HYBRID WEIGHT REPRESENTATION: A QUANTIZATION METHOD REPRESENTED WITH TERNARY AND SPARSE-LARGE WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Previous ternarizations such as the trained ternary quantization (TTQ), which quantized weights to three values (e.g., $\{-W_n,\ 0,\ +W_p\}$), achieved the small model size and efficient inference process. However, the extreme limit on the number of quantization steps causes some degradation in accuracy. To solve this problem, we propose a hybrid weight representation (HWR) method which produces a network consisting of two types of weights, i.e., ternary weights (TW) and sparse-large weights (SLW). The TW is similar to the TTQ's and requires three states to be stored in memory with 2 bits. We utilize the one remaining state to indicate the SLW which is referred to as very rare and greater than TW. In HWR, we represent TW with values while SLW with indices of values. By encoding SLW, the networks can preserve their model size with improving their accuracy. To fully utilize HWR, we also introduce a centralized quantization (CQ) process with a weighted ridge (WR) regularizer. They aim to reduce the entropy of weight distributions by centralizing weights toward ternary values. Our comprehensive experiments show that HWR outperforms the state-of-the-art compressed models in terms of the trade-off between model size and accuracy. Our proposed representation increased the AlexNet performance on CIFAR-100 by 4.15% with only 1.13% increase in model size.

1 INTRODUCTION

Deep Neural Networks (DNN) has made considerable progress in various tasks such as image classification (LeCun et al. 1998, Simonyan & Zisserman 2014, Szegedy et al. 2015), object detection (Ren et al. 2015, Liu et al. 2016), and speech recognition (Graves et al. 2013, Amodei et al. 2016). However, outstanding neural networks usually require deeper and/or wider layers, thus making them hard to deploy on mobile and embedded devices. In response to this problem, many studies set their sights on more efficient networks. Various methods such as pruning (He et al. 2017), transformation (Howard et al. 2017) and quantization (Li et al. 2016, Zhu et al. 2016) have been carried out to reduce the model size and/or computation complexity effectively. Our paper concentrates on quantization that aims to represent values as the finite number of states in a low bit-width by discretizing.

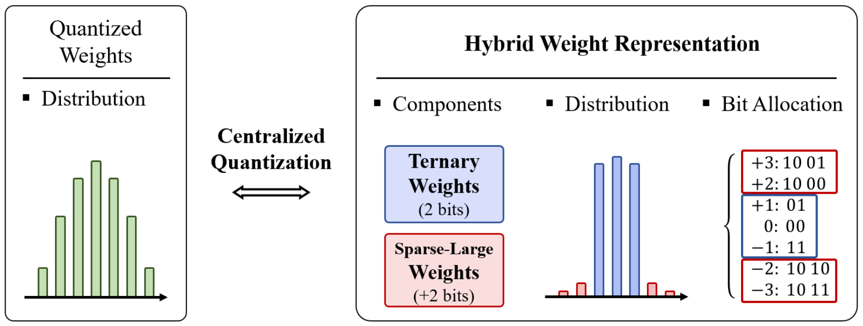


Figure 1: Comparison between conventional quantization and hybrid weight representation.

The accuracy degradation is resulted from quantizing values in a limited range with only 2bits. The ternary weights networks(TWN, Li et al. 2016) yields only three quantized values, which prohibits the networks from utilizing high weight values. As known in Han et al. 2015b, large-valued weights tend to have an important effect on the prediction. Therefore, the absence of large values can cause the accuracy degradation. To solve this problem, our paper proposes a hybrid weight representation (HWR), expressing networks with both ternary weights (TW) and sparse-large weights (SLW). By taking the advantages of both the TW and SLW, the proposed HWR method can reduce computation complexity and memory consumption as well as avoiding the accuracy degradation in networks.

To be specific, the large values of SLW help networks to improve their accuracy. Furthermore, SLW can be encoded with one remaining state which is not used to store TW in a 2 bits representation. It allows the networks to preserve their model size similarly to the trained ternary quantization (TTQ, Zhu et al. 2016). The compression rate of the encoding method is affected by the entropy of weight distributions. To train narrower distributions for the efficiency of HWR, we also introduce a centralized quantization (CQ) process and a weighted ridge (WR) regularizer. Figure 1 shows the differences between conventional quantization and HWR. As shown in Figure 1, there is only small number of SLW and the indices of encoded SLW are allocated in storage, unlike TW.

We conduct various experiments, showing that HWR obtains better classification accuracy with almost the same model size compared to the TTQ. The experiments are carried out on CIFAR-100 (Krizhevsky et al. 2009) and ImageNet (Russakovsky et al. 2015). We use AlexNet (Krizhevsky et al. 2012) and ResNet-18 (He et al. 2016) as baseline models. Our proposed representation increased the AlexNet performance on CIFAR-100 by 4.15% with only 1.13% increase in model size.

The contributions of this paper are as follows:

- We propose a representation method, namely HWR, including both values (TW) and indices of values (SLW). SLW allows the networks to improve their accuracy. Besides, the model size can be preserved by encoding SLW with one remaining state of TW.
- We also propose a training process, namely CQ, to improve the efficiency of HWR. By CQ, we can centralize most weights toward ternary values. The low entropy of centralization increases compression rate in encoding.
- In CQ, we introduce a new regularizer, namely WR, which gives penalty to large weights. WR is utilized to centralize weights for narrower distributions and categorize the weights into TW and SLW.

2 Related Work

Quantization In low-precision networks, full precision weights (w) are represented with the finite number of elements by a low bit-width. There is a perspective on the intervals between adjoining quantum for quantization.

The first way is linear quantization, meaning that the quantized weights are within the same intervals. For example, the binarized neural networks (BNN, Courbariaux & Bengio 2016) utilized a sign function to quantize weights and activations to {-1, +1}. The binarized weights w_b are defined as:

$$w_b = sign(w)$$

If inputs and weights are all binarized (Rastegari et al. 2016), the multiplications and additions of convolution layers can be compressed to XNOR and bit-count operations, respectively. Likewise, the ternary weight networks (TWN, Li et al. 2016) pruned the weights of BNN by thresholding (Δ).

$$w_t = \begin{cases} +W_E & if\ w > \Delta \\ 0 & if\ |w| \leq \Delta \\ -W_E & if\ w < -\Delta \end{cases}, \qquad \begin{array}{c} \Delta = 0.7 \cdot E(|w|) \\ W_E = \underset{i \in \{i | \Delta < |w(i)|\}}{E} (|w(i)|) \end{array}$$

DoReFa-Net (Zhou et al. 2016) carried out experiments in a wider bit-width and also quantize the gradients. The quantized weights w_q in DoReFa-Net are described as:

$$w_q = 2 \cdot Q_k(\frac{tanh(w)}{2max(|tanh(w)|)} + 0.5) - 1\ , \qquad Q_k(x) = \frac{round(x \cdot (2^k - 1))}{2^k - 1}$$

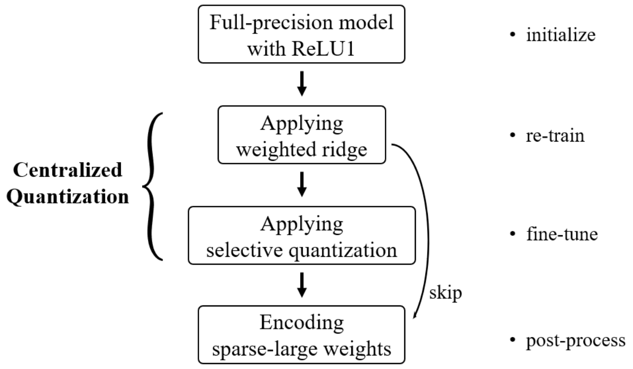


Figure 2: Overall processes of HWR. The weights are initialized from a full precision model trained with ReLU1. First of CQ, the weights are quantized with WR as a regularizer. By WR, the quantized weights are also centralized under threshold and categorized into TW and SLW by threshold. Second, selective quantization (SQ) is applied to fine-tune the previous weights without WR. Finally, SLW is encoded by the prefix that is one usable state of TW.

where $Q_k(\cdot)$ is a quantization function. These discretized weights (w_b, w_t, w_q) have derivatives that take zero-valued output in almost all input ranges. These zero-value outputs can cause the vanishing gradient problem. To solve this problem, the straight-through estimator (STE) method (Hinton et al. 2012, Bengio et al. 2013) is proposed where the derivative of w is replaced with the derivative of w_q instead of back-propagating zero gradients from discrete functions.

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial w_b} = \frac{\partial L}{\partial w_t} = \frac{\partial L}{\partial w_q}$$

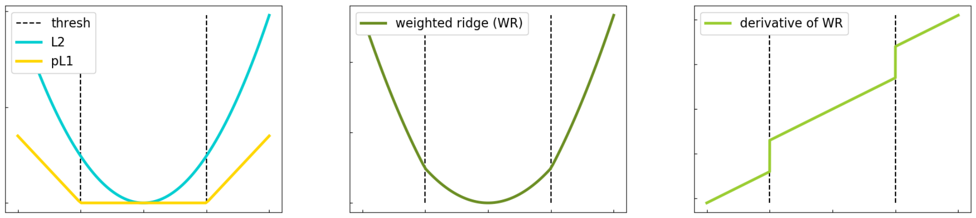
The same intervals make it possible to increase efficiency by replacing float operations to integer operations.

The second way is non-linear quantization where the weights are quantized with irregular intervals. For instance, the deep compression (Han et al. 2015a) quantized the weights through a clustering method and fine-tuned the representative values in each clustering group. Another example, TTQ (Zhu et al. 2016) quantized the weights to ternary values with two trainable scale coefficients for negative and positive weight values, i.e., $\{-W_n, 0, +W_p\}$. In these cases, the parameter sizes can be significantly reduced by replacing the weight values with the indices. However, the indices are needed to be transformed as weight values, and make it difficult to utilize more efficient integer base and/or bit-wise operations.

Entropy coding A Huffman coding (Van Leeuwen 1976) is a method to compress bit-streams in a lossless manner by reducing bit-length of each element with an optimal prefix tree. The entropy of weight distribution is one of the important factors that determine the compression rate in Huffman coding. Deep compression (Han et al. 2015a) pruned weights to make lower entropy and saved more storage by applying the Huffman coding.

Regularization Regularization is utilized to manipulate weights in artificial direction by imposing penalties. There are two regularizers such as Ridge (L2) and Lasso (L1) (Han et al. 2015b). The L2 weight decay usually prevents networks from being biased on the training dataset by restricting weights from growing. The penalty of Lasso makes the weights to be close to zero value which are unrelated to predict outputs during training-time. Moreover, the explicit loss of Zhou et al. 2018 makes it possible to quantize weights by controlling the strength of their regularizer. The regularization can be utilized to change the entropy of weight distribution with its penalty.

From previous studies, we focus on that ternary weights, linearly quantized, only require three states when to be stored in 2bits, and the remaining state has the potential to be utilized as a prefix of more extended weights for increasing efficiency in quantization. Furthermore, we also draw a deduction that regularization can help us to generate the lower entropy of the weight distribution, maximizing the compression efficiency in encoding the extended weights.



(a) L2 and pL1 (b) WR (c) the derivative of WR

Figure 3: Specifications of WR. (a) plots both a normal L2 and a part of L1 (pL1). WR is an addition of them as in (b). (c) is the derivative of WR.

3 Method

In this section, we explain: i) how CQ can centralize weights toward ternary values and categorize weights into TW and SLW; and ii) how the quantized weights are encoded to be expressed as HWR. The detailed processes are illustrated in Figure 2.

3.1 Basic Quantization Method

Our paper uses basic quantization (BQ) method, which start with a full precision model in which weights are pre-trained with ReLU1($clip(x,\ 0,\ 1)$ or $min(max(0,\ x),\ 1)$). Equation 1 shows how the weights (w) are quantized to w_q by a layer-wise quantization function ($Q_w(\cdot)$). The range (rng) of the w_q is determined by the maximum absolute value of the pre-trained weights ($max(|w_p|)$). the w is clipped to w_c by rng and entered into the $Q_w(\cdot)$. $Q_w(\cdot)$ also requires the total number of quantization steps (s) ascertained by the number of bits. In $Q_w(\cdot)$, the w_c is scaled by a float value to be discretized by a round function, then they are restored by the reciprocal to the float value. The derivative of w_c ($\frac{\partial L}{\partial w_c}$) is defined as $\frac{\partial L}{\partial w_q}$. To take a saturation effect as in BNN (Courbariaux & Bengio 2016), we restate w by clipping after updating gradients.

$$w_q = Q_w(w_c,\ rng,\ s) = round(w_c \cdot \frac{\frac{s-1}{2}}{rng}) \cdot \frac{rng}{\frac{s-1}{2}}, \qquad w_c = clip(w,\ -rng,\ rng) \tag{1}$$

In DoReFa-Net, ReLU1 was exploited to restrict the activated values. In our experiments, we select 4 bits to quantize the activations. The reason for using 4 bits as activation precision is that the 4 bits for weights and activations are enough to reproduce the accuracy of full precision models (Jung et al. 2019). The output (x) of ReLU1 can be quantized to a_q by a round function in Equation 2.

$$a_q = Q_a(x) = round[x(2^4 - 1)]\ /\ (2^4 - 1) \tag{2}$$

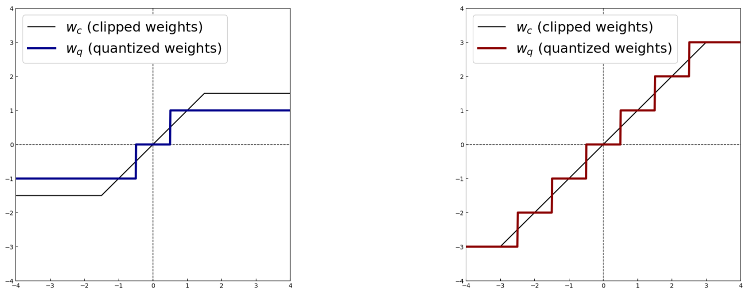
3.2 Weighted Ridge

We also introduce a new version of L2 weight decay, namely weighted ridge (WR), which aims to achieve two objectives: i) centralizing almost all weights which are below threshold; and ii) categorizing the weights into TW and SLW by threshold after re-training.

We quantize full precision weights by BQ method with WR as a regularizer. This step's quantizer is defined as:

$$w_q = Q_w(clip(w,\ -M_{w_p},\ M_{w_p}),\ \ M_{w_p},\ \ s_t + s_{sl}) \tag{3}$$

where the rng in Eq. 1 is fixed as the maximum absolute value of the pre-trained weights ($M_{w_p} = max(|w_p|)$). The s in Eq. 1 is derived from the number of bits of SLW (b_{sl}). If b_{sl} is 2, 3 or 4 bits, then the number of quantization steps of SLW (s_{sl}) is calculated as 4, 8 or 16 while the number of bits and quantization steps of TW (b_t, s_t) are fixed to 2 and 3.



(a) $mask(w) = TW$ (b) $mask(w) = SLW$

Figure 4: Two different quantization functions of SQ are applied for each category, TW and SLW.

Figure 3 (b) denotes WR which is a mixture of both, a normal L2 and a part of L1 (pL1) in (a). The loss of pL1 is only proportionate to the absolute values of the weights which are greater than threshold ($thresh$). In other words, the large weights receive more penalties compared to the weights under the $thresh$ in back-propagation as illustrated in Fig. 3 (c). This penalty allows us to sparsify weights that do not have the necessity for keeping largeness. WR, pL1 and $thresh$ are described as:

$$WR(w,\ \lambda_1,\ \lambda_2) = \lambda_1(L2(w) + \lambda_2 \cdot M_{w_p} \cdot pL1(w)) \tag{4}$$

$$pL1(w) = \begin{cases} |w| - thresh & if\ |w| > thresh \\ 0 & if\ |w| \leq thresh \end{cases},\quad (thresh = M_{w_p} \cdot \frac{s_t}{s_t + s_{sl} - 1}) \tag{5}$$

There are two hyper-parameters (λ_1, λ_2) which are tools to control the intensity of WR. By adjusting the greater λ_2, more weights can be sparsified below $thresh$. M_{w_p} is also one factor for applying different intensity to each layer since some layers have large difference of M_{w_p} from each other.

At the end of applying WR, we classify the re-trained weights into TW and SLW. $thresh$ is utilized as a criterion for labeling them as Equation 6.

$$mask(w) = \begin{cases} TW & if\ \ |w| \leq thresh \\ SLW & if\ \ |w| > thresh \end{cases} \tag{6}$$

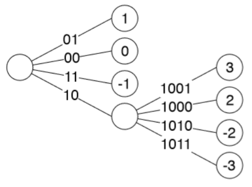
An observation is that WR tends to avoid overfitting as effectively as L2-regularizer. By adjusting WR, we can obtain better accuracy than L2 weight decay in some experiments.

3.3 Selective Quantization

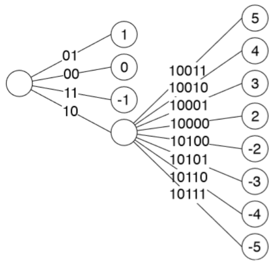
A selective quantization (SQ), the second step of CQ, aims to fine-tune the previous step's weights which are restricted by the additional penalty of pL1. By removing pL1, the restricted weights can grow again and be more optimized for classification. In SQ, two different quantization functions are applied for each category. As shown in Figure 4 (a), the weights in TW category are always ternarized and the percentage of TW is guaranteed during fine-tuning. Otherwise, the weights in SLW category can be quantized to both ternary values and large values by Fig. 4 (b). The quantizer of each category is based on Q_w so SQ can be expressed as Equation 7. Finally, the fine-tuned weights are re-categorized in the same way as Eq. 6. Additionally, if the result of SQ yields only slight difference compared to WR, this fine-tuning step can be skipped.

$$Q_s(w) = \begin{cases} Q_w(w,\ thresh,\ s_t + 1) & if\ mask(w) = TW \\ Q_w(w,\ M_{w_p},\ s_t + s_{sl}) & if\ mask(w) = SLW \end{cases} \tag{7}$$

Specifically, we use the stochastic gradient descent optimizer (SGD, Bottou 2010) for fine-tuning and the momentum optimizer (Sutskever et al. 2013) can misrepresent gradients since the weights are clipped after updating gradients.



(a) $\begin{array}{ll} b_t = 2, & b_{sl} = 2 \\ s_t = 3, & s_{sl} = 4 \end{array}$



(b) $\begin{array}{ll} b_t = 2, & b_{sl} = 3 \\ s_t = 3, & s_{sl} = 8 \end{array}$

Figure 5: Two examples for HWR encoding trees. In HWR, we represent TW with values while SLW with indices of values. The number of bits and quantization steps of TW (b_t, s_t) are fixed to 2 and 3. In contrast, them of SLW (b_{sl}, s_{sl}) can be changed to (2, 4), (3, 8) or (4, 16).

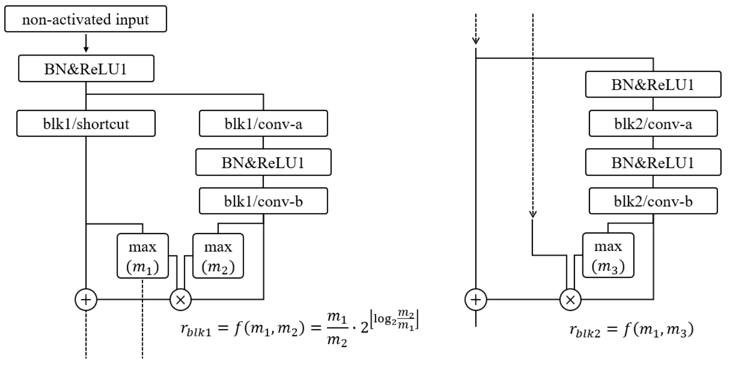


Figure 6: Before add operations in ResNet, the outputs of last convolution layers from each block are scaled by a scale factor r, derived from the maximum absolute values of convolution layers.

3.4 Encoding Sparse-Large Weights

The distributions of centralized-quantized models have low entropy since only small number of SLW. It allows achieving a higher compression rate with encoding method. Moreover, we utilize one remaining state of TW as a prefix to encode SLW as shown in Figure 5. As a result, HWR represents both values (TW) and indices (SLW) when they are allocated in memory. There are some advantages of using HWR. The model size can be maintained similar to TTQ since SLW is very rare. Furthermore, the restoration of SLW is not that difficult. The value of SLW can be restored by taking the first sign bit of b_{sl} and adding 2 to the rest of b_{sl}. The remaining TW still has the potential to keep some benefits of ternary weights such as replacing multiplication operations to sign assignment operations (Rastegari et al. 2016) and utilizing gates for skipping zero weights (Deng et al. 2018).

3.5 The mediation of quantum intervals in add operations

In our quantization method, the weights and activations are quantized in the same intervals ($w_q = i_w \cdot I_w$, $a_q = i_a \cdot I_a$). The I_w and I_a are fixed float intervals while the i_w and i_a are integer variables. Therefore, convolution and fully-connected layers can be inferred by integer operations as Equation 8. Other operations such as batch normalization (BN, Ioffe & Szegedy 2015), ReLU1 and the $Q_a(\cdot)$ also can be integrated and compressed by integer comparators (Lahoud et al. 2019) since the coefficients of BN and boundaries of ReLU1 and $Q_a(\cdot)$ are fixed so can be prearranged.

$$o_q = layer(a_q,\ w_q) = \sum\sum a_q w_q = I_a I_w \cdot \sum\sum i_a i_w \tag{8}$$

However, ResNet (He et al. 2016) has identity mappings and add operations that prevent the data streams from keeping the same quantum intervals due to the layer-wise intervals. To deal with this

Table 1: The results of WR in full precision models with ReLU activation.

Model Name	**Quantizaton Plan** (b_t/b_{sl})	**Activation function**	λ_1	λ_2	**Top-1 Accuracy**
AlexNet	-	ReLU	0.02	-	75.06%
AlexNet	2 / 3	ReLU	0.02	1.2	**75.32%**
ResNet-18	-	ReLU	0.0002	-	76.07%
ResNet-18	2 / 2	ReLU	0.0002	1	**76.49%**
ResNet-18	2 / 3	ReLU	0.0002	1	**76.28%**
ResNet-18	2 / 4	ReLU	0.0002	1	**76.88%**

Table 2: The results of applying the scaling factor r on ResNet-18 in full precision.

Model Name	**Scaling**	**Activation function**	λ_1	**Top-1 Accuracy**
ResNet-18	-	ReLU	0.0002	76.07%
ResNet-18	$\frac{m_{short}}{m_{conv_b}}$	ReLU	0.0002	75.80%
ResNet-18	r	ReLU	0.0002	**76.15%**
ResNet-18	-	ReLU1	0.0002	74.54%
ResNet-18	$\frac{m_{short}}{m_{conv_b}}$	ReLU1	0.0002	74.16%
ResNet-18	r	ReLU1	0.0002	**74.54%**

problem, we scaled one input of add operations by a scale factor r as defined in Equation 9. The r is derived from the maximum absolute values of the shortcut (m_{short}) and last convolution (m_{conv_b}) layers from each block as in Figure 6. Since we use a log and floor function, r is always in the range (0.5, 1] and it helps to keep the effect of identity mapping of ResNet. If r is zero, then nothing is added in the add operation or if r is larger than 1, the shortcut can be perturbed by a large residual.

$$r = f(m_{short}, m_{conv_b}) = \frac{m_{short}}{m_{conv_b}} \cdot 2^{\left\lfloor \log_2 \frac{m_{conv_b}}{m_{short}} \right\rfloor}, \qquad (0.5 < r \leq 1) \tag{9}$$

By adjusting r, two different intervals from jth ($shortcut$) and kth ($conv_b$) convolution layers can be mediated and it will allow us to add two inputs with only shift operations as:

$$add(o_q^j, o_q^k) = m_{short} \cdot (2^{-\left\lfloor \log_2 \frac{I_a^k I_w^k}{I_a^j I_w^j} \right\rfloor} \cdot \sum\sum i_a^j i_w^j + \sum\sum i_a^k i_w^k) \tag{10}$$

4 Experiment

4.1 Full precision models

We utilize AlexNet (Krizhevsky et al. 2012) and ResNet-18 (He et al. 2016) as baseline models, implemented on Tensorflow (Abadi et al. 2016) framework. In AlexNet, we use a variant of it by adding batch normalization (Ioffe & Szegedy 2015) layers instead of dropout layers and set the learning rate to 0.01 with decay 0.1 after every 100 epochs until 300 epochs. In ResNet, we set the learning rate to 0.1 with the same schedule. To estimate the performance, we select CIFAR-100 Krizhevsky et al. 2009) and set the minibatch size as 128. We also use Nesterov Accelerated Gradients (Sutskever et al. 2013) as an optimizer. To prevent the fluctuation of validation accuracy, we take simple moving averages in 5 epochs during training-time and average it over four repetitions.

Table 3: The results of the top-1 accuracy for CQ and the average bit length for HWR in multiple bit-width. The complete experiments are in Appendices A.

Model Name	Bit-W (b_t/b_{sl})	Bit-A	λ_2	Percentage of SLW	Average Bit Length	Top-1 Error	Remarks
AlexNet	32	32	-	-	-	74.78%	FP-ReLU1
AlexNet	2 / -	4	-	0%	2	70.57%	TTQ
AlexNet	2 / 3	4	-	12.05%	2.362	73.54%	BQ
AlexNet	2 / 3	4	1.2	1.39%	2.042	73.39%	WR
AlexNet	2 / 3	4	-	0.83%	2.025	**74.72%**	SQ
ResNet-18	32	32	-	-	-	74.54%	FP-ReLU1
ResNet-18	2 / -	4	-	0%	2	72.79%	TTQ
ResNet-18	2 / 2	4	-	0.59%	2.0118	73.92%	BQ
ResNet-18	2 / 2	4	1.0	0.014%	2.0003	73.92%	WR
ResNet-18	2 / 2	4	-	0.0015%	2.00003	**74.22%**	SQ
ResNet-18	2 / 3	4	-	7.34%	2.2203	74.07%	BQ
ResNet-18	2 / 3	4	1.0	0.434%	2.013	74.05%	WR
ResNet-18	2 / 3	4	-	0.321%	2.0096	**74.67%**	SQ
ResNet-18	2 / 4	4	-	28.80%	3.1521	74.54%	BQ
ResNet-18	2 / 4	4	1.0	4.995%	2.2	74.32%	WR
ResNet-18	2 / 4	4	-	4.145%	2.166	**74.84%**	SQ

We perform experiments on full precision models with WR as a regularizer as shown in Table 1. We adjust λ_2 to 1 which is sufficiently large because the penalty M_{w_p} is imposed on the weights over threshold in Fig. 3 (c). The results of AlexNet and ResNet-18 with WR obtain 0.26% and 0.81% higher accuracy than state-of-the-art models. The additional penalty of pL1 also prevents the networks from overfitting as effectively as L2 regularizer. We conjecture that the large weights also can be biased toward training data. Thus, the restriction for large weights enables us to make more generalized models.

In Table 2, the experiments are conducted to observe the effects of r scaling on the accuracy of ResNet-18. We compare three coefficients such as zero, r (Eq. 9) and m_{short}/m_{conv_b}. We observe that r performs as similar as non-scaled models. As mentioned in Section 3.5, the m_{short}/m_{conv_b}, with a value greater than 1, can perturb the shortcut connections. We apply r scaling for all following quantization experiments on ResNet-18.

4.2 Centralized quantization and hybrid weight representation

In this section, we compare the methods such as TTQ, BQ, WR and SQ. In quantization, we quantize weights with multiple bit-width and activations in 4 bits precision. The activations of TTQ are also quantized in the same bits. We keep the first and last layers as full precision. We show our evaluation results in Table 3.

To observe the effect of the fixed range of our BQ method, we perform a particular experiment. There are only two different conditions compared to TTQ: i) fixing the range of the weights during initialization; and ii) using one coefficient as $\{-W_s, 0, +W_s\}$. By applying them, the accuracy of AlexNet and ResNet-18 are rather improved by 1.47% and 0.76%. We believe that ReLU1 activation can make it difficult to optimize the range of weights by training in TTQ.

The rows remarked as BQ denotes the results of basic quantization in multiple bit-width. As shown in Table 3, BQ tends to perform better with the cost of additional bits. The BQ and TTQ are used as baselines of quantization. Additionally, TTQ set a constant t to 0.05 for quantizing weights under $t \cdot max(|w|)$ as zero values. In our method, however, if b_{sl} is 2, 3 or 4 bits then the t is derived as

Table 4: The results of HWR and CQ on ImageNet dataset.

Model Name	**Bit-W** (b_t/b_{sl})	**Bit-A**	λ_2	**Percentage of SLW**	**Average Bit Length**	**Top-1 Error**	**Remarks**
ResNet-18	32	32	-	-	-	70.41%	FP
ResNet-18	32	32	-	-	-	62.51%	FP-ReLU1
ResNet-18	2 / 3	4	-	1.11%	2.033	56.12%	BQ
ResNet-18	2 / 3	4	0.25	0.84%	2.025	55.73%	WR
ResNet-18	2 / 3	4	-	0.54%	2.016	**58.34%**	SQ

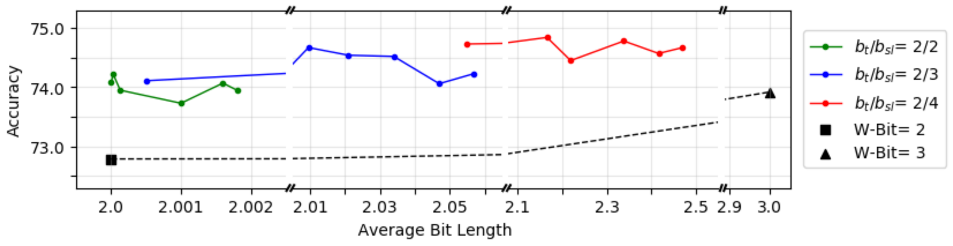


Figure 7: A trade-off between accuracy and model size in various $\lambda_2(2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16})$.

$\frac{1}{6}$, $\frac{1}{10}$ or $\frac{1}{18}$. All our t are higher than 0.05 and it can cause worse results since L2 regularizer gives more penalty to larger weights.

Other rows commented as WR, the weights which are over threshold are restricted by WR so the model has a lower percentage of SLW than BQ after this step. The top-1 accuracy of WR tends to show worse results than BQ, as dissimilar to Table 1. Thus, we deduce that the quantization can weaken the generalization effect of the additional penalty of pL1.

The other rows tagged as SQ are the results of the last fine-tuning step, which also imply the results of CQ. We use the same learning rate schedule from previous steps and use SGD (Bottou 2010) optimizer. By removing pL1, the restricted weights can grow during fine-tuning. As a result, we obtain better accuracy and compression rate than WR. Furthermore, after there encoding step, the difference between the average bit length of SQ and TTQ becomes almost same. When b_{sl} is 3 in AlexNet, SQ reaches top 1 accuracy of 74.72%, which is 4.15% higher than TTQ with a 1.25% increase in model size. In ResNet-18, if b_{sl} is 3 then the classification accuracy of SQ is 1.88% higher while the model size increases only 0.5% compared to TTQ.

4.3 ImageNet

Table 4 shows the results of ResNet-18 on ImageNet (Russakovsky et al. 2015) dataset. We set the minibatch size as 256 and the learning rate as 0.1 with decay by 0.1 at 25, 50, 75 and 80 epochs until 85 epochs. We take the best validation error during training. Compared to the full precision model, using ReLU1 activation causes large degradation in accuracy. The results of BQ, WR and SQ shows a similar tendency to Table 3. SQ reaches top1 accuracy of 58.34%, which is 2.22% higher than BQ, while reducing the average bit length. Compared to CIFAR-100, the average bit length of ImageNet is lower in the same condition. We can make an assumption that if the fixed range (M_{w_p}) is large, then clipping of the range can be required for more optimized quantization.

5 Discussion

5.1 Trade off between accuracy and model size

We perform extended experiments in various λ_2 to monitor the correlation between the percentage of SLW and the accuracy of ResNet-18 on CIFAR-100. The complete results are listed in Appendices

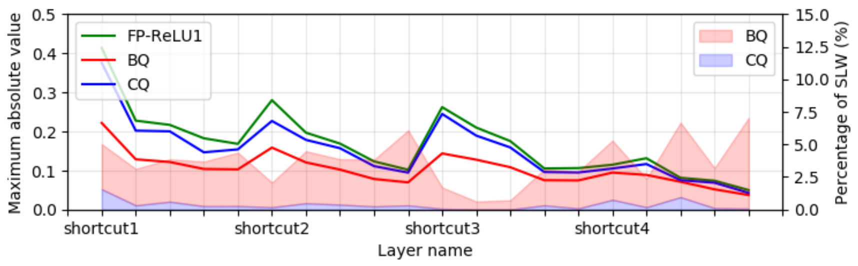


Figure 8: Layer-wise comparison of the percentage of SLW and the maximum absolute value.

A. Figure 7 shows that the intensity of pL1 (λ_2) and classification accuracy are less related. In spite of the greater λ_2, some results show even better accuracy than the results of the lower λ_2. Therefore we can perceive that if the number of parameters in networks is already enough then most of the large weights can be reduced with comparable accuracy, accompanied of generalization.

5.2 Analysis of quantization method.

In Table 3, we observe that there is a difference of the percentage of SLW between WR and SQ. Some results of SQ shows two times lower percentage than WR. The forcibly restricted weights from WR can grow in SQ step. Thus, we assume that when a weight grow and begin to contribute more importantly for prediction then some other weights, which make a similar contribution to the growing weight, become smaller during fine-tuning. Appendices B.1 illustrates a difference of the weights distributions between BQ and SQ. The -2 and +2 weight values of WR are wide-spread after SQ step. To make more flatten distributions from BQ, we suggest using another version of WR which utilize a part of exponential lasso (Breheny 2015) instead of pL1 as shown in Appendices B.2.

Figure 8 displays the layer-wise maximum absolute value and percentage of SLW. The graph denotes three: i) If a layer has many weights in SLW after BQ, then the layer still has a relatively higher number of SLW after SQ; and ii) The maximum absolute values of CQ tend to be larger than BQ since they have only small number of large weights; and iii) The maximum absolute values of BQ are remarkably lower than the full precision model. We can infer that if the fixed range is too large, then the L2 regularizer can prevent the quantized weights from growing. For this reason, clipping the maximum absolute values can be a method for effective quantization.

6 Conclusion

We propose a hybrid weight representation (HWR), consisting of two types of weights such as ternary weights (TW) and sparse-large weights (SLW). In HWR, we represent TW with values while SLW with indices of values. To represent SLW as indices, we encode SLW by utilizing one usable state as a prefix, which is not used to store TW in 2 bits. To maximize the effect of encoding, we use a variant of L2 regularizer, namely weights ridge (WR), for making low entropy from narrower distributions. We also propose a centralized quantization (CQ) training process, including both applying WR and fine-tuning by a selective quantization (SQ) without WR. In other words, SLW helps the networks to improve their accuracy while preserving their model size by encoding SLW compared to a 2-bit representation. Additionally, we propose a scaling factor r to mediate two different quantum intervals of add operations in ResNet. We increase the ResNet-18 performance on CIFAR-100 by 1.88% with only 0.5% increase in model size and on ImageNet by 2.22% with only 0.8% increase in model size. As a result, we can achieve more efficient networks in terms of the trade-off between model size and accuracy.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Patrick Breheny. The group exponential lasso for bi-level variable selection. *Biometrics*, 71(3): 731–740, 2015.

Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. URL `http://arxiv.org/abs/1602.02830`.

Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks*, 100:49–58, 2018.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14:8, 2012.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Fayez Lahoud, Radhakrishna Achanta, Pablo Márquez-Neila, and Sabine Süsstrunk. Self-binarizing networks. *arXiv preprint arXiv:1902.00730*, 2019.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Jan Van Leeuwen. On the construction of huffman trees. In *ICALP*, pp. 382–410, 1976.

Aojun Zhou, Anbang Yao, Kuan Wang, and Yurong Chen. Explicit loss-error-aware quantization for low-bit deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9426–9435, 2018.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

A FULL EXPERIMENTS

B GRAPHS

B.1 A CHANGE IN WEIGHT DISTRIBUTION OF EACH PROCESS

B.2 A VERSION OF WR WITH EXPONENTIAL LASSO

Table 5: The experimental results of HWR and CQ in multiple bit-width and various λ_2.

Model Name	Bit-W (b_t/b_{sl})	Bit-A	λ_2	Percentage of SLW	Average Bit Length	Top-1 Error	Remarks
AlexNet	32	32	-	-	-	74.78%	FP-ReLU1
AlexNet	2 / -	4	-	0%	2	70.57%	TTQ
AlexNet	2 / 3	4	-	12.05%	2.362	73.54%	BQ
AlexNet	2 / 3	4	1.2	1.39%	2.042	73.39%	WR
AlexNet	2 / 3	4	-	0.83%	2.025	**74.72%**	SQ
ResNet-18	32	32	-	-	-	74.54%	FP-ReLU1
ResNet-18	2 / -	4	-	0%	2	72.79%	TTQ
ResNet-18	2 / 2	4	-	0.59%	2.0118	73.92%	BQ
ResNet-18	2 / 2	4	0.0625	0.373%	2.0075	73.57%	WR (1)
ResNet-18	2 / 2	4	-	0.092%	2.0018	**73.95%**	SQ (1)
ResNet-18	2 / 2	4	0.125	0.198%	2.004	73.93%	WR (2)
ResNet-18	2 / 2	4	-	0.079%	2.0016	**74.07%**	SQ (2)
ResNet-18	2 / 2	4	0.25	0.095%	2.002	73.73%	WR (3)
ResNet-18	2 / 2	4	-	0.049%	2.001	**73.73%**	SQ (3)
ResNet-18	2 / 2	4	0.5	0.023%	2.0004	74.11%	WR (4)
ResNet-18	2 / 2	4	-	0.0068%	2.00013	**73.95%**	SQ (4)
ResNet-18	2 / 2	4	1.0	0.014%	2.0003	73.92%	WR (5)
ResNet-18	2 / 2	4	-	0.0015%	2.00003	**74.22%**	SQ (5)
ResNet-18	2 / 2	4	2.0	0.0002%	2.000004	73.93%	WR (6)
ResNet-18	2 / 2	4	-	0.0004%	2.0+8e-7	**74.08%**	SQ (6)
ResNet-18	2 / 3	4	-	7.34%	2.2203	74.07%	BQ
ResNet-18	2 / 3	4	0.0625	4.955%	2.149	74.19%	WR (1)
ResNet-18	2 / 3	4	-	1.904%	2.057	**74.23%**	SQ (1)
ResNet-18	2 / 3	4	0.125	3.393%	2.102	74.05%	WR (2)
ResNet-18	2 / 3	4	-	1.563%	2.047	**74.06%**	SQ (2)
ResNet-18	2 / 3	4	0.25	2.11%	2.063	74.28%	WR (3)
ResNet-18	2 / 3	4	-	1.147%	2.034	**74.52%**	SQ (3)
ResNet-18	2 / 3	4	0.5	0.967%	2.029	74.1%	WR (4)
ResNet-18	2 / 3	4	-	0.7%	2.021	**74.54%**	SQ (4)
ResNet-18	2 / 3	4	1.0	0.434%	2.013	74.05%	WR (5)
ResNet-18	2 / 3	4	-	0.321%	2.0096	**74.67%**	SQ (5)
ResNet-18	2 / 3	4	2.0	0.018%	2.0006	74.3%	WR (6)
ResNet-18	2 / 3	4	-	0.016%	2.0005	**74.11%**	SQ (6)
ResNet-18	2 / 4	4	-	28.80%	3.1521	74.54%	BQ
ResNet-18	2 / 4	4	0.0625	20.67%	2.827	74.56%	WR (1)
ResNet-18	2 / 4	4	-	11.749%	2.47	**74.67%**	SQ (1)
ResNet-18	2 / 4	4	0.125	16.608%	2.664	74.28%	WR (2)
ResNet-18	2 / 4	4	-	10.395%	2.416	**74.57%**	SQ (2)
ResNet-18	2 / 4	4	0.25	12.38%	2.495	74.53%	WR (3)
ResNet-18	2 / 4	4	-	8.462%	2.338	**74.78%**	SQ (3)
ResNet-18	2 / 4	4	0.5	8.108%	2.324	74.26%	WR (4)
ResNet-18	2 / 4	4	-	5.485%	2.219	**74.45%**	SQ (4)
ResNet-18	2 / 4	4	1.0	4.995%	2.2	74.32%	WR (5)
ResNet-18	2 / 4	4	-	4.145%	2.166	**74.84%**	SQ (5)
ResNet-18	2 / 4	4	2.0	1.611%	2.064	74.06%	WR (6)
ResNet-18	2 / 4	4	-	1.381%	2.055	**74.73%**	SQ (6)

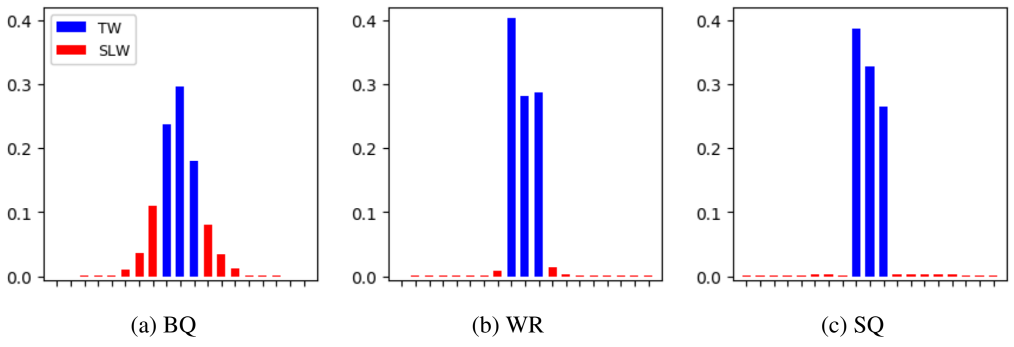


Figure 9: The weights distributions of each step.

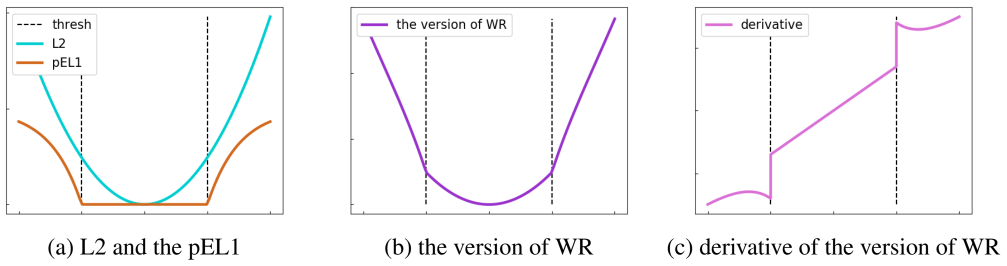


Figure 10: Another version of WR, using the part of elastic lasso (pEL1) instead of pL1. (a) denotes both L2 and pEL1. The version of WR is addition of them as in (b). (c) is the derivative of the version of WR.