Equal Bits: Enforcing Equally Distributed Binary Network Weights

Yunqiang Li*, Silvia L. Pintea* and Jan C. van Gemert

Computer Vision Lab, Delft University of Technology,
Delft, Netherlands

Abstract

Binary networks are extremely efficient as they use only two symbols to define the network: $\{+1, -1\}$. One can make the prior distribution of these symbols a design choice. The recent IR-Net of Qin *et al.* argues that imposing a Bernoulli distribution with equal priors (equal bit ratios) over the binary weights leads to maximum entropy and thus minimizes information loss. However, prior work cannot precisely control the binary weight distribution during training, and therefore cannot guarantee maximum entropy. Here, we show that quantizing using optimal transport can guarantee any bit ratio, including equal ratios. We investigate experimentally that equal bit ratios are indeed preferable and show that our method leads to optimization benefits. We show that our quantization method is effective when compared to state-of-the-art binarization methods, even when using binary weight pruning. Our code is available at https://github.com/liyunqianggyn/Equal-Bits-BNN.

1 Introduction

Binary networks allow compact storage and swift computations by limiting the network weights to only two bit symbols $\{-1, +1\}$. In this paper we investigate weights priors before seeing any data: is there a reason to prefer predominantly positive bit weights? Or more negative ones? Or is equality preferable? Successful recent work (Qin et al. 2020b) argues that a good prior choice is to have an equal bit-ratio: i.e. an equal number of $+1$ and -1 symbols in the network. This is done by imposing an equal prior under the standard Bernoulli distribution (Qin et al. 2020b; Peters and Welling 2018; Zhou et al. 2016). Equal bit distributions minimize information loss and thus maximizes entropy, showing benefits across architectures and datasets (Qin et al. 2020b). However, current work cannot add a hard constraint of making symbol priors exactly equal, and therefore cannot guarantee maximum entropy.

Here, we propose a method to add a hard constraint to binary weight distribution, offering precise control for any desired bit ratio, including equal prior ratios. We add hard constraints in the standard quantization setting (Bulat and Tzimiropoulos 2019; Qin et al. 2020b; Rastegari et al. 2016) making use of real-valued latent weights that approximate the binary weights. We quantize these real-valued weights by aligning them to any desired prior Bernoulli distribution, which incorporates our preferred binary weight prior. Our quantization uses optimal transport (Villani 2003) and can guarantee any bit ratio. Our method makes it possible to experimentally test the hypothesis in (Qin et al. 2020b) that equal bit ratios are indeed preferable to other bit ratios. We baptize our approach with equal bit ratios: *bi-half*. Furthermore, we show that enforcing equal priors using our approach leads to optimization benefits by reducing the problem search-space and avoiding local minima.

We make the following contributions: (i) a binary network optimization method based on optimal transport; (ii) exact control over weight bit ratios; (iii) validation of the assumption that equal bit ratios are preferable; (iv) optimization benefits such as search-space reduction and good minima; (v) favorable results compared to the state-of-the-art, and can ensure half-half weight distribution even when pruning is used.

2 Related Work

For a comprehensive survey on binary networks, see (Qin et al. 2020a). In Table 1 we show the relation between our proposed method and pioneering methods, that are representatives of their peers, in terms of the binarization choices made. The XNOR method (Table 1(a)) was the first to propose binarizing latent real-valued weights using the sign function (Rastegari et al. 2016). Rather than making each binary weight depend only on its associated real-value weight or gradient value, IR-Net (Qin et al. 2020b) (Table 1(b)) is a prototype method that uses filter-weight statistics to update each individual binary weight. Here, we also use filter-weight statistics to update the binary weights, however similar to (Helwegen et al. 2019) Table 1(d)) we do not rely on the sign function for binarization, but instead use binary weight flips. This is a natural choice, as flipping the sign of a binary weight is the only operation one can apply to binary weights.

Sign versus bit flips. The front-runners of binary networks are BinaryConnect (Courbariaux, Bengio, and David 2015) and XNOR (Rastegari et al. 2016) and rely on auxiliary real weights and the sign function to define binary weights.

*Both authors contributed equally.

	ⓐ Sign, **no filter statistics** (Rastegari et al. 2016)	ⓑ Sign, **filter statistics** (Qin et al. 2020b)	ⓒ **Flip, filter statistics** Our *bi-half*	ⓓ **Flip, no filter statistics** (Helwegen et al. 2019)
Initialization	Gradient g; Latent weight w;	Gradient g; Latent weight w;	Gradient g; Latent weight w; Threshold dependent on w;	Gradient g; Predefined threshold τ;
Binarization, b	$b \leftarrow \text{sign}(w)$	$b \leftarrow \text{sign}\left(\frac{w-\text{avg}(w)}{\text{std}(w-\text{avg}(w))}\right)$	$b \leftarrow \text{flip}(b),\ \text{if} \begin{cases} \text{rank}(w) < \frac{D}{2}, \text{ and } \text{rank}(w-\alpha g) \geq \frac{D}{2} \\ \text{rank}(w) \geq \frac{D}{2}, \text{ and } \text{rank}(w-\alpha g) < \frac{D}{2} \end{cases}$	$b \leftarrow \text{flip}(b), \text{if} \begin{cases} \tau < \lvert g \rvert, \text{ and} \\ \text{sign}(g) = \text{sign}(b) \end{cases}$

Table 1: **Optimization perspectives.** (a) Classical binarization methods tie each binary weight b to an associated real-valued latent variable w, and quantize each weight by only considering its associated real-valued by using the sign function. (b) Rather than updating the weights independent of each other, recent work uses filter-weight statistics when updating the binary weights. (c) Our proposed optimization method does not focus on using the sign function, but rather flips the binary weights based on the distribution of the real weights, thus the binary weight updates depend on the statistics of the other weights through the rank of w. (d) Recent work moves away from using the sign of the latent variables, and instead trains the binary network with bit sign flips, however they still consider independent weight updates.

These works are subsequently extended with focus on the scaling factors in XNOR++ (Bulat and Tzimiropoulos 2019) and BNN+ (Darabi et al. 2019), while HWGQ (Li et al. 2017) uses the sign function recursively for binarization. Bi-Real (Liu et al. 2018) also uses the sign function for binarization and analyzes better approximations of the gradient of the sign function. From a different perspective, recent work tries to sidestep having to approximate the gradient of the sign function, and uses bit flips to train binary networks (Helwegen et al. 2019). The sign of the binary weights can be flipped based on searchable (Yang et al. 2020) or learnable thresholds (Liu et al. 2020). Here, we also rely on bit flips based on a dynamic thresholding of the real weights, entailed by our optimal transport optimization strategy.

Using filter statistics or not. Commonly, binarization methods define each binary weight update by considering only its associated value in the real-valued latent weights (Bulat and Tzimiropoulos 2019; Li et al. 2017; Rastegari et al. 2016; Liu et al. 2018) or in the gradient vector (Helwegen et al. 2019). However, binary weights can also be updated using explicit statistics of the other weights in the filter (Lin, Zhao, and Pan 2017) or implicitly learned through a function (Han et al. 2020). The real-valued filter statistics are used in IR-Net (Qin et al. 2020b) to enforce a Bernoulli distribution with equal priors. Similarly, our optimal transport optimization leads to ranking the real weights, and therefore making use of the statistics of the real-weights in each filter.

Network pruning. Pruning has been shown to improve the efficiency of deep networks (Frankle et al. 2020; Huang et al. 2018; Lin et al. 2017; Xiao, Wang, and Rajasekaran 2019; Ye et al. 2020). However, the reason why pruning can bring improvements remains unclear in real-valued networks. It is commonly believed (Evci et al. 2020; Frankle and Carbin 2020; Malach et al. 2020; Zhou et al. 2019) that finding the "important" weight values is crucial for retraining a small pruned model. Specifically, the "important" weight values are inherited (Han, Mao, and Dally 2016) or re-winded (Frankle and Carbin 2020) from a large trained model. In contrast, (Liu et al. 2019) claims that the selected important weights are typically not useful for the small pruned model, while the pruned architecture itself is more relevant. The lottery-ticket idea has recently been applied to binary networks (Diffenderfer and Kailkhura 2021). Here, we show that having equal $+1$ and -1 ratios is also optimal when the networks rely on pruning and that our optimal transport optimization can easily be adapted to work with methods using pruning.

3 Binarizing with optimal transport

3.1 Binary weights

We define a binary network where the weights $\mathbf{B}$ take binary values $\{1, -1\}^D$. The binary weights $\mathbf{B}$ follow a Bernoulli distribution $\mathbf{B} \sim \text{Be}(p_{pos})$, describing the probabilities of individual binary values $b \in \{-1, 1\}$ in terms of the hyper-parameters p_{pos} and p_{neg}:

$$p(b) = \text{Be}(b \mid p_{pos}) = \begin{cases} p_{pos} & \text{if } b = +1 \\ p_{neg} = 1 - p_{pos}, & \text{if } b = -1 \end{cases} \tag{1}$$

To be consistent with previous work, we follow XNOR-Net (Rastegari et al. 2016) and apply the binary optimization per individual filter.

Because the matrix $\mathbf{B}$ is discrete, we follow (Courbariaux, Bengio, and David 2015; Rastegari et al. 2016) by using real-valued latent weights $\mathbf{W}$ to aid the training of discrete values, where each binary weight in $\mathbf{B}$ has an associated real-valued weight in $\mathbf{W}$. In the forward pass we quantize the real-valued weights $\mathbf{W}$ to estimate the matrix $\mathbf{B}$. Then, we use the estimated matrix $\mathbf{B}$ to compute the loss, and in the backward pass we update the associated real-valued weights $\mathbf{W}$.

3.2 Optimal transport optimization

The optimization aligns the real-valued weight distribution $\mathbf{W}$ with the prior Bernoulli distribution in Eq. (1) and quantizes the real-valued weights $\mathbf{W}$ to $\mathbf{B}$.

The empirical distribution $\mathbb{P}_w$ of the real-valued variable $\mathbf{W} \in \mathbb{R}^D$ and the empirical distribution $\mathbb{P}_b$ for the discrete variable $\mathbf{B}$ can be written as:

$$\mathbb{P}_w = \sum_{i=1}^{D} p_i \delta_{w_i}, \quad \mathbb{P}_b = \sum_{j=1}^{2} q_j \delta_{b_j}, \tag{2}$$

where $\delta_{\mathbf{x}}$ is the Dirac function at location $\mathbf{x}$. The p_i and q_j are the probability mass associated to the corresponding distribution locations w_i and b_j, where $\mathbb{P}_b$ has only 2 possible locations in the distribution space $\{-1, 1\}$.

To align $\mathbb{P}_w$ with the Bernoulli prior $\mathbb{P}_b$ in Eq. (1) we use optimal transport (OT) (Villani 2003) which minimizes the cost of moving the starting distribution $\mathbb{P}_w$ to the target distribution $\mathbb{P}_b$. Because $\mathbb{P}_w$ and $\mathbb{P}_b$ are only accessible through a finite set of values, the corresponding optimal transport cost is:

$$\pi_0 = \min_{\pi \in \Pi(\mathbb{P}_w, \mathbb{P}_b)} \langle \pi, \mathcal{C} \rangle_F , \tag{3}$$

where $\Pi(\mathbb{P}_w, \mathbb{P}_b)$ is the space of the joint probability with marginals $\mathbb{P}_w$ and $\mathbb{P}_b$, and π is the general probabilistic coupling that indicates how much mass is transported to push distribution $\mathbb{P}_w$ towards the distribution $\mathbb{P}_b$. The $\langle ., . \rangle_F$ denotes the Frobenius dot product, and $\mathcal{C} \geq 0$ is the cost function matrix whose element $\mathcal{C}(w_i, b_j)$ denotes the cost of moving a probability mass from location w_i to location b_j in distribution space. When the cost is defined as a distance, the OT becomes the Wasserstein distance. We minimize the 1-Wasserstein distance between $\mathbb{P}_w$ and $\mathbb{P}_b$. This minimization has an elegant closed-form solution based on simply sorting. For a continuous-valued weights vector $\mathbf{W} \in \mathbb{R}^D$, we first sort the elements of $\mathbf{W}$, and then assign the top $p_{pos}D$ elements to $+1$, and the bottom $(1 - p_{pos})D$ portion of the elements to -1:

$$\mathbf{B} = \pi_0 (\mathbf{W}) = \begin{cases} +1, & \text{top } \mathrm{p_{pos}D} \text{ of sorted } \mathbf{W} \\ -1, & \text{bottom } (1 - \mathrm{p_{pos}})\mathrm{D} \text{ of sorted } \mathbf{W} \end{cases} \tag{4}$$

Rather than using the sign function to define the binarization, we flip the binary weights based on the distribution of $\mathbf{W}$. Thus the flipping of a binary weight depends on the distribution of the other binary weights through $\mathbf{W}$, which is optimized to be as close as possible to $\mathbf{B}$.

When applying our method in combination with pruning as in (Diffenderfer and Kailkhura 2021), we first mask the binary weights $\mathbf{B}' = \mathbf{M} \odot \mathbf{B}$ with a mask $\mathbf{M} \in \{0, 1\}^D$. This leads to a certain percentage of the weights being pruned. Subsequently, we apply the Eq. (4) to the remaining non-pruned weights, where D in Eq. (4) become the L_1 norm of the mask, $|\mathbf{M}|$.

3.3 *Bi-half*: Explicitly controlling the bit ratio

Our optimal transport optimization allows us to enforce a hard constraint on precise bit ratios by varying the p_{pos} value. Therefore, we can test a range of prior binary weight distributions.

Following (Qin et al. 2020b), a good prior over the binary weights is one maximizing the entropy. Using optimal transport, we maximize the entropy of the binary weights by setting the bit ratio to half in Eq. (4):

$$p^*_{pos} = \text{argmax}_{p_{pos}} H(\mathbf{B} \sim \text{Be}(p_{pos})) = \frac{1}{2}, \tag{5}$$

where $H(\cdot)$ denotes the entropy of the binary weights $\mathbf{B}$. We dub this approach *bi-half*. Unlike previous work (Qin et al. 2020b), we can guarantee equal symbol distributions and therefore maximum entropy throughout the complete training procedure.

Initialization and scaling factor. We initialize the real-valued weights using Kaiming normal (He et al. 2015). The binary weights are initialized to be equally distributed per filter according to Eq. (5). To circumvent exploding gradients, we use one scaling factor α per layer for the binary weights to keep the activation variance in the forward pass close to 1. Based on the ReLU variance analysis in (He et al. 2015) it holds that $\frac{1}{2} D \cdot Var(\alpha \mathbf{B}) = 1$, where D is the number of connections and $\mathbf{B}$ are our binary weights. $\mathbf{B}$ is regularized to a *bi-half* distribution, thus $Var(\mathbf{B}) = 1$, which gives $\alpha = \sqrt{2/D}$.

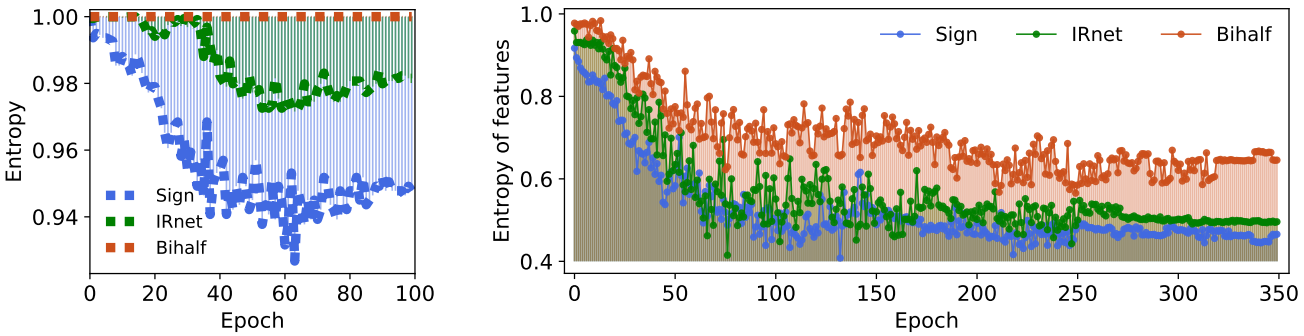
To better clarify, for an L-layer network with input data y_1 standardized to $Var[y_1] = 1$, where the variance of each binary layer l is $Var[\mathbf{B}_l] = 1$, and D_l is the number of connections in that layer: *i)* Without the scaling, the output variance is $Var[y_L] = Var[y_1] \prod_{l=2}^{L} \frac{D_l}{2} Var[\mathbf{B}_l] = \prod_{l=2}^{L} \frac{D_l}{2}$. Typically D_l is large, leading to exploding gradients; *ii)* With the scaling, we scale $\mathbf{B}_l$ by $\alpha = \sqrt{2/D_l}$, leading to $Var[y_L] = Var[y_1] \prod_{l=2}^{L} \frac{D_l}{2} Var(\alpha \mathbf{B}_l) = 1$ which stabilizes learning.

4 Experiments

Datasets and implementation details. We evaluate on Cifar-10, Cifar-100 (Krizhevsky, Hinton et al. 2009) and ImageNet (Deng et al. 2009), for a number of network architectures. Following (Frankle and Carbin 2020; Ramanujan et al. 2020) we evaluate 4 shallow CNNs: Conv2, Conv4, Conv6, and Conv8 with 2/4/6/8 convolutional layers. We train the shallow models on Cifar-10 for 100 epochs, with weight decay $1e^{-4}$, momentum 0.9, batch size 128, and initial learning rate 0.1 using a cosine learning rate decay (Loshchilov and Hutter 2016). Following (Qin et al. 2020b) we also evaluate their ResNet-20 architecture and settings on Cifar-10. On Cifar-100, we evaluate our method on 5 different models including VGG16 (Simonyan and Zisserman 2015), ResNet18 (He et al. 2016), ResNet34 (He et al. 2016), InceptionV3 (Szegedy et al. 2016), ShuffleNet (Zhang et al. 2018). We train the Cifar-100 models for 350 epochs using SGD with weight decay $5e^{-4}$, momentum 0.9, batch size 128, and initial learning rate 0.1 divided by 10 at epochs 150, 250 and 320. For ImageNet we use ResNet-18 and ResNet-34 trained for 100 epochs using SGD with momentum 0.9, weight decay $1e^{-4}$, and batch size 256. Following (Liu et al. 2018; Qin et al. 2020b), the initial learning rate is set as 0.1 and we divide it by 10 at epochs 30, 60, 90. All our models are trained from scratch without any pre-training. For the shallow networks we apply our method on all layers, while for the rest we follow (Liu et al. 2018; Qin et al. 2020b), and apply it on all convolutional and fully-connected layers except the first, last and the downsampling layers.

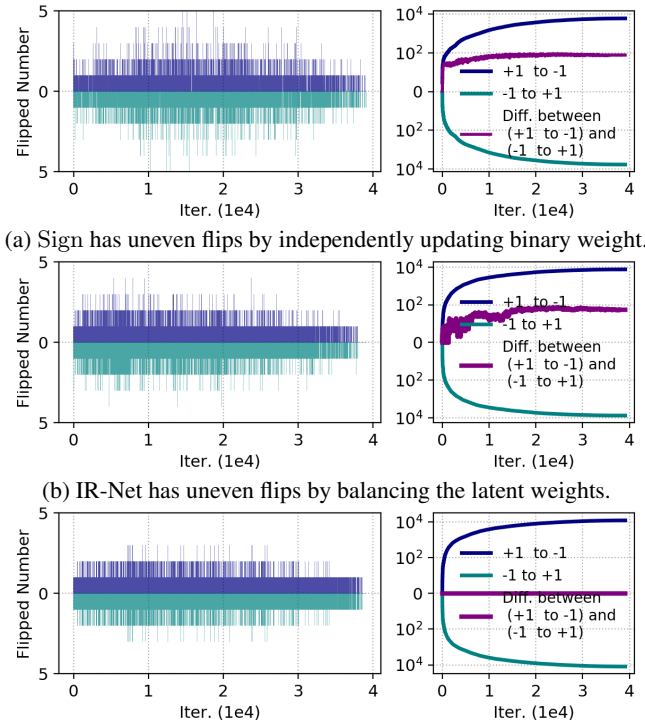
4.1 Hypothesis: *Bi-half* maximizes the entropy

Here we test whether our proposed *bi-half* model can indeed guarantee maximum entropy and therefore an exactly equal ratio of the -1 and $+1$ symbols. Fig. 2 shows the bit flips performed in our proposed *bi-half* method during training when compared to two baselines: sign (Rastegari et al. 2016) and IR-Net (Qin et al. 2020b). We train a Conv2 network on Cifar-10 and plot the flips of binary weights in a single binary weight filter during training. The binary weights



(a) Binary weight entropy on Conv2. (b) Activation entropy on ResNet-18.

Figure 1: **Hypothesis: bi-half maximizes the entropy.** Entropy of binary weights and activations. We compare our bi-half method to $sign$ (Rastegari et al. 2016) and IR-Net (Qin et al. 2020b). (a) Entropy of the binary weights during training for Conv2 on Cifar10. (b) Entropy of the network activations for ResNet-18 on Cifar100. Our *bi-half* model can guarantee maximum entropy during training for the binary weight distribution and it is able to better maximize the entropy of the activations.



(a) Sign has uneven flips by independently updating binary weight.

(b) IR-Net has uneven flips by balancing the latent weights.

(c) *Bi-half* has even flips by precisely controlling the flipping.

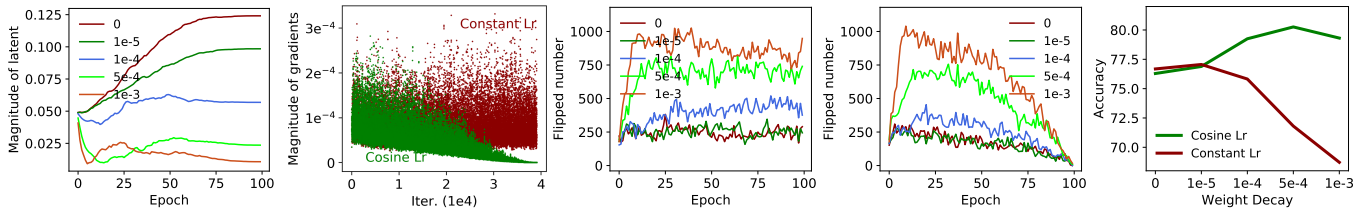
Figure 2: **Hypothesis: *bi-half* maximizes the entropy.** Bit flips during training. We compare the bit flips during training in our *bi-half* with the sign (Rastegari et al. 2016) and IR-Net (Qin et al. 2020b) on the Conv2 network on Cifar-10. The x-axis shows the training iterations. *Left:* Bit flips during training to +1 (dark blue) or to -1 (cyan). *Right:* Accumulated bit flips over the training iterations, as well as the difference between the bit flips from $(+1$ to $-1)$ and the ones from $(-1$ to $+1)$. In contrast to sign and IR-Net, our *bi-half* method can guarantee an equal bit ratio.

are initialized to be equal distributed (half of the weights positive and the other half negative). The classical sign method (Rastegari et al. 2016) in Fig. 2(c) binarizes each weight independent of the other weights, therefore during training the flips for $(+1$ to $-1)$ and $(-1$ to $+1)$ are uneven. The recent IR-Net (Qin et al. 2020b) in Fig. 2(b) balances the latent weights by using their statistics to obtain evenly distributed binary weight values. However, it can not guarantee evenly distributed binary weights throughout training. Our *bi-half* model in Fig. 2(c) updates the binary weight based on the statistics of the other weights. For our method the binary weights are evenly flipped during training, offering exact control of bit weight ratios.

Fig. 1(a) shows the binary weights entropy changes during training on Conv2 when compared to sign (Rastegari et al. 2016) and IR-Net (Qin et al. 2020b). IR-Net aims to maximize entropy by subtracting the mean value of the weights, yet, this is not exact. In contrast, we maximize the information entropy by precisely controlling the binary weight distribution. In Fig. 1(b) we show the entropy of the binary activations. Adjusting the distribution of binary weights retains the information in the binary activation. For our *bi-half* method, the binary activation of each channel is close to the maximum information entropy under the Bernoulli distribution.

4.2 Empirical analysis

(a) Effect of hyper-parameters. In Fig. 3 we study the effectiveness of the commonly used training techniques of varying the weight decay and learning rate decay, when training the Conv2 network on Cifar-10. Fig. 3(a) shows that using a higher weight decay reduces the magnitude of latent weights during training and therefore the magnitude of the cut-off point (threshold) between the positive and negative values. Fig. 3(b) compares the gradient magnitude of two different learning rate (lr) schedules: "constant lr" and "cosine lr". The magnitude of the gradients reduces during training when using the cosine learning rate. In Fig. 3(c) we find that increasing the weight decay for binary network with a constant learning rate schedule, increases binary weights flips.



(a) Latent weights magnitude (b) Gradient magnitude (c) flips with constant lr. (d) flips with cosine lr. (e) accuracy

Figure 3: **Empirical analysis (a): Effect of hyper-parameters.** We show the effect of weight decay and learning rate decay on binary weights flips using the Conv2 network on Cifar-10. Carefully tuning these hyper-parameters is important for adequately training the binary networks.

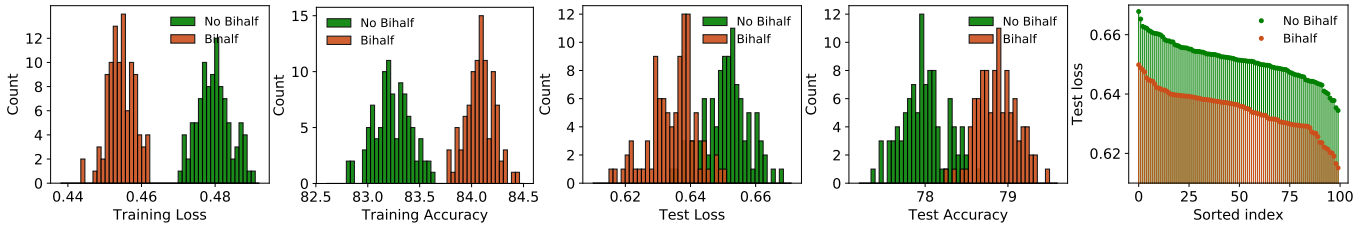


Figure 4: **Empirical analysis (c): Optimization benefits.** We train our *bi-half* model 100 times on Cifar-10 and plot the distribution of the losses and accuracies over the 100 repetitions. We compare our results using optimal transport to the results using the standard sign function. On average our *bi-half* model tends to arrive at better losses and accuracies than the baseline.

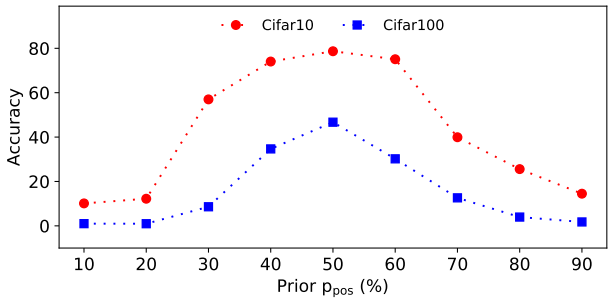
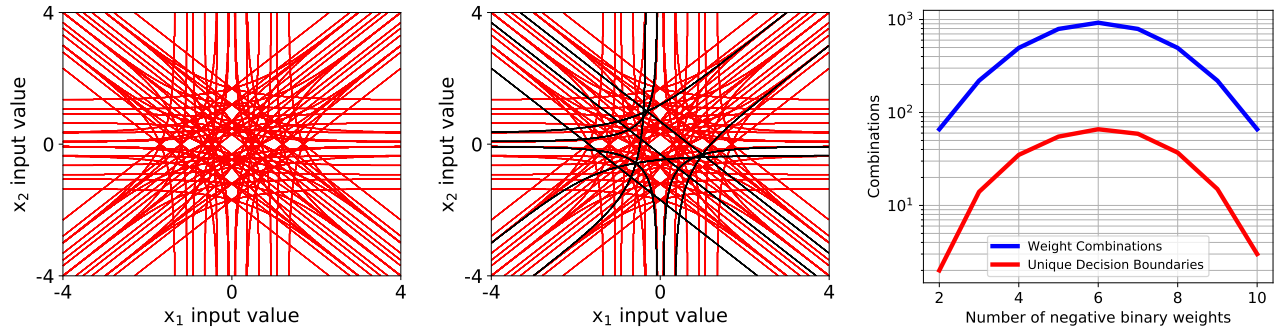


Figure 5: **Empirical analysis (b): Which bit-ratios are preferred?** We varying the bit-ratios on Cifar-10 and Cifar-100 using Conv2 the choice of the priorp_{pos} under the Bernoulli distribution. The x-axis is the probability of the $+1$ connections denoted by p_{pos} in the Bernoulli prior distribution, while the y-axis denotes the top-1 accuracy values. Results are in agreement with the hypothesis of Qin *et al.* (Qin et al. 2020b) that equal priors as imposed in our *bi-half* model are preferable.

Fig. 3(d) shows that decaying the learning rate when using a cosine learning rate schedule gradually decreases the number of flipped weights. Fig. 3(e) shows that the choice of weight decay and learning rate decay affect each other. Our bi-half method uses the rank of latent weights to flip the binary weights. A proper tuned hyper-parameter of weight decay and learning rate decay will affect the flipping threshold. Therefore in the experiments, we carefully tune the hyper-parameters of weight decay and learning rate decay to build a competitive baseline.

(b) Which bit-ratio is preferred? In Fig. 5, we evaluate the choice of the prior p_{pos} in the Bernoulli distribution for Conv2 on Cifar-10 and Cifar-100. By varying the bit-ratio, the best performance is consistently obtained when the negative and positive symbols have equal priors as in the *bi-half* model. Indeed, as suggested in (Qin et al. 2020b), when there is no other a-priori reason to select a different p_{pos}, having equal bit ratios is a good choice.

(c) Optimization benefits with bi-half. The uniform prior over the -1 and $+1$ under the Bernoulli distribution regularizes the problem space, leading to only a subset of possible weight combinations available during optimization. We illustrate this intuitively on a $2D$ example for a simple fully-connected neural network with one input layer, one hidden layer, and one output layer in a two-class classification setting. We consider a $2D$ binary input vector $\mathbf{x} = [x_1, x_2]^\intercal$, and define the network as: $\sigma(\mathbf{w_2}^\intercal \sigma(\mathbf{w_1}^\intercal \mathbf{x} + \mathbf{b_1}))$, where $\sigma(\cdot)$ is a sigmoid nonlinearity, $\mathbf{w_1}$ is a $[2 \times 3]$ binary weight matrix, $\mathbf{b_1}$ is $[3 \times 1]$ binary bias vector, and $\mathbf{w_2}$ is a $[3 \times 1]$ binary vector. We group all 12 parameters as a vector $\mathbf{B}$. We enumerate all possible binary weight combinations in $\mathbf{B}$, *i.e.* $2^{12} = 4096$, and plot all decision boundaries that separate the input space into two classes as shown in Fig. 6(a). All possible 4096 binary weights combinations offer only 76 unique decision boundaries. In Fig. 6.(b) the Bernoulli distribution over the weights with equal prior (*bi-half*) regularizes the problem space: it reduces the weight combinations to 924, while retaining 66 unique solutions, therefore the ratio of the solutions to the complete search spaces is increased nearly 4 times. Fig. 6.(c) shows in a half-log plot how the numbers of weight combinations and unique network solutions change with varying bit-ratios. Equal bit ratios is optimal.



Weight combinations: 100% (4096).
Unique solutions: 100% (76).
(a) Full binary network

Weight combinations: 22.5% (924).
Unique solutions: 86.8% (66).
(b) *Bi-half* network

(c) Solution space vs bit-ratios

Figure 6: **Empirical analysis (c): Optimization benefits.** *Bi-half* regularization: 2D example for a 12-parameter fully connected binary network $\sigma(\mathbf{w_2}^\intercal \sigma(\mathbf{w_1}^\intercal \mathbf{x} + \mathbf{b_1}))$, where $\sigma(\cdot)$ is a sigmoid nonlinearity. Weights are in $\{-1, 1\}$. (a) Enumeration of all decision boundaries for 12 binary parameters ($4096 = 2^{12}$ combinations). (b) Weight combinations and unique solutions when using our *bi-half* constraint. (c) The weight combinations and unique decision boundaries for various bit-ratios. When the number of negative binary weights is 6 on the x-axis, we have equal bit-ratios, which is the optimal ratio. Using the bi-half works as a regularization, reducing the search-space while retaining the majority of the solutions.

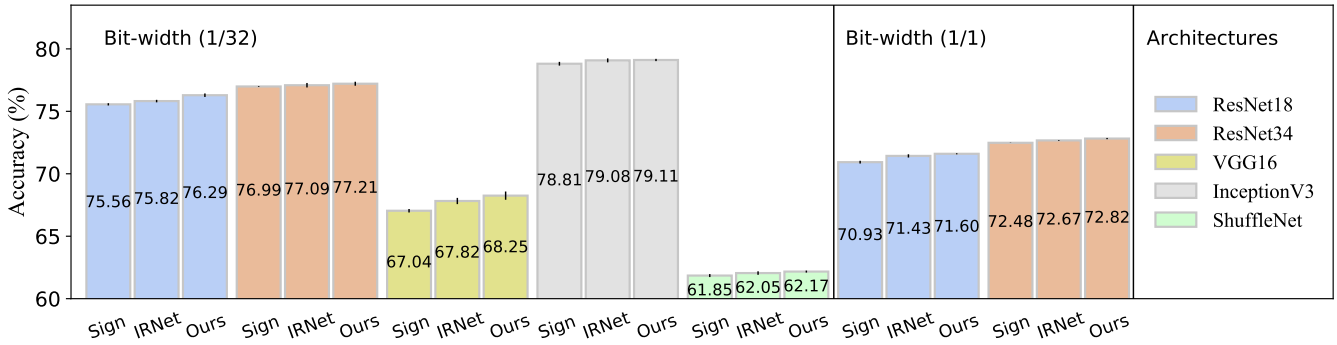


Figure 7: **Architecture variations: Different architectures on Cifar-100.** We evaluate on Cifar-100 over 5 different architectures: VGG16 (Simonyan and Zisserman 2015), ResNet18 (He et al. 2016), ResNet34 (He et al. 2016), InceptionV3 (Szegedy et al. 2016), ShuffleNet (Zhang et al. 2018). We compare sign (Rastegari et al. 2016), IR-Net (Qin et al. 2020b) and our *bi-half*. The 1/32 and 1/1 indicate the bit-width for weights and for activations, where 1/1 means we quantize both the weights and the activations to binary code values. Our method achieves competitive accuracy across different network architectures.

In Fig. 4 we train the Conv2 networks 100 times on Cifar-10 and plot the distribution of the training and test losses and accuracies. We plot these results when using the *bi-half* model optimization with optimal transport and by training the network using the standard sign function. The figure shows the *bi-half* method consistently finds better solutions with lower training and test losses and higher training and test accuracy. To better visualize this trend we sort the values of the losses for our *bi-half* and the baseline sign method over the 100 repetitions and plots them next to each other. On average the *bi-half* finds better optima.

4.3 Architecture variations

In Table 2 we compare the Sign (Rastegari et al. 2016), IR-Net (Qin et al. 2020b) and our *bi-half* on four shallow Conv2/4/6/8 networks on Cifar-10 (averaged over 5 trials). As the networks become deeper, the proposed *bi-half* method consistently outperforms the other methods.

In Fig. 7, we further evaluate our method on Cifar-100 over 5 different architectures: VGG16 (Simonyan and Zisserman 2015), ResNet18 (He et al. 2016), ResNet34 (He et al. 2016), InceptionV3 (Szegedy et al. 2016), ShuffleNet (Zhang et al. 2018). Our method is slightly more accurate than the other methods, especially on the VGG16 architecture, it never performs worse.

4.4 Comparison with state-of-the-art

(a) Comparison on ImageNet. For the large-scale ImageNet dataset we evaluate a ResNet-18 and ResNet-34 backbone (He et al. 2016). Table 3 shows a number of state-of-the-art quantization methods over ResNet-18 and ResNet-34, including: ABC-Net (Lin, Zhao, and Pan 2017), XNOR (Rastegari et al. 2016), BNN+ (Darabi et al. 2019), Bi-Real (Liu et al. 2018), RBNN (Lin et al. 2020), XNOR++ (Bulat and Tzimiropoulos 2019), IR-Net (Qin

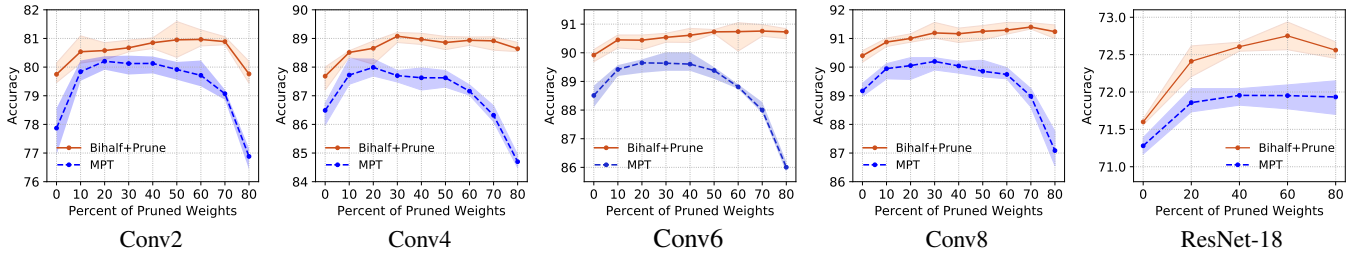


Figure 8: **Comparison with state-of-the-art (b): Pruned networks.** Test accuracy of Conv2/4/6/8 on CIFAR-10, and ResNet-18 on CIFAR-100 when varying the % pruned weights. We compare with the MPT baseline (Diffenderfer and Kailkhura 2021) using binary weight masking and the sign function. Having equal $+1$ and -1 ratios is also optimal when the networks rely on pruning and that our optimal transport optimization can easily be adapted to work in combination with pruning.

Method	Conv2	Conv4	Conv6	Conv8
Sign	77.86± 0.69	86.49± 0.24	88.51± 0.35	89.17±0.26
IR-Net	78.32± 0.25	87.20± 0.26	89.61± 0.11	90.06±0.06
Bi-half (ours)	79.25± 0.28	87.68± 0.32	89.92± 0.19	90.40±0.17

Table 2: **Architecture variations.** Accuracy comparison of sign (Rastegari et al. 2016), IR-Net (Qin et al. 2020b) and our *bi-half* on Conv2/4/6/8 networks using Cifar-10, over 5 repetitions. As the depth of the network increases, the accuracy of our method increases.

et al. 2020b), and Real2binary (Martinez et al. 2020). Of all the methods, RBNN is the closest in accuracy to our *bi-half* model. This is because RBNN relies on the sign function but draws inspiration from hashing, and adds an activation-aware loss to change the distribution of the activations before binarization. On the other hand, our method uses the standard classification loss but outperforms most other methods by a large margin on both ResNet-18 and ResNet-34 architectures.

(b) Comparison on pruned networks. In Fig. 8 we show the effect of our *bi-half* on pruned models. Following the MPT method (Diffenderfer and Kailkhura 2021) we learn a mask for the binary weights to prune them. However, in our *bi-half* approach for pruning we optimize using optimal transport for equal bit ratios in the remaining unpruned weights. We train shallow Conv2/4/6/8 networks on CIFAR-10, and ResNet-18 on CIFAR-100 while varying the percentage of pruned weights. Each curve is the average over five trials. Pruning consistently finds subnetworks that outperform the full binary network. Our *bi-half* method with optimal transport retains the information entropy for the pruned subnetworks, and consistently outperforms the MPT baseline using the sign function for binarization.

5 Conclusion

We focus on binary networks for their well-recognized efficiency and memory benefits. To that end, we propose a novel method that optimizes the weight binarization by aligning a real-valued proxy weight distributions with an idealized distribution using optimal transport. This optimization allows to test which prior bit ratio is preferred in a binary network, and we show that the equal bit ratios, as advertised by (Qin et al. 2020b), indeed work better. We confirm that our optimal transport binarization has optimization benefits such as: reducing the search space and leading to better local optima. Finally, we demonstrate competitive performance when compared to state-of-the-art, and improved accuracy on 3 different datasets and various architectures. We additionally show accuracy gains with pruning techniques.

Backbone	Method	Bit-width (W/A)	Top-1(%)	Top-5(%)
ResNet-18	FP	32/32	69.3	89.2
	ABC-Net	1/1	42.7	67.6
	XNOR	1/1	51.2	73.2
	BNN+	1/1	53.0	72.6
	Least-squares	1/1	58.9	81.4
	XNOR++	1/1	57.1	79.9
	IR-Net	1/1	58.1	80.0
	RBNN	1/1	59.9	81.9
	Sign (Baseline)	1/1	59.98	82.47
	Bi-half (ours)	1/1	**60.40**	**82.86**
ResNet-34	FP	32/32	73.3	91.3
	ABC-Net	1/1	52.4	76.5
	Bi-Real	1/1	62.2	83.9
	IR-Net	1/1	62.9	84.1
	RBNN	1/1	63.1	84.4
	bi-half (ours)	1/1	**64.17**	**85.36**

Table 3: **Comparison with state-of-the-art (a): ImageNet results.** We show Top-1 and Top-5 accuracy on ImageNet for a number of state-of-the-art binary networks. Sign is our baseline by carefully tuning the hyper-parameters. Our proposes *bi-half* model consistently outperforms the other binarization methods on this large-scale classification task.

References

Bulat, A.; and Tzimiropoulos, G. 2019. XNOR-Net++: Improved Binary Neural Networks. arXiv:1909.13863. 1, 2, 6

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Bi-

naryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*. 1, 2

Darabi, S.; Belbahri, M.; Courbariaux, M.; and Nia, V. P. 2019. Bnn+: Improved binary network training. *ICLR*. 2, 6

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee. 3

Diffenderfer, J.; and Kailkhura, B. 2021. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. *ICLR*. 2, 3, 7

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the lottery: Making all tickets winners. *ICML*. 2

Frankle, J.; and Carbin, M. 2020. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*. 2, 3

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020. Pruning Neural Networks at Initialization: Why are We Missing the Mark? *CoRR*. 2

Han, K.; Wang, Y.; Xu, Y.; Xu, C.; Wu, E.; and Xu, C. 2020. Training binary neural networks through learning with noisy supervision. In *International Conference on Machine Learning*, 4017–4026. PMLR. 2

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149. 2

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*. 3

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778. 3, 6

Helwegen, K.; Widdicombe, J.; Geiger, L.; Liu, Z.; Cheng, K.-T.; and Nusselder, R. 2019. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in neural information processing systems*, 7533–7544. 1, 2

Huang, Q.; Zhou, K.; You, S.; and Neumann, U. 2018. Learning to prune filters in convolutional neural networks. In *WACV*. 2

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer. 3

Li, Z.; Ni, B.; Zhang, W.; Yang, X.; and Gao, W. 2017. Performance guaranteed network acceleration via high-order residual quantization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2584–2592. 2

Lin, J.; Rao, Y.; Lu, J.; and Zhou, J. 2017. Runtime neural pruning. In *NeurIPS*. 2

Lin, M.; Ji, R.; Xu, Z.; Zhang, B.; Wang, Y.; Wu, Y.; Huang, F.; and Lin, C.-W. 2020. Rotated binary neural network. *ECCV*. 6

Lin, X.; Zhao, C.; and Pan, W. 2017. Towards accurate binary convolutional neural network. In *NeurIPS*. 2, 6

Liu, Z.; Shen, Z.; Savvides, M.; and Cheng, K.-T. 2020. Reactnet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision*, 143–159. Springer. 2

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2019. Rethinking the value of network pruning. *ICLR*. 2

Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; and Cheng, K.-T. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*. 2, 3, 6

Loshchilov, I.; and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*. 3

Malach, E.; Yehudai, G.; Shalev-Shwartz, S.; and Shamir, O. 2020. Proving the Lottery Ticket Hypothesis: Pruning is All You Need. *CoRR*. 2

Martinez, B.; Yang, J.; Bulat, A.; and Tzimiropoulos, G. 2020. Training binary neural networks with real-to-binary convolutions. *arXiv preprint arXiv:2003.11535*. 7

Peters, J. W.; and Welling, M. 2018. Probabilistic binary neural networks. *CoRR*. 1

Qin, H.; Gong, R.; Liu, X.; Bai, X.; Song, J.; and Sebe, N. 2020a. Binary neural networks: A survey. *Pattern Recognition*. 1

Qin, H.; Gong, R.; Liu, X.; Shen, M.; Wei, Z.; Yu, F.; and Song, J. 2020b. Forward and Backward Information Retention for Accurate Binary Neural Networks. In *CVPR*. 1, 2, 3, 4, 5, 6, 7

Ramanujan, V.; Wortsman, M.; Kembhavi, A.; Farhadi, A.; and Rastegari, M. 2020. What's Hidden in a Randomly Weighted Neural Network? In *CVPR*. 3

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, 525–542. Springer. 1, 2, 3, 4, 6, 7

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*. 3, 6

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*, 2818–2826. 3, 6

Villani, C. 2003. *Topics in optimal transportation*. 58. American Mathematical Soc. 1, 3

Xiao, X.; Wang, Z.; and Rajasekaran, S. 2019. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *NeurIPS*. 2

Yang, Z.; Wang, Y.; Han, K.; Xu, C.; Xu, C.; Tao, D.; and Xu, C. 2020. Searching for Low-Bit Weights in Quantized Neural Networks. In *NeurIPS*. 2

Ye, M.; Gong, C.; Nie, L.; Zhou, D.; Klivans, A.; and Liu, Q. 2020. Good Subnetworks Provably Exist: Pruning via Greedy Forward Selection. *ICML*. 2

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 6848–6856. 3, 6

Zhou, H.; Lan, J.; Liu, R.; and Yosinski, J. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *NeurIPS*. 2

Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; and Zou, Y. 2016. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *CoRR*. 1