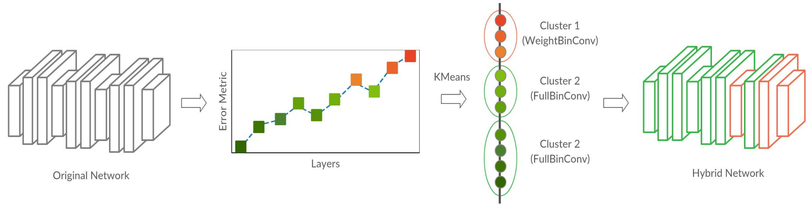
Hybrid Binary Networks

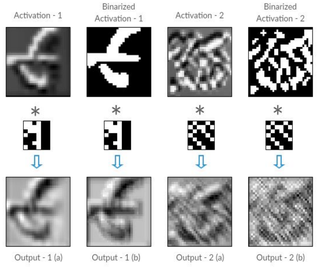
Ameya Prabhu, Vishal Batchu, Rohit Gajawada, Sri Aurobindo Munagala, Anoop Namboodiri
Center for Visual Information Technology, KCIS, IIIT Hyderabad, India

Introduction

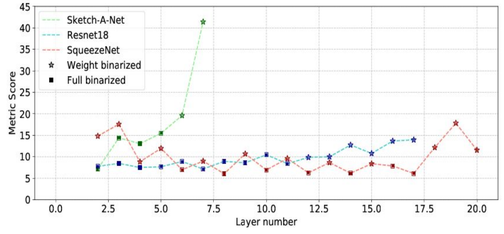
- Network Compression aims to bring down the size of the neural network, and attempts to speed up computations - to achieve the goal of making DNNs easy to run on mobile devices.
- Network Quantization involves quantizing layer weights/activations from floats to a discrete set of values. Binarization is the extreme form of it - allowing for 58x computational speedups through XNOR-Popcount operations, 16x compression, and reduced power consumption.
- Classical network binarization techniques such as XNOR-Nets or BWNs (Binary-Weight Networks) cause significant accuracy drops.
- We explore the question of *where* to binarize a network at layer-level granularity and show that selectively binarizing the inputs to specific layers in the network could lead to significant improvements in accuracy while preserving most of the advantages of binarization.
- We analyze the binarization tradeoff using a metric that jointly models the input binarization-error and computational cost and introduce an efficient algorithm to select layers whose inputs are to be binarized.
- Practical guidelines based on insights obtained from applying the algorithm to a variety of models are discussed.



Our pipeline. We score each layer using the metric proposed and then cluster them into two classes which correspond to Weight binarized and Fully binarized layers



All layers were not made equal!
Qualitative examples showing why hybrid binarization is necessary



Plots of metric score vs layer number for various architectures

Approach

- Full-precision inputs **I** are approximated by binary matrix $\mathbf{I_B}$, whose representation is given below, along with XNOR-Net's approximation-error function:

$$\mathbf{I_B}^* = argmin(\| \mathbf{I} - \mathbf{I_B} \|^2)$$

$$\mathbf{E} = \frac{\| \mathbf{I} - \mathbf{I_B} \|^2}{n}$$

- To find *where* to apply input-binarization in the network, we need to measure the efficacy of binary approximation for inputs to any given layer.
- A good metric for this is the average error function calculated over a subset of training images used to calculate the optimal $\mathbf{I_B}$ itself, which is explicitly being minimized in the process.
- We also factor in computational intensity - where a layer with high NF (Number of FLOPs) value is a good candidate for binarization compared to a layer with a low NF value, due to increased computational speedup on binarization. We jointly optimize average error and computational intensity into a single metric as:

$$\mathbf{M} = \mathbf{E} + \gamma \cdot \frac{1}{\mathbf{NF}}$$

- Where γ is the tradeoff ratio, **NF** is the number of FLOPs in the layer, and **E** is the binarization error per neuron.
- We propose a *partitioning* algorithm to replace layers in a source model with WeightBinConv (Only weights binarized) or FullBinConv (Weights and Inputs binarized) layers, which gives us informed guesses on where to binarize by performing K-Means clustering on error terms and and finding a suitable number of clusters such that the ratio of layers in the highest-error cluster to the total number of convolutional layers is less than a fixed hyperparameter, *R* (the Hybridization Ratio). Layers falling in the highest mean cluster are converted to WeightBinConv layers, while other layers are converted to FullBinConv layers.

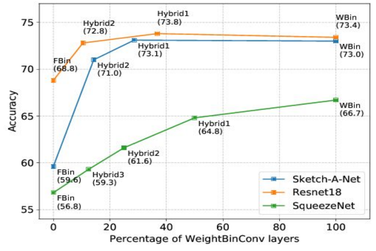
Algorithm 1 Partition Algorithm
Marks layers for binarization and creates a hybrid network.

```
Initialization
P = Total convolutional layers
R = Hybridization Ratio
ToConvert = List()

Mark binary layers
for N = 2 to P do
    Compute KMeans with N means
    K = Number of layers in highest-error cluster
    if K/P ≤ R then
        for Q in high-error clusters do
            ToConvert.add(Q)          ▷ Add layer Q
        Break

Create Hybrid Network
HybridNet = ()
HybridNet.Add(Conv)

for N = 2 to P do
    if N in ToConvert then
        HybridNet.Add(WeightBinConv)
    else
        HybridNet.Add(FullBinConv)
```



Comparisons of accuracies for hybrid models created by varying the hybridisation ratio

Results

- We measured accuracies for FBin and Hybrid variants of Sketch-A-Net and ResNet-18 models on TU-Berlin and Sketchy Datasets with weights of the last layer binarized as well as non-binarized. Our hybrid versions are able to similar accuracies to WBin counterparts due to last-layer binarization, which other binarization works do not do.
- The proposed algorithm for selective CNN binarization strikes a balance between performance, memory-savings and accuracy. Hybrid model accuracies are on par with corresponding FPrec networks on TU-Berlin and Sketchy datasets, while providing the benefits of network binarization in terms of speedups, compression and energy efficiency.
- We can successfully combine the advantages of our approach with other architectural compression strategies such as SqueezeNet, to obtain highly efficient models with negligible accuracy penalties.

Model	Method	Accuracy		Memory Savings	FLOPs
		Top-1	Top-5		
AlexNet	FPrec	57.1%	80.2%	1x	1135 (9.4x)
	WBin (BWN)	56.8%	79.4%	10.4x	780 (6.4x)
	FBin (XNOR)	43.3%	68.4%	10.4x	**121 (1x)**
	Hybrid-1	48.6%	72.1%	10.4x	174 (1.4x)
	Hybrid-2	**48.2%**	**71.9%**	**31.6x**	174 (1.4x)
Increase	Hybrid vs FBin	+4.9%	+3.5%	+21.2x	+53 (+0.4x)
ResNet-18	FPrec	69.3%	89.2%	1x	1814 (13.5x)
	WBin (BWN)	60.8%	83.0%	13.4x	1030 (7.7x)
	FBin (XNOR)	51.2%	73.2%	13.4x	**134 (1x)**
	Hybrid-1	54.9%	77.9%	13.4x	359 (2.7x)
	Hybrid-2	**54.8%**	**77.7%**	**31.2x**	359 (2.7x)
Increase	Hybrid vs FBin	+3.6%	+4.5%	+17.8x	+225 (+1.7x)

Results of our hybrid models on ImageNet

Model	Method	Accuracy		Memory Savings	FLOPs
		TU-Berlin	Sketchy		
Sketch-A-Net	FPrec	72.9%	85.9%	1x	608 (7.8x)
	WBin (BWN)	73%	85.6%	29.2x	406 (5.2x)
	FBin (XNOR)	59.6%	68.6%	19.7x	**78 (1x)**
	Hybrid	**73.1%**	**83.6%**	**29.2x**	**85 (1.1x)**
Increase	Hybrid vs FBin	+13.5%	+15.0%	+9.5%	+7 (+0.1x)
ResNet-18	FPrec	74.1%	88.7%	1x	1814 (13.5x)
	WBin (BWN)	73.4%	89.3%	31.2x	1030 (7.7x)
	FBin (XNOR)	68.8%	82.8%	31.2x	**134 (1x)**
	Hybrid	**73.8%**	**87.9%**	**31.2x**	359 (2.7x)
Increase	Hybrid vs FBin	+5.0%	+5.1%	-	+225 (+1.7x)

Results of our hybrid models on the TU-Berlin and Sketchy datasets

Link: http://bit.do/ameyap
Email: ameya.prabhu@research.iiit.ac.in