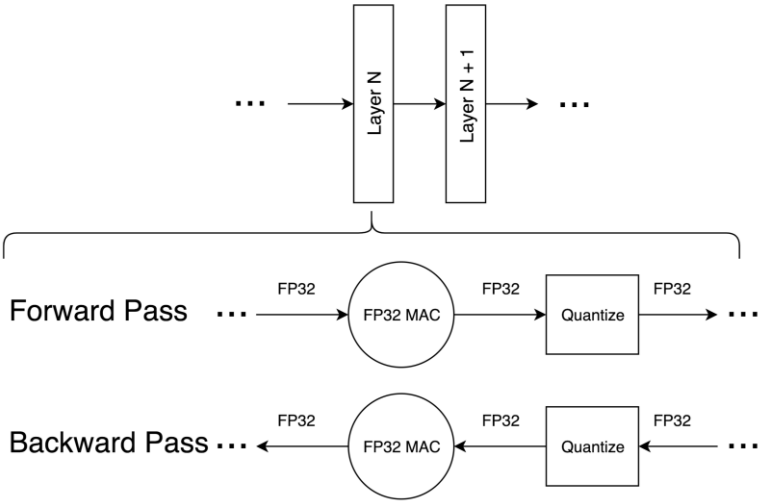
- Deeper neural networks generate smaller gradients
- As the training proceeds, the gradients get smaller
- Full precision is not needed during the whole process of training



Trend of loss during training with CIFAR-10

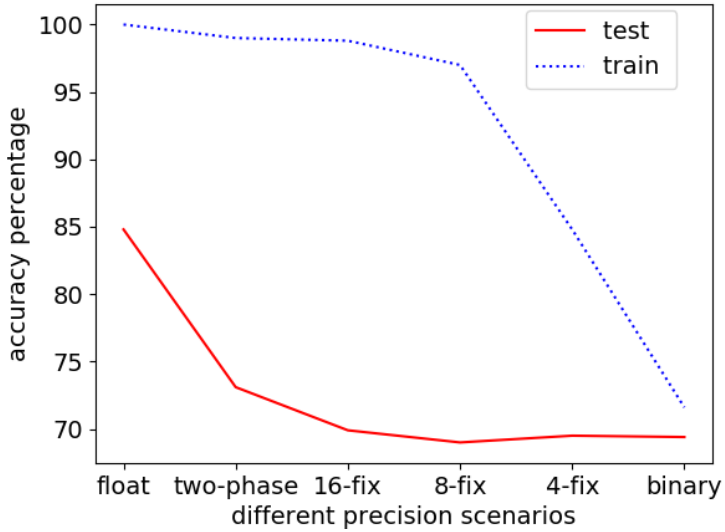
Experimental Methodology

- Finite Precision simulated on GPU with QuantizedNN tool, KU Leuven [2]
  - A tool written in TensorFlow and Keras with CIFAR-10 dataset
- Quantize the output of operations in each stage



Experiments

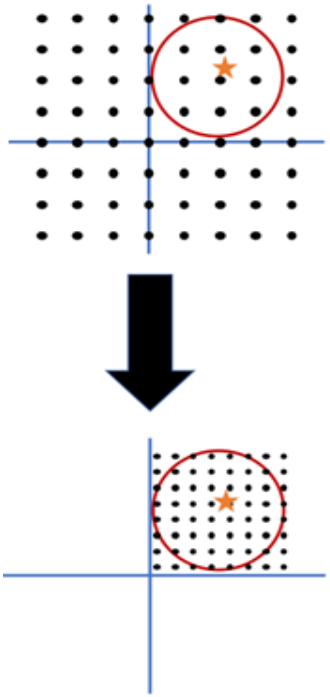
- Two-phase Fixed Point representation, 16 bit fixed in the first half and 32 bit fixed point in the second half



Two-phase training

Proposed Work

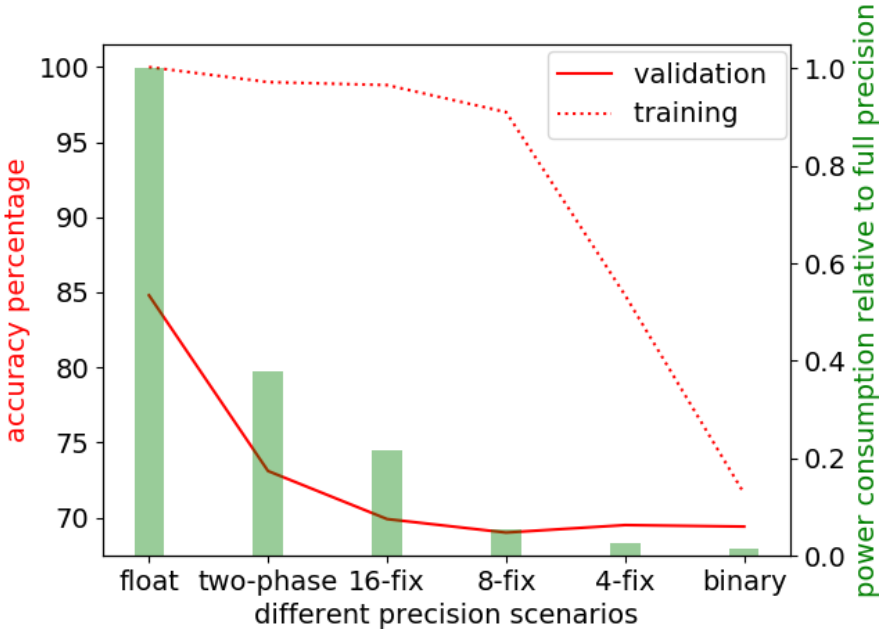
- Optimize where to switch to higher precision to save power and minimize impact on accuracy
  - Scale precision aggressively and use classification accuracy as a metric, GaTech 2016 [3]
- Monitor more statistically significant metrics
  - Standard deviation of first quantile, max/min, mean of gradients, etc.
- Alleviate the accuracy loss by other methods including high accuracy low precision training (HALP), Stanford 2018 [4]
  - Loss Scaling
  - Retain full-precision model for weight update



Loss Scaling [4]

Potential Benefits

- Relative Power Consumption for Different Representations in custom hardware (Horowitz 2014)



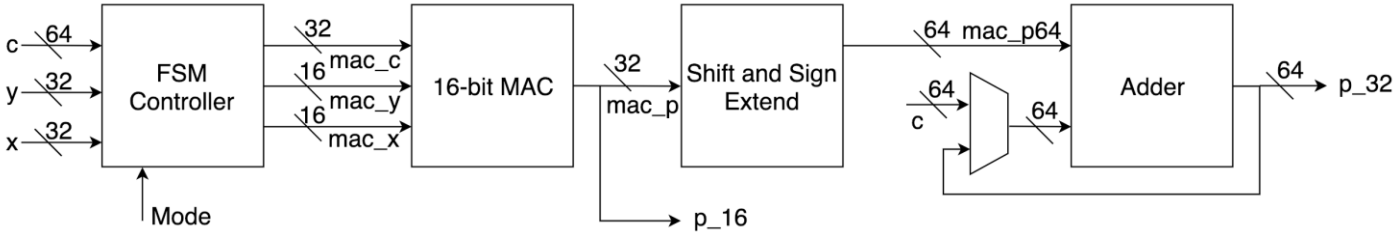
Relative Power Consumption of Different Scenarios

Potential Benefits

- Lower Memory Bandwidth for weights and activations
  - Proper design guarantees higher throughput when precision is low
  - Better usage of available on-chip memory with low precision
- Speed up the training process
  - Better computational resource usage
  - e.g. Theoretically, a hybrid MAC unit can perform two 16 bit operations or one 32 bit operation based on the target accuracy

Multiplier Architecture

- Need a MAC architecture that supports multiple precision configurations and can change on the fly
  - Lower precision configurations need to be faster than higher precision ones
- GaTech 2016 [3] created 16/32 bit MAC unit
  - Feeds through 16-bit MAC multiple times
  - Trades slightly increased power consumption and delay for flexibility



16/32 bit MAC [3]

Larger Datasets

- Fixed point is not suitable for deeper networks (e.g. AlexNet) on larger datasets (e.g. ImageNet)
- Block floating point is a possible replacement
  - Extending FlexPoint (Intel 2018 [5]) to shared exponent of 8 bits, variable length mantissa at different stages of training
- Enough dynamic range to support larger datasets
- Memory capacity and bandwidth can be reduced by 30%

eeeee.mmmmmmmmmmmmmmmm

eeeee.mmmmmmmm

Conclusion and Path Forward

- DNN training implementations have a lot of room for improvement
- Custom number representations can replace the full precision floating point with minimal accuracy loss
- Power consumption, memory bandwidth, and training speed can be improved by using dynamic precision number representation
- We are currently working on finding the best metrics for dynamic scaling methods
- We are also working on an FPGA Implementation for Dynamic Precision Training

References

[1] Paulius Micikevicius, et al. "Mixed precision training." *arXiv preprint arXiv:1710.03740, 2017*.

[2] Bert Moons. Quantized Neural Networks - networks trained for inference at arbitrary low precision. https://github.com/BertMoons/QuantizedNeuralNetworks-Keras-Tensorflow, 2017

[3] Taesik Na and Saibal Mukhopadhyay. Speeding up convolutional neural network training with dynamic precision scaling and flexible multiplier-accumulator. In Proceedings of the International Symposium on Low Power Electronics and Design. ACM, 2016

[4] Christopher De Sa, et al. High-Accuracy Low-Precision Training. arXiv preprint arXiv:1803.03383, 2018

[5] Urs Koster, et al. "Flexpoint: An adaptive numerical format for efficient training of deep neural networks." Advances in Neural Information Processing Systems, 2017

Thank You!

Questions?