Hybrid Binary Networks: Optimizing for Accuracy, Efficiency and Memory

by

Prabhu Ameya Pandurang, Vishal Batchu, Rohit Gajawada, Sri Aurobindo Munagala, Anoop Namboodiri

in

IEEE Winter Conference on Applications of Computer Vision
(*WACV-2018*)

Lake Tahoe, NV, USA

Report No: IIIT/TR/2018/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2018

Hybrid Binary Networks: Optimizing for Accuracy, Efficiency and Memory

Ameya Prabhu Vishal Batchu Rohit Gajawada Sri Aurobindo Munagala Anoop Namboodiri
Center for Visual Information Technology, Kohli Center on Intelligent Systems
IIIT-Hyderabad, India
{ameya.prabhu@research., vishal.batchu@students., rohit.gajawada@students.,
s.munagala@research., anoop@}iiit.ac.in

Abstract

Binarization is an extreme network compression approach that provides large computational speedups along with energy and memory savings, albeit at significant accuracy costs. We investigate the question of where to binarize inputs 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.

Experiments on Imagenet dataset using AlexNet and ResNet-18 models show 3-4% improvements in accuracy over fully binarized networks with minimal impact on compression and computational speed. The improvements are even more substantial on sketch datasets like TU-Berlin, where we match state-of-the-art accuracy as well, getting over 8% increase in accuracies. We further show that our approach can be applied in tandem with other forms of compression that deal with individual layers or overall model compression (e.g., SqueezeNets). Unlike previous quantization approaches, we are able to binarize the weights in the last layers of a network, which often have a large number of parameters, resulting in significant improvement in accuracy over fully binarized models.

1. Introduction

Convolutional Neural Networks (CNNs) have found applications in many vision-related domains ranging from generic image-understanding for self-driving cars [3] and automatic image captioning [32, 20] to recognition of specific image parts for scene-text recognition [24, 26] and face-based identification [29].

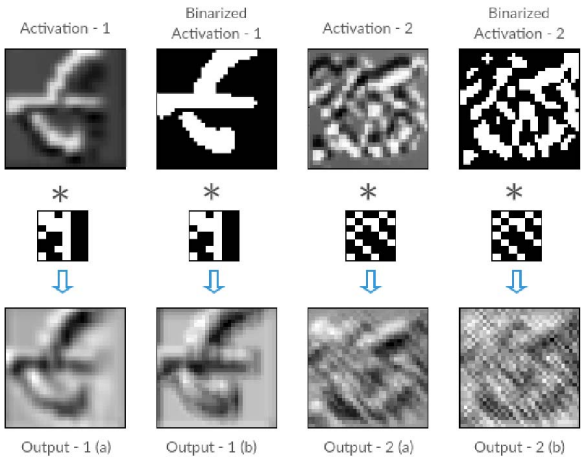

Figure 1: Convolution of binary and non-binary activations of two different layers. Note that the error introduced due to binarization is minimal in the first pair compared to the second. Hence, efficiently deciding *which* layers to binarize could contribute significantly to the overall accuracy of the network and not damage the speed-ups.

After the introduction of AlexNet [21], several architectural improvements were proposed to push image recognition accuracy, such as VGG-Net [28], but these models were massive both in terms of memory usage and computational costs. AlexNet has around 60 million parameters in the network, while VGG has around 138 million, requiring 1.5 billion FLOPs and 19.6 billion FLOPs respectively for inference. The computational requirements make these architectures inappropriate for smaller portable systems such as mobiles and other embedded systems. These networks also use large amounts of energy, creating a bottleneck for performance improvements. Full-precision multiply-accumulate (MAC) operations in convolutional layers consume 30x more power than integer MAC operations (see Table 1).

Since these applications would be deployed on resource-constrained systems, CNN compression is an important

978-1-5386-4886-5/18/$31.00 ©2018 IEEE
DOI 10.1109/WACV.2018.00095

Operation	MUL	Power	ADD	Power
32-bit Float	3.7pJ	18.5x	0.9pJ	30x
16-bit Float	1.1pJ	5.5x	0.4pJ	13.3x
8-bit Integer	0.2pJ	1x	0.03pJ	1x

Table 1: As shown by Horowitz *et al*. [14], power consumption for various operations at 45nm 0.9V. Observe that 8-bit integers require significantly less energy than their equivalent 32-bit floating point operations.

emerging area for research on vision applications [18, 36, 11, 23, 25, 31, 13, 19]. One of the methods of compression: Quantization, can help networks consume far less power, memory, and incur lower computational costs.

Quantization has proven to be a powerful compression strategy. Our paper is based on the most extreme form of quantization - Binarization. There are many benefits to binarizing a network. Primarily, having binary weights/activations enables us to use xnor and popcount operations to calculate weighted sums of the inputs to a layer as compared to full-precision multiply-accumulate operations (MACs). This results in significant computational speedup compared to other compression techniques. Secondly, as each binary weight requires only a single bit to represent, one can achieve drastic reductions in run-time memory requirements. Previous research [27, 18] shows that it is possible to perform weight and input binarization on large networks with up to 58x speedups and 10.4x compression ratios, albeit with significant drops in accuracy.

In this paper, we explore the problem of hybrid binarization of a network. We propose a technique devised from our investigation into the question as to *where and which quantities of a network should one binarize*, with respect to inputs to a layer - to the best of our knowledge, this is the first work that explores this question. We observe in Figure 1 that in a trained fully binarized model, binarization in certain layers induces minimal error, whereas in others, the error obtained is significant. Our proposed partition algorithm, when run on trained fully binarized models can design effective architectures. When these hybrid models are trained from scratch, they achieve a balance between compression, speedup, energy-efficiency, and accuracy, compared to fully binarized models. We conduct extensive experiments applying our method to different model architectures on popular large-scale classification datasets over different domains. The resulting models achieve significant speedups and compression with significant accuracy improvements over a fully binarized network.

Our main contribution includes:

1. A metric to jointly optimize binarization-errors of layers and the associated computational costs;
2. A partitioning algorithm to find suitable layers for input binarization, based on the above metric, which generates hybrid model architectures which if trained from scratch, achieve a good balance between compression, speedup, energy-efficiency, and accuracy;
3. Insights into what the algorithm predicts, which can provide an intuitive framework for understanding why binarizing certain areas of networks give good benefits;
4. Hybrid model architectures for AlexNet, ResNet-18, Sketch-A-Net and SqueezeNet with over 5-8% accuracy improvements on various datasets; and
5. A demonstration that our technique that achieves significant compression in tandem with other compression methods.

Reproducibility: Our implementation can be found on GitHub [1].

2. Related Work

CNNs are often over-parametrized with high amounts of redundancy, increasing memory costs and making computation unnecessarily expensive. Several methods were proposed to compress networks and eliminate redundancy, which we summarize below.

Space-efficient architectures: Designing compact architectures for deep networks helps save memory and computational costs. Architectures such as ResNet [13], DenseNet [17] significantly reduced model size compared to VGG-Net by proposing a bottleneck structure to reduce the number of parameters while improving speed and accuracy. SqueezeNet [19] was another model architecture that achieved AlexNet-level accuracy on ImageNet with 50x fewer parameters by replacing 3x3 filters with 1x1 filters and late downsampling in the network. MobileNets [16] and ShuffleNets [35] used depthwise separable convolutions to create small models, with low accuracy drop on ImageNet.

Pruning and Quantization: Optimal Brain Damage [8] and Optimal Brain Surgeon [12] used the Hessian of the loss function to prune a network by reducing the number of connections. Deep Compression [11] reduced the number of parameters by an order of magnitude in several state-of-the-art neural networks through pruning. It further reduced non-runtime memory by employing trained quantization and Huffman coding. Network Slimming [23] took advantage of channel-level sparsity in networks, by identifying and pruning out non-contributing channels during training. HashedNets [5] performed binning of network weights using hash functions. INQ [2] used low-precision 16 bit-quantized weights and achieved an 8x reduction in memory consumption, using 4 bits to represent 16 distinct quantized values and 1 bit to represent zeros specifically.

[1] https://github.com/erilyth/HybridBinaryNetworks-WACV18

Binarization: BinaryConnect [6] obtained huge compression in CNNs where all weights had only two allowed states (+1, -1) using Expectation Back Propagation (EBP). Approaches like [18, 22, 37] train deep neural networks using low precision multiplications, bringing down memory required drastically, showing that these models could be fit on memory constrained devices. DoReFa-net [36] applied low bit width gradients during back-propagation. XNOR-Net [27] multiplied binary weights and activations with scaling constants based on layer norms. QNNs [18] extended BNNs[7], the first method using binary weights and inputs to successfully achieve accuracy comparable to their corresponding 32-bit versions on constrained datasets using higher bit quantizations. HWGQ-Net [4] introduces a better suited activation function for binary networks. HTCBN [30] introduce helpful techniques such as replacing ReLU layers with PReLU layers and a scale layer to recover accuracy loss on binarizing the last layer, to effectively train a binary neural network. Hou *et al*. [15] use Hessian approximations to minimize loss w.r.t binary weights during training. Anderson *et al*. [1] offers a theoretical analysis of the workings of binary networks, in terms of high-dimensional geometry.

Unlike previous works in this area, we look at binarizing specific parts of a network, instead of simply binarizing the inputs to all the layers end-to-end. We see in later sections, binarizing the right areas in the network contributes significantly to the overall accuracy of the network and does not damage its speed-ups.

3. Hybrid Binarization

We define certain conventions to be used throughout the paper. We define a WBin CNN to be a CNN having the weights of convolutional layers binarized (referred to as WeightBinConv layers), FBin CNN to be a CNN having both inputs and weights of convolutional layers binarized (referred to as FullBinConv layers) and FPrec CNN to be the original full-precision network having both weights and inputs of convolutional layers in full-precision (referred to as Conv layers). We compare the FBin and WBin networks with FPrec networks at specific layers.

Table 3 and Table 4 in the Experiments section show test accuracies for WBin, FBin and FPrec networks of different models. Observe that there is very little loss in accuracy from FPrec to WBin networks with significant memory compression and fewer FLOPs. However, as we go from WBin to FBin networks, there is a significant drop in accuracy along with the trade-off of significantly lower FLOPs in FBin over WBin networks. Hence, we focus on improving the accuracies of FBin networks along with preserving the lower FLOPs as far as possible by investigating which activations to binarize.

3.1. Error Metric: Optimizing Speed & Accuracy

Full-precision inputs $\mathbf{I} \in \mathbb{R}^n$, are approximated by binary matrix $\mathbf{I_B} \in \{-1, +1\}^n$. The optimal binary representation $\mathbf{I_B}$ is calculated by

$$\mathbf{I_B}^* = argmin(\| \mathbf{I} - \mathbf{I_B} \|^2) \tag{1}$$

XNOR-Net[27] minimized the error function:

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

In order to do that, they maximized $\mathbf{I}^\top \mathbf{I_B}$ and proposed the binary activation $\mathbf{I_B}$ to be

$$\mathbf{I_B}^* = \underset{\mathbf{I_B}}{\operatorname{argmax}}(\mathbf{I}^\top \mathbf{I_B}), \mathbf{I_B} \in \{-1, +1\}^n, \mathbf{I} \in \mathbb{R}^n \tag{3}$$

, obtaining the optimal $\mathbf{I_B}^*$ can be shown to be $sgn(\mathbf{I})$.

We need to investigate *where* to replace FullBinConv with WeightBinConv layers. In order to optimize for accuracy, we need to measure the efficacy of the binary approximation for inputs to any given layer. A good metric of this is the average error function calculated over a subset of training images $\mathbf{E}$ (defined in Eq. 2) used to calculate the optimal I_B itself, which is explicitly being minimized in the process. Hence, we use that error function to capture the binarization error.

Similarly to optimize speed, we need to convert layers with low number of FLOPs to WeightBinConv and layers having high number of FLOPs should be kept in FullBinConv. Since we need to jointly optimize both, we propose a metric that tries to achieve a good tradeoff between the two quantities. A simple but effective metric is the linear combination

$$\mathbf{M} = \mathbf{E} + \gamma \cdot \frac{1}{\mathbf{NF}} \tag{4}$$

where γ is the tradeoff ratio, $\mathbf{NF}$ is the number of flops in the layer and $\mathbf{E}$ is the binarization error per neuron. The trade-off ratio γ is a hyperparameter which ensures that both the terms are of comparable magnitude. Figure 2, captures the layer-wise variation of the error metric across multiple models.

3.2. Partitioning Algorithm

We aim to partition the layers of a network into two parts, one set of layers to keep FullBinConv and the other set which are replaced with WeightBinConv layers. A naive but intuitive partitioning algorithm would be to sort the list of metric errors M and replace FullBinConv layers which have highest error values M_i one-by-one with WeightBinConv layers, train new hybrid models and stop when the accuracies in the retrained models stop improving i.e when the maxima in accuracy v/s flops tradeoff is reached. However, we need a partitioning algorithm which gives informed

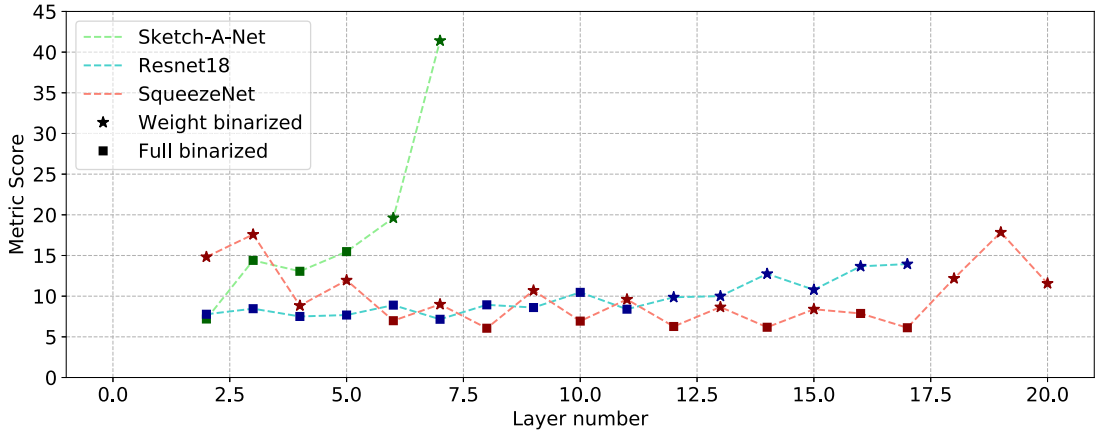

Figure 2: Binarization-error metric across layers for Sketch-A-Net, ResNet-18, and SqueezeNet. Stars indicate that the layer was replaced with a WeightBinConv layer, while squares indicate the FullBinConv layer was retained in the FBin model. We see that the algorithm selects the last layers in the case of Sketch-A-Net and ResNet, while in the case of SqueezeNet, it selects the first four, last three and some alternate intermediate layers to be replaced by WeightBinConv layers, retaining the rest as FullBinConv layers.

guesses on where are the effective places to partition the set. This would avoid the long retraining times and large resources required to try every possible option for a hybrid model. We propose a layer selection algorithm that gives informed partitions from a trained FBin model, helping us to determine which layers are to be converted to WeightBinConv and which layers are to be converted to FullBinConv without having to train all possible hybrid models from scratch.

Our algorithm starts by taking a trained FBin model. We pass in a subset of the training images and calculate the average error metric for all layers over them. Then we perform K-Means Clustering on the metric values with each point being the metric error of layers as shown in Figure 2. We perform the K-Means Clustering for different values of the number of clusters. We find a suitable number of clusters such that the ratio of layers in the highest-error cluster (K) to the total number of convolutional layers (P) is less than a hyperparameter, which we define as the Hybridization Ratio R. Layers with terms falling in the highest mean cluster are converted to WeightBinConv, while the ones in all other clusters are left as FullBinConv. A flow of the algorithm is illustrated in Figure 3 and is explained step-by-step in Algorithm 1. We show metric scores of various layers for different networks in Figure 2 and indicate which layers are replaced with WeightBinConv/FullBinConv layers. This algorithm guides in forming the architecture of the hybrid model, which is then trained from scratch obtaining the accuracies given in the tables presented in the Experiment section. Note that this algorithm does not change the configuration of the model; it only converts certain layers to their binarized versions.

Algorithm 1 Partition Algorithm

Marks layers for binarization and creates a hybrid network.

```
Inputs ⇒Layer-wise Binarization Errors

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)

Output ⇒ HybridNet
```

To give an intuition of what the Hybridization ratio R means, a low R would indicate we need the number of

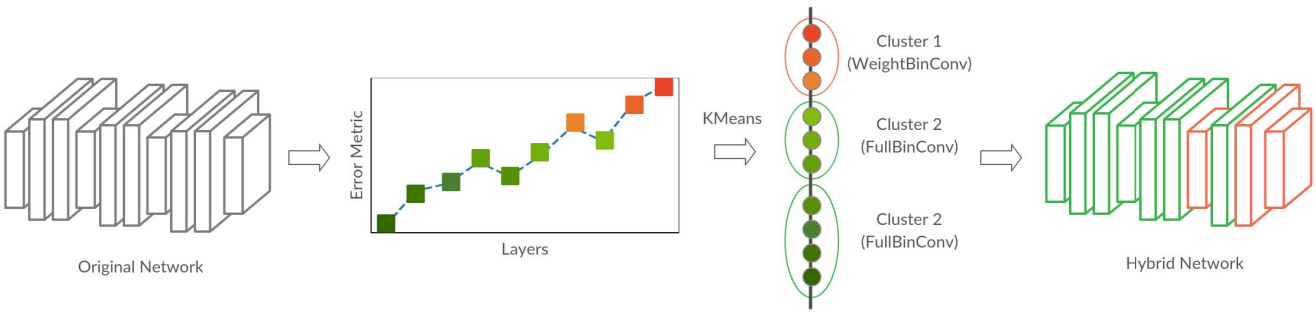

Figure 3: The Procedure: Error metrics from binarization of inputs to the network layers are partitioned into clusters using K-means. The highest error cluster indicates the inputs that are not binarized to generate the hybrid version.

WeightBinConv layers to be low, ensuring a high asymmetry between errors in WeightBinConv and FullBinConv layers, prioritizing saving computational cost. Conversely, a higher R would prioritize accuracy over computational cost. R was set to be 0.4 for AlexNet and ResNet-18, and 0.6 for Squeezenet. Variation with different values of R is further discussed in the experiments section.

3.3. Impact on Speed and Energy Use

Computational Speedups: Convolutional operations are computationally expensive. For each convolution operation between an image $\mathbf{I} \in \mathbb{R}^{c_{in} \times h_I \times w_I}$ and weight $\mathbf{W} \in \mathbb{R}^{c_{out} \times h \times w}$, the number of MAC operations required N are $\approx C_{in}C_{out}N_W N_I$ where $N_W = wh$ and $N_I = w_I h_I$. According to benchmarks done in XNOR-Net, the current speedup obtained in these operations is 58x after including the overhead induced by computing α. Accordingly, in later sections, we take one FLOP through a layer as equivalent to 58 binary operations when weights and inputs are binarized.
Exploiting filter repetitions: The number of unique convolutional binary filters is bounded by the size of the filter [18]. As most of our intermediate convolutional layers have 3×3 filters which only have 2^9 unique filters, we find that the percentage of unique filters decreases as we go deeper into the network. We can exploit this fact to simply prune filters and use that in calculating speedups for binary networks. More details regarding how the speedup was computed is included in the supplementary material.

4. Experiments and Results

We report and compare accuracies, speedups and compression between the FPrec model, different kinds of binarization models (WBin and FBin), and their generated hybrid versions of the same. We also present a detailed comparison of our method with several different compression techniques applied on AlexNet [21], ResNet-18 [13], Sketch-A-Net [10] and SqueezeNet [19].
We empirically demonstrate the effectiveness of hybrid binarization on several benchmark image and sketch datasets. We show that our approach is robust and can generalize to different types of CNN architectures across domains.

4.1. Datasets and Models

Binary Networks have achieved accuracies comparable to full-precision networks on limited domain/simplified datasets like CIFAR-10, MNIST, SVHN, but show drastic accuracy losses on larger-scale datasets. To compare with state-of-the-art vision, we evaluate our method on ImageNet[9]. To show the robustness of our approach, we test it on sketch datasets, where models fine-tuned with ImageNet are demonstrably not suitable as shown in[34]. Binary networks might be better suited for sketch data due to its binary nature and sparsity of information in the data.

ImageNet: The benchmark dataset for evaluating image recognition tasks, with over a million training images and 50,000 validation images. We report the single-center-crop validation errors of the final models.

TU-Berlin: The TU-Berlin [10] sketch dataset is the most popular large-scale free-hand sketch dataset containing sketches of 250 categories, with a human sketch-recognition accuracy of 73.1% on average.

Sketchy: It is a recent large-scale free-hand sketch dataset containing 75,471 hand-drawn sketches from across 125 categories. This dataset was primarily used to cross-validate results obtained on the TU-Berlin dataset and ensure that our approach is robust to the variation in collection of data.

We use the standard splits with commonly used hyper-parameters to train our models. Each FullBinConv block was structured as in XNOR-Net (Batchnorm-Activ-Conv-ReLU). Each WeightBinConv and Conv block has the standard convolutional block structure (Conv-Batchnorm-ReLU). Weights of all layers except the first were binarized throughout our experiments unless specified otherwise. Note that FLOPs are stated in millions in all diagrams and sections. All networks are trained from scratch independently. The architecture of the hybrid network once designed does not change during training. Additional details

Technique	Acc-Top1	Acc-Top5	W/I	Mem	FLOPs
AlexNet					
BNN	39.5%	63.6%	1/1	32x	121 (1x)
XNOR	43.3%	68.4%	1/1	10.4x	**121 (1x)**
Hybrid-1	48.6%	72.1%	1/1	10.4x	174 (1.4x)
Hybrid-2	**48.2%**	**71.9%**	1/1	**31.6x**	174 (1.4x)
HTCBN	46.6%	71.1%	1/2	31.6x	780 (6.4x)
DoReFa-Net	47.7%	-	1/2	10.4x	780 (6.4x)
Res-Net 18					
BNN	42.1%	67.1%	1/1	32x	134 (1x)
XNOR	51.2%	73.2%	1/1	13.4x	**134 (1x)**
Hybrid-1	54.9%	77.9%	1/1	13.4x	359 (2.7x)
Hybrid-2	**54.8%**	**77.7%**	1/1	**31.2x**	359 (2.7x)
HTCBN	53.6%	-	1/2	31.2x	1030 (7.7x)

Table 2: A detailed comparison of accuracy, memory use, FLOPs with popular benchmark compression techniques on ImageNet. Our hybrid models outperform other 1-bit activation models and perform on par with 2-bit models while having a significantly higher speedup. Hybrid-2 models have the last layer binarized.

about the datasets, model selection and layer-wise description of each of the hybrid models along with experimental details can be found in the supplementary material.

4.2. Results

We compare FBin, WBin, Hybrid and FPrec recognition accuracies across models on ImageNet, TU-Berlin and Sketchy datasets. Note that higher accuracies are an improvement, hence stated in green in the table, while higher FLOPs mean more computational expense, hence are stated in red. W/I indicates the number of bits used for weights and inputs to the layer respectively. Note that in the table, the compression obtained is only due to the weight binarization, while the decrease in effective FLOPs are due to activation binarization.

On the ImageNet dataset in Table 3, hybrid versions of AlexNet and ResNet-18 models outperform their FBin counterparts in top-1 accuracy by 4.1% and 3.6% respectively, and around 20x compression for both. We also compare with the results of other compression techniques in Table 2. On the TU-Berlin and Sketchy datasets in Table 4, we find that Sketch-A-Net and ResNet-18 have significantly higher accuracies in the hybrid models compared to their FBin counterparts, a 13.5% gain for Sketch-A-Net and 5.0% for ResNet-18.

These hybrid models also achieve over 29x compression over FPrec models and with a reasonable increase in the number of FLOPs - a mere 7M increase in Sketch-A-Net and a decent 225M increase in ResNet-18. We also compare them with state-of-the-art sketch classification models in Table 5. Our hybrid Sketch-A-Net and ResNet-18 models achieve similar accuracies to state-of-the-art, while also highly compressing the models upto 233x compared to the

Model	Method	Accuracy		Mem	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)

Table 3: Our hybrid models compared to FBin, WBin and NoBin models on Imagenet in terms of accuracy, memory and computations expense.

Model	Method	Accuracy		Mem	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%	+9.5x	+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)

Table 4: Our hybrid models compared to FBin, WBin and full prec models on TU-Berlin and Sketchy datasets in terms of accuracy, memory and speed tradeoff.

Model	Acc	Mem	FLOPs
AlexNet-SVM	67.1%	1x	1135 (13.4x)
AlexNet-Sketch	68.6%	1x	1135 (13.4x)
Sketch-A-Net SC	72.2%	8x	608 (7.2x)
Sketch-A-Net-Hybrid	**73.1%**	**233x**	**85 (1x)**
ResNet18-Hybrid	**73.8%**	-	**359**
Humans	73.1%	-	-
Sketch-A-Net-2 [2][33]	**77.0%**	8x	608 (7.2x)

Table 5: A comparison between state-of-the-art single model accuracies of recognition systems on the TU-Berlin dataset.

AlexNet FPrec model. Thus, we find that our hybrid binarization technique finds a balance between sacrificing accuracy and gaining speedups and compression for various models on various datasets.

4.3. Algorithmic Insights

We gained some insights into where to binarize from our investigation. We provide them as a set of practical guidelines to enable rapid prototyping of hybrid models, which gives meaningful insights into which layers were par-

[2]It is the sketch-a-net SC model trained with additional imagenet data, additional data augmentation strategies and considering an ensemble, hence would not be a direct comparison

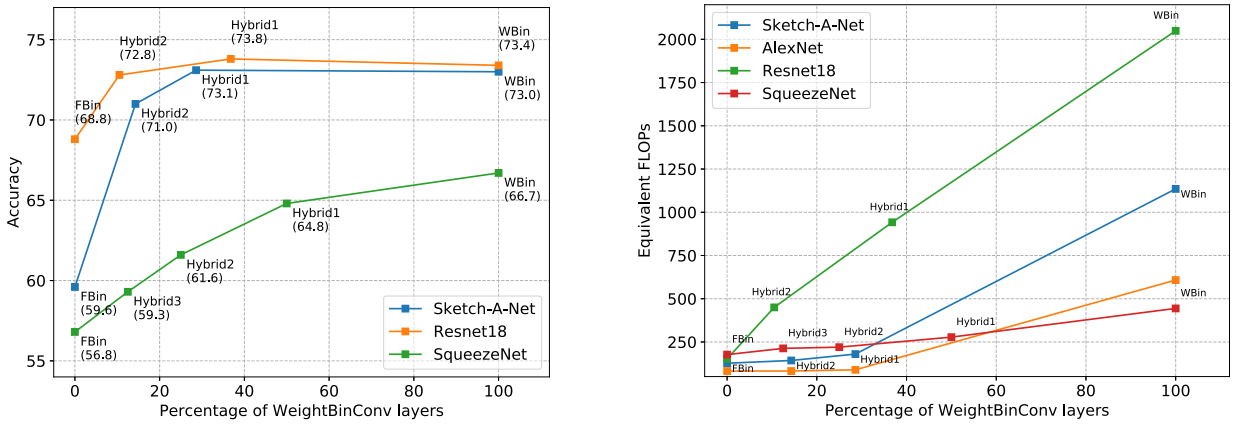

Figure 4: Trade-off between WeightBinConv layers and accuracy on the TU-Berlin dataset is shown in the left figure, while the trade-off between weight binarized layers and speedup is shown in the right figure. Early on, we observe that a small increase in the percentage of WeightBinConv layers leads to a large increase in accuracy and a marginal decrease in speed. We achieve accuracies comparable to the WBin model with much fewer WeightBinConv layers.

titioned.

Convert layers towards the end to WeightBinConv: It is observed that later layers typically have high error rates, more filter repetitions, and lower computational cost. Hence, the algorithm tends to start converting models to Hybrid from the last layers.

Convert the smaller of the layer placed parallely to WeightBinConv: It is a good idea to convert the smaller of the parallely placed layers in the architecture like Residual layers in the ResNet architecture to WeightBinConv, since converting them to WeightBinConv would not damage the computational speedup obtained by the parallel FullBinConv layers.

Pick a low Hybridization Ratio: Try to pick low values of the Hybridization Ratio R, ensuring a low proportion of number of layers the highest-error cluster.

Relax the Hybridization Ratio for compact models: Having a higher Hybridization Ratio for compact models which inherently have fewer flops leaves more layer inputs un-binarized and retains accuracy.

4.4. Why are layer-wise errors independent?

Can binarization noise introduced in a layer propagate further into the network and influence other layers? Hubara *et al*. [18] provide some insights for the same. Let $\mathbf{W}$ be the weight and $\mathbf{I}$ be the input to the convolutional layer. The output of the convolution between the binary weights and inputs can be represented by

$$\mathbf{O_B} = \alpha \cdot (sgn(\mathbf{W^T}) \odot sgn(\mathbf{I})) \tag{5}$$

The desired output $\mathbf{O}$ is modelled by $\mathbf{O_B}$ along with the binarization noise $\mathbf{N}$ introduced due to the function $sgn(.)$.

$$\mathbf{O} = \mathbf{W} * \mathbf{I} = \sum_i \mathbf{O_{B}}_i + \mathbf{N_i} \tag{6}$$

When the layer is wide, we expect the deterministic term $\mathbf{O_B}$ to dominate, because the noise term $\mathbf{N}$ is a summation over many independent binarizations from all the neurons in the layer. Thus, we argue that the binarization noise $\mathbf{N}$ should have minimal propagation and do little to influence the further inputs. Hence, it is a reasonable approximation to consider the error across each layer independently of the other layers.

4.5. Variation with the Hybridization Ratio (R)

To observe the trade-off between accuracy and speedup on different degrees of binarization, we chose different values of the Hybridization Ratio (R) to create multiple hybrid versions of the AlexNet, ResNet-18 and SqueezeNet models. Picking a larger R would result in a higher number of WeightBinConv layers. We compare these hybrid networks to their corresponding FBin and WBin versions.

In Figure 4, we show model accuracies of AlexNet, ResNet-18 and SqueezeNet on the ImageNet dataset plotted against the number of WeightBinConv layers, starting from only FBin versions on the left, to only WBin versions on the right. We observe that in the case of AlexNet and ResNet-18, which are large models, we recover WBin accuracies quickly, at around the 35% mark (Roughly a third of the network containing WeightBinConv layers), with low computational trade-off. We also observe that on sketch data, hybrid models tend to perform significantly better and perform on par with their WBin counterparts.

We also notice that the smaller a model, the more trade-off must be made to achieve WBin accuracy, i.e a larger Hybridization Ratio must be used. AlexNet, the largest model crosses WBin accuracy at around 32%, while ResNet-18, being smaller, saturates at around 40%. SqueezeNet, a much more compact model, reaches its WBin accuracy at

Model	BinType	Last Bin?	Acc	Mem
Sketch-A-Net	FBin (XNOR)	No	59.6%	19.7x
		Yes	48.3%	**29.2x**
Sketch-A-Net	Hybrid	No	73.1%	19.7x
		Yes	72.0%	**29.2x**
Resnet-18	FBin (XNOR)	No	69.9%	13.4x
		Yes	68.8%	**31.2x**
Resnet-18	Hybrid	No	73.9%	13.4x
		Yes	73.8%	**31.2x**

Table 6: Effects of last layer weight-binarization on TU-Berlin dataset, for Sketch-A-Net and ResNet-1. Observe that our hybrid models do not face drastic accuracy drop when the last layer is weight-binarized.

60%.

4.6. Optimizing Memory

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 and the results are presented in Table 6. For AlexNet-styled architectures (Sketch-A-Net), we observe a drastic drop in accuracies (From 59.1% to 48.3%) on binarizing the last layer, similar to observations made in previous binarization works [36, 30].

Many efforts were made to quantize the last layer and avoid this drop. DoReFaNet and XNOR-Net did not binarize the last layer choosing to incur a degradation in model compression instead while [30] proposed an additional scale layer to mitigate this effect. However, our hybrid versions are able to achieve similar accuracies (a 1% drop for hybrid Sketch-A-Net and no drop for ResNet-18 or AlexNet) since the last layer is weight binarized instead. Hence, our method preserves the overall speedup even though we only weight-binarize the last layer, owing to the comparatively smaller number of computations that occur in this layer.

Note that the first layer is always a full-precision Conv layer. The reasons behind this are the insights obtained from [1]. They state that the first layer of the network functions are fundamentally different than the computations being done in the rest of the network because the high variance principal components are not randomly oriented relative to the binarization. Also, since it contains fewer parameters and low computational cost, it does not affect our experiments.

4.7. Compressing Compact Models

Whether compact models can be compressed further, or *need* all of the representational power afforded through dense floating-point values is an open question asked originally by [19].

We show that our hybrid-binarization technique can work in tandem with other compression techniques, which do not involve quantization of weights/activations and that hybrid binarization is possible even on compact models. We apply hybrid binarization to SqueezeNet[19] a recent model that employed various architectural design strategies to achieve compactness. SqueezeNet achieves an 8x compression on the compact architecture of Sketch-A-Net. On applying hybrid binarization we achieve a further 32x compression, an overall 256x compression with merely 6% decrease in accuracy. This is due to the high rate of compression inherent and further compression is difficult due to the small number of parameters. After showing that efficacy of hybrid binarization in the previous section, we show that hybrid binarization can work in combination with other compression techniques here.

Model	Method	Accuracy		Mem	FLOPs
		TU-Berlin	Sketchy		
Sketch-A-Net	FPrec	72.9%	85.9%	1x	1135 (12.3x)
Squeezenet	FPrec	71.2%	86.5%	8x	610 (6.6x)
Squeezenet	WBin	66.7%	81.1%	23.7x	412 (4.5x)
Squeezenet	FBin	56.8%	66.0%	23.7x	**92 (1x)**
Squeezenet	Hybrid	**64.8%**	**79.6%**	**23.7x**	164 (1.8x)
Improvement	Hybrid vs FBin	+8.0%	+13.6%	-	+72 (+0.8x)

Table 7: Our performance on SqueezeNet, an explicitly compressed model architecture. Although SqueezeNet is an inherently compressed model, our method still achieves further compression on it.

Results for SqueezeNet are shown in Table 7 for the TU-Berlin and Sketchy datasets, and we see that accuracy is only slightly lower compared to the hybridized versions of ResNet-18 and Sketch-A-Net on the same. Hybrid SqueezeNet achieves a total compression of 256x. Similarly, this technique can be combined with many techniques such as HWGQ-Net [4] which proposes an alternative layer to ReLU and repeated binarization as illustrated in [30] among others. Since our primary goal is to investigate the viability of hybrid binarization, these investigations- albeit interesting, are out of the scope of our current work.

5. Conclusion

We proposed a novel algorithm for selective binarization of CNNs, which strikes a balance between performance, memory-savings and accuracy. The accuracies of our hybrid models were on par with their corresponding full-precision networks on TU-Berlin and Sketchy datasets, while providing the benefits of network binarization in terms of speedups, compression and energy efficiency. We successfully weight-binarized the last layers without significant accuracy drops, a problem faced by previous works in this area. We also showed that we can successfully combine the advantages of our approach with other architectural compression strategies, to obtain highly efficient models with negligible accuracy penalties.

References

[1] A. G. Anderson and C. P. Berg. The high-dimensional geometry of binary neural networks. *arXiv preprint arXiv:1705.07199*, 2017.

[2] Z. Aojun, Y. Anbang, G. Yiwen, X. Lin, and C. Yurong. Incremental network quantization: Towards lossless cnns with low-precision weights. *ICLR*, 2017.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] Z. Cai, X. He, J. Sun, and N. Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. *CVPR*, 2017.

[5] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. *ICML*, 2015.

[6] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[8] Y. L. Cunn, J. S. Denker, and S. A. Solla. Nips. chapter Optimal Brain Damage. 1990.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[10] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ICLR*, 2016.

[12] B. Hassibi, D. G. Stork, G. Wolff, and T. Watanabe. Optimal brain surgeon: Extensions and performance comparisons. NIPS, 1993.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] M. Horowitz. Computing's energy problem (and what we can do about it). In *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014.

[15] L. Hou, Q. Yao, and J. T. Kwok. Loss-aware binarization of deep networks. *ICLR*, 2017.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *ICLR*, 2017.

[20] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[22] F. Li, B. Zhang, and B. Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[23] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. *ICCV*, 2017.

[24] A. Mishra, K. Alahari, and C. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR*, pages 2687–2694, 2012.

[25] M. Moczulski, M. Denil, J. Appleyard, and N. de Freitas. Acdc: A structured efficient linear layer. *ICLR*, 2016.

[26] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR*, pages 3538–3545, 2012.

[27] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[29] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[30] W. Tang, G. Hua, and L. Wang. How to train a compact binary neural network with high accuracy? In *AAAI*, pages 2625–2631, 2017.

[31] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *ICCV*, pages 1476–1483, 2015.

[32] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *CVPR*, pages 4651–4659, 2016.

[33] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net: A deep neural network that beats humans. *International Journal of Computer Vision*, 122(3):411–425, 2017.

[34] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales. Sketch-a-net that beats humans. *BMVC*, 2015.

[35] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

[36] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *ICLR*, 2016.

[37] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. In *ICLR*, 2017.