A. Experimental Setup

Configuration of hyperparameters. cResNet-20, ResNet-18 and MobileNet were trained with batch sizes of 128 at an initial learning rate of 0.1. For TinyImageNet batch sizes were reduced to 64 and initial learning rate increased to 0.2. We used stochastic gradient descent optimizer with a momentum value of 0.9 and weight decay of 10^{-4}, whereas for MobileNet weight decay of 10^{-5} was used. Cross Entropy Loss was used with label smoothing of 0.01. We use step learning rate strategy to decay learning rate by 0.1 after 50% and 75% of the total epochs. For CIFAR100, models were trained for 160 epochs with RandomCrop and RandomHorizontalFlip augmentations whereas for TinyImageNet the number of epochs were reduced to 120, and RandomAffine and RandomHorizontalFlip were used as augmentations. Moreover, for ImageNet, models were trained for 60 epochs with RandomResizedCrop and RandomHorizontalFlip as augmentations.

Backbone of SIAMFC was replaced by the first 3 blocks of ResNet-18. Model was trained with a batch size of 8 at an initial learning rate of 10^{-3} using SGD optimizer with momentum 0.9 and weight decay 5×10^{-4}. We use Expontial learning rate schedular with gamma value of 0.59 and final learning rate of 10^{-5}. All experiments were trained for 50 epochs with early stopping enabled. Please refer to open-source implementation of https://github.com/huanglianghua/siamfc-pytorch for further details.

Hardware and training time. Single Nvidia V100 32GB card with 512 GB RAM and a 64-core processor was used for running all the experiments. The training time for pretraining ResNet-18 on ImageNet was 30 hours, and each subsequent run on the specified budget also took 30 hours.

B. Iterative Selection

In iterative search strategy we identify the right layers of any given network to be converted to binary or full precision, one-by-one. This approach is described in Algorithm 1. For the k out of n layers to be binarized, the j^{th} step of binarization, where $j \in [1, k]$, can be stated as finding the optimal layer $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}_{(j)}$ to be binarized. It can be mathematically stated as follows.

Algorithm 1: Iterative Selection

Given : Empty set $\{\}$;Current layer chosen $\boldsymbol{\theta}_j$; Optimal layer chosen $\boldsymbol{\theta}_j^*$; Current objective function $\mathcal{L}$; Optimal objective function $\mathcal{L}^*$;
Input : Network weight set $\boldsymbol{\Theta}$
Output : Binarized weight set $\boldsymbol{\Phi}$

```
Φ ← {}
for j = 1 ... k do
    L* ← ∞
    for θ_j ∈ Θ do
        L ← ITERATIVE SELECTION(θ_j + Φ, Θ − θ_j, )
        if L < L* then
            L* ← L
            θ*_j ← θ_j
        end
    end
    Φ ← PUSH(θ*_j)
    Θ → POP(θ*_j)
end
```

$$\boldsymbol{\theta}^*_{(j)} = \underset{\boldsymbol{\theta} \subset \boldsymbol{\Theta}_{(j)}, \mathbf{W}}{\operatorname{argmin}} \; \mathcal{L}(\mathcal{F}(\boldsymbol{\theta} + \boldsymbol{\Phi}_{(j)}, \boldsymbol{\Theta}_{(j)} - \boldsymbol{\theta}, \mathbf{x}), \mathbf{y})$$
$$\text{s.t.} \quad \mathcal{B}(\boldsymbol{\theta} + \boldsymbol{\Phi}_{(j)}, \boldsymbol{\Theta}_{(j)} - \boldsymbol{\theta}) \leq \mathcal{B}_0,$$

where $\boldsymbol{\Theta}_{(j)} = \boldsymbol{\Theta} - \boldsymbol{\Phi}_{(j)}$. Here, $\boldsymbol{\Phi}_{(j)}$ denotes the layers that have already been binarized in the previous $j-1$ steps and is defined as $\boldsymbol{\Phi}_{(j)} = \{\boldsymbol{\theta}^*_{(1)}, \boldsymbol{\theta}^*_{(2)}, \ldots, \boldsymbol{\theta}^*_{(j-1)}\}$, where $\boldsymbol{\theta}^*_j$ denotes the optimal layer chosen at the j^{th} step of binarization to obtain B2NN. For calculating $\boldsymbol{\theta}^*_j$, we perform brute search over all elements of $\boldsymbol{\Theta}_{(j)}$ and choose the layer, which when binarized, maximizes the performance of the intermediate B2NN model.

C. Broader Impact

We proposed a new paradigm to perform partial binarization wherein layers are either full precision or full binary layers. As highlighted in our experiments, it results in improved model efficiency, enhanced model compression, performance gains in image classification and object tracking, and transferability across datasets. These broader impacts contribute to advancements in efficient and accurate neural networks, enabling their deployment in various resource-constrained scenarios and application domains. Moreover, the reduced footprint of such AI systems is particularly important because the demand for AI continues to grow, and every consumption is becoming a critical concern. Further, the improved inference speed of the devices due to model compression can enhance the user experience and enable new applications in areas such as real-time object tracking, medical diaganosis, among others. Since the B2NN models are light, these also allow sensitive data to be processed

locally rather than on cloud servers. This can enhance privacy by minimizing the sharing of personal data on external servers.

This highlights the need of more research focused on layer selection and the potential need of specialized hardware which can benefit from B2NNs. Also, binarization, which is one of the strongest compression techniques, will help reduce the hardware and computation constraints which inhibit the use of larger and complex models for production. Further, as machine learning continues to grow in scale and complexity, the environmental impact of training and inference becomes increasingly significant. Model compression techniques can reduce the computational resources required for training and inference, leading to lower energy consumption and a reduced carbon footprint.

D. Experiments: Additional Details

D.1. Choice of activation function

Table 1. Performance of B2NNs with different activation function obtained at various FLOPs on CIFAR100 with cResNet20. FLOPs for full precision network is 4.14×10^7.

Remaining	Activation function				
FLOPs (%)	Identity	ReLU	HardTanh	BinReLU	RPReLU
100.00	16.40	**65.44**	61.15	64.98	63.83
88.78	42.72	54.66	60.64	**64.53**	63.89
77.57	48.12	54.33	60.23	**64.72**	64.4
66.35	50.60	41.23	59.65	**64.66**	63.72
52.33	53.13	41.28	58.87	**63.90**	63.43
32.71	53.68	29.18	54.00	61.11	**61.65**
4.67	51.69	20.39	48.05	**53.72**	53.96

BinReLU activation function. The proposed BinReLU activation function is designed to enhance the stability of the full-precision as well as binary components of a B2NN in general. Mathematically, BinReLU can be stated as

$$f(x) = \begin{cases} -1 \text{ if } x \leq -1 \\ x \text{ otherwise} \end{cases}, \tag{1}$$

where x and $f(x)$ denote the input and output of the activation function.

Table 1 shows the BinReLU function together with ReLU, HardTanh, RPRelU and Identity mapping. For full-precision networks, ReLU is considered a very effective choice of activation, however, since it eliminates the activation information below 0, it does not work well for binary networks. For BNNs, either of HardTanh or Identity functions are preferred. However, both these activations do not work well for the real-valued networks (Table 1). Note that an identity mapping works well for BNN since for such cases, nonlinearity is inherently introduced through the squashing of the activation values to -1 and 1 using Sign$(\cdot)$. BinReLU is inspired from the other activations stated here in a sense that it preserves the characteristics of ReLU for positive activations and keeps them real-valued, and also ensures that the activation information between -1 and 0 is preserved.

D.2. Performance on simple classification datasets.

This section presents results obtained with `MixBin`, along with various model compression baselines, on the CIFAR-100 and TinyImageNet datasets. Baselines here include Network Slimming, HAWQ-V2, Adaptive MBQ, BNAS and random selection.

Table 3 shows performance scores for cResNet-20 for CIFAR-100. Additionally, we provide a sensitivity analysis for this setting, in which we calculate the error in accuracy by conducting three separate runs on different random seeds. Table 2 and 4 present result on TinyImageNet dataset.

D.3. Effect of binarizing different parts of a network.

The performance of the constructed B2NN model depends heavily on the choice of $\mathbf{\Phi}$. In Table 5, we compare the performance of MixBin with several trivial baselines.

Table 2. Performance scores for the MobileNetV1 architecture on TinyImageNet datasets for five compression methods: Network Slimming, HAWQ-V2, Adaptive MBQ, BNAS and `MixBin` (ours). The term "budget" refers to the percentage of FLOPs that remain after applying the respective compression method.

Method	Budget (%)	Acc. (%) ↑	FLOPs (%) ↓
Full Precision Network	-	52.85	100
Binary Network	-	34.28	4.67
Network Slimming	80	53.67	80
HAWQ-V2		51.86	81.03
Adaptive MBQ		50.43	81.72
BNAS		51.09	81.20
$\texttt{MixBin}_{Grad}$		56.62	80.15
$\texttt{MixBin}_{Loss}$		54.26	77.03
Random		50.05	81.32
Network Slimming	60	51.44	60
HAWQ-V2		49.82	61.88
Adaptive MBQ		49.69	61.88
BNAS		49.48	61.88
$\texttt{MixBin}_{Grad}$		52.11	63.18
$\texttt{MixBin}_{Loss}$		50.75	61.61
Random		49.27	61.88
Network Slimming	40	45.83	40
HAWQ-V2		46.10	41.24
Adaptive MBQ		43.03	41.03
BNAS		46.48	40.08
$\texttt{MixBin}_{Grad}$		48.50	39.68
$\texttt{MixBin}_{Loss}$		47.69	38.38
Random		42.3	39.45
Network Slimming	20	35.33	20
HAWQ-V2		39.31	21.43
Adaptive MBQ		39.73	9.13
BNAS		42.06	16.71
$\texttt{MixBin}_{Grad}$		41.12	13.83
$\texttt{MixBin}_{Loss}$		41.12	13.83
Random		38.34	20.31

Table 3. Performance scores for the cResNet-20 architecture on CIFAR-100 datasets for five compression methods: Network Slimming, HAWQ-V2, Adaptive MBQ, BNAS and `MixBin` (ours). The Full precision network has a total of 4.14×10^7 FLOPs, while the binary network generated using the Bi-RealNet method has 1.93×10^6 FLOPs. The term "budget" refers to the percentage of FLOPs that remain after applying the respective compression method.

Method	Budget(%)	Acc.(%)↑	FLOPs(%)↓	Error↓
Full Precision Network	-	64.76	100	0.48
Binary Network	-	53.18	4.67	0.45
Network Slimming	80	66.67	80	0.48
HAWQ-V2		65.90	77.57	0.18
Adaptive MBQ		65.77	77.57	0.15
BNAS		65.57	77.57	0.22
$\texttt{MixBin}_{Grad}$		65.72	77.57	0.24
$\texttt{MixBin}_{Loss}$		66.01	77.57	0.23
Random		65.03	80.37	0.53
Network Slimming	60	65.75	60	0.31
HAWQ-V2		65.77	60.75	0.38
Adaptive MBQ		65.70	60.75	0.18
BNAS		65.61	60.75	0.31
$\texttt{MixBin}_{Grad}$		65.85	60.75	0.19
$\texttt{MixBin}_{Loss}$		65.62	60.75	0.39
Random		65.31	60.75	0.34
Network Slimming	40	63.87	40	0.14
HAWQ-V2		64.97	38.31	0.35
Adaptive MBQ		61.96	38.31	2.25
BNAS		61.15	38.31	1.31
$\texttt{MixBin}_{Grad}$		65.32	38.31	0.42
$\texttt{MixBin}_{Loss}$		65.36	38.31	0.32
Random		61.11	43.92	0.89
Network Slimming	20	59.20	20	0.42
HAWQ-V2		62.03	21.49	0.17
Adaptive MBQ		58.46	18.69	1.93
BNAS		57.59	21.49	0.19
$\texttt{MixBin}_{Grad}$		63.76	21.49	0.47
$\texttt{MixBin}_{Loss}$		63.49	21.49	0.23
Random		58.56	21.49	0.63
Network Slimming	10	52.52	10	0.39
HAWQ-V2		58.41	13.12	0.31
Adaptive MBQ		56.86	10.28	1.81
BNAS		57.03	10.28	0.50
$\texttt{MixBin}_{Grad}$		60.27	11.08	0.50
$\texttt{MixBin}_{Loss}$		57.54	10.28	0.75
Random		56.47	10.28	0.59

Table 4. Performance scores for the ResNet-18 architecture on TinyImageNet datasets for five compression methods: Network Slimming, HAWQ-V2, Adaptive MBQ, BNAS and `MixBin` (ours). The Full precision network has a total of 5.63×10^8 FLOPs. The term "budget" refers to the percentage of FLOPs that remain after applying the respective compression method.

Method	Budget (%)	Acc. (%) ↑	FLOPs (%) ↓
Full Precision Network	-	56.88	100
Binary Network	-	44.43	4.67
Network Slimming	80	55.45	80
HAWQ-V2		57.58	81.25
Adaptive MBQ		57.55	81.25
BNAS		56.99	81.25
$\texttt{MixBin}_{Grad}$		57.11	81.25
$\texttt{MixBin}_{Loss}$		57.45	81.25
Random		56.89	81.25
Network Slimming	60	55.3	60
HAWQ-V2		56.80	62.50
Adaptive MBQ		56.71	62.50
BNAS		57.06	62.50
$\texttt{MixBin}_{Grad}$		56.67	62.50
$\texttt{MixBin}_{Loss}$		57.51	62.50
Random		56.6	62.50
Network Slimming	40	54.12	40
HAWQ-V2		55.70	43.76
Adaptive MBQ		56.10	43.76
BNAS		55.71	43.76
$\texttt{MixBin}_{Grad}$		56.44	43.76
$\texttt{MixBin}_{Loss}$		56.18	43.76
Random		56.0	40.62
Network Slimming	20	54.00	20
HAWQ-V2		54.36	21.88
Adaptive MBQ		54.58	25.01
BNAS		53.69	21.88
$\texttt{MixBin}_{Grad}$		55.08	21.88
$\texttt{MixBin}_{Loss}$		54.86	21.88
Random		54.25	24.12
Network Slimming	10	51.83	10
HAWQ-V2		53.57	12.51
Adaptive MBQ		53.87	12.51
BNAS		53.43	12.51
$\texttt{MixBin}_{Grad}$		54.25	12.51
$\texttt{MixBin}_{Loss}$		54.25	12.51
Random		54.02	10.92

Table 5. Performance comparision of B2NNs obtained using `MixBin` vs. the various trivial approaches. The B2NNs are constructed using cResNet-20 architecture on CIFAR-100 datasets. Here, *Rear-BN* and *Front-BN* represent B2NNs constructed with binary layer placed in the rear and front parts of the network, respectively. *Pseudo-random* involves modifying 30% of the MixBin generated B2NNs. The Full Precision network and Binary Network have 4.14×10^7 and 1.93×10^6 FLOPs respectively.

Method	Budget(%)	Acc.(%)↑	FLOPs(%)↓	Error↓
Full Precision Network	-	64.76	100	0.48
Binary Network	-	53.18	4.67	0.45
Front-BN	80	65.12	83.17	0.52
Rear-BN		63.51	83.17	1.31
Pseudo-random		65.18	83.17	0.07
$\texttt{MixBin}_{Grad}$		65.72	77.57	0.24
$\texttt{MixBin}_{Loss}$		66.01	77.57	0.23
Front-BN	60	64.13	63.54	0.37
Rear-BN		59.32	63.54	0.02
Pseudo-random		64.60	63.54	0.22
$\texttt{MixBin}_{Grad}$		65.85	60.75	0.19
$\texttt{MixBin}_{Loss}$		65.62	60.75	0.39
Front-BN	40	63.64	41.12	0.59
Rear-BN		57.78	41.12	0.43
Pseudo-random		64.28	38.31	0.62
$\texttt{MixBin}_{Grad}$		65.32	38.31	0.42
$\texttt{MixBin}_{Loss}$		65.36	38.31	0.32
Front-BN	20	61.79	21.49	0.66
Rear-BN		56.49	21.49	0.60
Pseudo-random		61.40	18.68	0.49
$\texttt{MixBin}_{Grad}$		63.76	21.49	0.47
$\texttt{MixBin}_{Loss}$		63.49	21.49	0.23
Front-BN	10	59.28	10.28	0.41
Rear-BN		55.91	10.28	0.56
Pseudo-random		55.28	10.28	0.57
$\texttt{MixBin}_{Grad}$		60.27	11.08	0.50
$\texttt{MixBin}_{Loss}$		57.54	10.28	0.75