DNN approximation for custom hardware

Anshul Bansal
Aayush Bansal

Q: Why we need DNN approximation?

→ To develop algorithms for DNN inference that

1. Reduce computational and storage cost
2. Increases throughput
3. Decreases latency

Two types:

1. Quantization
2. Weight Reduction

Quantization

→ reducing precision of the weights

- Floating point - Flexible in data representation range but expensive
- Floating point - predetermined precision

- Using fixed point, accuracy goes down merely by 1-2 pp.
- Data in different layers have have varied range hence constant quantization can provide sub optimal bandwidth efficiency.
- Block Floating Point - Dynamic fixed points.
- Adaptive Quantization - Each filter has independent precision.
- Binarization - Quantization of parameters into two values {1,-1}.
- Ternarization - {1,0,-1}.

- Fine Grained Quantization - Ternarization of Fp32 in group and then ternarizing each group independently.

- Logarithmic Quantization - Parameters are quantized in the powers of two with scaling factor .

Weight Reduction

→ removing redundant parameters

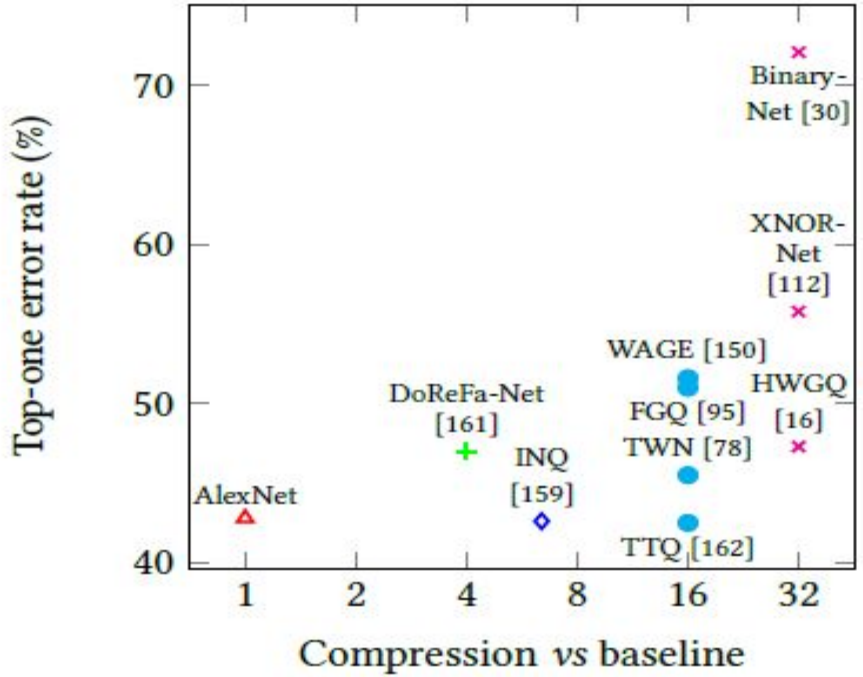
1. Pruning and structural simplification
2. Leads to reduction in number of activations

- Leads to workload reduction since it removes unnecessary parameters.

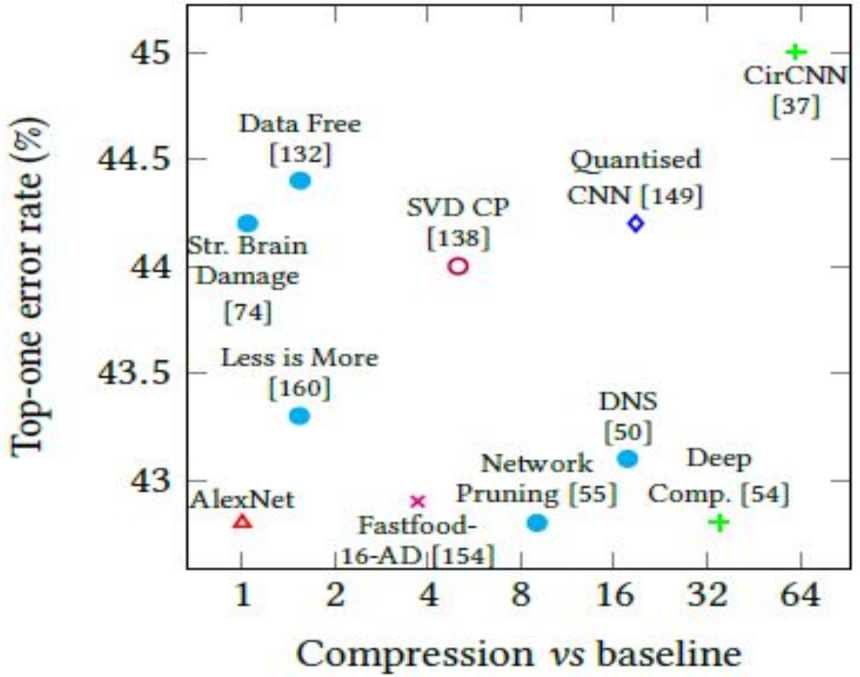
- Pruning - Similar neurons are wired together by removing redundant connections
- Iterative Pruning - Pruning followed by retraining , allowing remaining connections to learn to compensate for pruning loss.

- Weight Sharing - Groups Parameters into buckets reducing workload size.

- Structured matrices - Weight matrix represented as a structure of repeated pattern such that it can be expressed in fewer parameters.



(a) Quantisation methods: baseline (△), eight-bit fixed point (+), logarithmic (◇), ternary (●) and binary (×).



(b) Weight-reduction methods: baseline (△), hybrid (+), weight sharing (◇), pruning (●), structured matrix (×) and factorisation (○).

		Cheaper arithmetic operations	Memory reduction	Workload reduction
Quant-isation	Fixed-point representation	✔	✔	✘
	Binarisation and ternarisation	✔	✔	✘
	Logarithmic quantisation	✔	✔	✔ (if shift lengths are constant)
Weight reduction	Pruning	✘	✔	✔
	Weight sharing	✘	✔	✔ (if multiplications are precomputed)
	Low-rank factorisation	✘	✔	✔
	Structured matrices	✘	✔	✔
	Knowledge distillation	✘	✔	✔
	Input-dependent computation	✘	✘	✔
	Activation function approximation	✘	✘	✔
	Hybrid strategies	✔	✔	✔

Future Research Directions

1. Convergence Guarantees
2. Self Adaptive Hyper Parameter Fine Tuning
3. FPGA-ASIC Heterogeneous System