Flexible Network Binarization with Layer-wise Priority

Lixue Zhuang*, Yi Xu*, Bingbing Ni*, Hongteng Xu†
Shanghai Jiao Tong University*, Duke University†
{qingliang, xuyi, nibingbing}@sjtu.edu.cn
hongteng.xu@duke.edu

Abstract
How to effectively approximate real-valued parameters with binary codes plays a central role in neural network binarization. In this work, we reveal an important fact that binarizing different layers has a widely-varied effect on the compression ratio of network and the loss of performance. Based on this fact, we propose a novel and flexible neural network binarization method by introducing the concept of layer-wise priority which binarizes parameters in inverse order of their layer depth. In each training step, our method selects a specific network layer, minimizes the discrepancy between the original real-valued weights and its binary approximations, and fine-tunes the whole network accordingly. During the iteration of the above process, it is significant that we can flexibly decide whether to binarize the remaining floating layers or not and explore a trade-off between the loss of performance and the compression ratio of model. The resulting binary network is applied for efficient pedestrian detection. Extensive experimental results on several benchmarks show that under the same compression ratio, our method achieves much lower miss rate and faster detection speed than the state-of-the-art neural network binarization method.

Introduction
Deep learning methods have shown excellent performance in many domains, especially in the field of computer vision. Actually, in almost all visual tasks, the state-of-the-art methods are all based on deep neural networks. However, these deep learning-based methods heavily rely on devices with high computational power and large memory, which limits their applications seriously. For example, the AlexNet model (Krizhevsky, Sutskever, and Hinton 2012) that won the ImageNet competition in 2012 is over 200 MB with 61 million parameters and requires about one billion floating-point operations per image. The VGG-16 model (Simonyan and Zisserman 2014) is over 500 MB and needs about 40 billion floating-point operations per image. The growing depth and size of deep neural network brings a great challenge for the deployment on mobile and embedded devices, such as cell phones and FPGAs. Therefore, network compression has become an urgent and important research topic.

Currently, a group of network compression methods (Courbariaux, Bengio, and David 2015) Courbariaux et al. 2016) are advanced to binarize real-valued networks and attract a lot of researchers. In particular, they utilize different iterative thresholding strategies to approximate the weights of real-valued networks with ±1 or ±1 according to their signs. With the help of these neural network binarization methods, we can greatly speed up the forward network computation and reduce network storage at least by a factor of 32 (when the original weights are single-precision floating-point numbers). However, most of these methods binarize all parameters in networks simultaneously. Such a strategy suffers from two problems. Firstly, the approximation errors caused by binarization are dramatically aggregated through all layers and thus difficult to converge. The performance of a binarized network is highly dependent on the discrepancy between the binary codes and the corresponding real-valued weights, generally resulting in serious degradations, such as the loss of accuracy in classification tasks or the increase of miss rate in detection tasks. Secondly, binarizing all parameters limits the flexibility of these methods on exploring a trade-off between the loss of performance and the compression ratio of network.

To explicitly address the above problems, we propose a novel and flexible layer-wise network binarization (LWB) method. Specifically, we find that 1) the effects of parameter binarization are widely varied across different network layers in terms of the loss of performance and the compression ratio of the network model, and 2) generally the binarization of deeper layers in a network leads to high compression ratio with little impact on performance. Based on these two facts, we propose a novel and flexible neural network binarization method by introducing the concept of layer-wise priority which binarizes parameters in inverse order of their layer depth. In each training step, our method selects a specific network layer, minimizes the discrepancy between the original real-valued weight and its binary approximations, and fine-tunes the whole network accordingly. During the binarization process, it is significant that we can flexibly determine whether to continue binarizing the remaining floating-point layers and achieve the trade-off between the loss of performance and the compression ratio of network.

The scheme of our method is illustrated in Fig. 1

Our method is validated in the task of pedestrian detection, in which case both detection accuracy and computation efficiency are highly demanded. Experimental results
show that the proposed LWB method is effective and flexible for network binarization, which converges more quickly and achieves much lower miss rate under the same compression ratio compared with its competitors.

Related work

Network Pruning: Network pruning has proven to be an effective way to reduce the network size by removing the non-informative parameters. Optimal Brain Damage (Le Cun, Denker, and Solla 1989) and Optimal Brain Surgeon (Hassibi and Stork 1993) prune connections between neurons based on the Hessian of the loss function. As for deep neural networks, these methods are computationally intensive in terms of computation of the second-order derivatives. Later, Ciresan et al. (Ciresan et al. 2011) propose to drop the weights randomly and achieve good performance. Han et al. (Han et al. 2015) put forward a new method called Deep Compression, which combines network pruning, parameter quantization and Huffman coding together and reduce the model size of AlexNet by $35 \times$ and VGG-16 by $49 \times$ without sacrificing accuracy. Additionally, network compression can also be achieved by reusing predefined or learned filters for different layers, e.g., the scatter transformation network (Bruna and Mallat 2011) and the fractal-based CNN (Xu et al. 2016).

Different from the methods mentioned above, in this work we focus on the flexibility problem of neural network binarization and aim to explore a trade-off between the loss of performance and the compression ratio of model.

Motivation

Binary-Weight-Networks Revisited

Denote an $L$-layer CNN as $W = \{W_l\}_{l=1,..,L}$, where $W_l = \{W_{lk} \in \mathbb{R}^{C_{x_l} \times w_l \times h_l}\}_{k=1,..,K_l}$ represents the set of filters in the $l^{th}$ layer and $(c, w, h)$ denotes the number of channels, width and height of each filter. And $K_l$ represents the number of filters in $l^{th}$ layer. Similarly, for each layer, the input tensor can be represented as $I_l$, $l = 1, .., L$. The BWN method estimates each float-point filter $W_{lk}$ with a real-valued scale factor $\alpha_{lk} \in \mathbb{R}^+$ and a binary filter $B_{lk} \in \{+1, -1\}^{C_{x_l} \times w_l \times h_l}$, i.e., $W_{lk} \approx \alpha_{lk} B_{lk}$. As a result, the original convolution operation can be approximated by

$$I_l * W_{lk} \approx (I_l \oplus B_{lk}) \alpha_{lk},$$

where the symbol $*$ represents traditional convolution operation while $\oplus$ indicates the convolution operation only involving additions and subtractions as the weights of the filter are binary. The optimal binarization is achieved by solving the following optimization problem:

$$\alpha_{lk}^*, B_{lk}^* = \arg \min_{\alpha_{lk}, B_{lk}} \|\text{vec}(W_{lk}) - \alpha_{lk} \text{vec}(B_{lk})\|_2^2,$$
where \( \text{vec}(\cdot) \) vectorizes its input and \( \| \cdot \|_2 \) represents the \( \ell_2 \) norm. The solution of (2) is
\[
\alpha^*_{jk} = \frac{1}{n} \| \text{vec}(W_{jk}) \|_1, \quad B^*_{jk} = \text{sign}(W_{jk}),
\]
where \( n = c \times w \times h \), \( \| \cdot \|_1 \) represents the \( \ell_1 \) norm, and \( \text{sign}(\cdot) \) returns 1 for nonnegative element and \(-1\) otherwise.

All of the parameters in the network are binarized simultaneously. In particular, alternating optimization between real-valued network and binary network is conducted based on a variant of stochastic gradient descent (SGD). In each iteration (or batch), the forward propagation of activations and the backward propagation of gradients are calculated based on current binary weights (i.e. the solution of (2) with the real-valued weights given in previous iteration), then the real-valued weights are updated by gradient descent. A more detailed description of the algorithm can be found in \cite{Rastegari et al. 2016}. It should be noted that the BWN method and our LWB algorithm proposed in the following section are not only suitable for CNNs. They can also be extended directly to other neural network architectures.

**Observable Layer-wise Priority**

As aforementioned, BWN (and other network binarization methods) suffers nontrivial loss of performance when the target neural network is deep. One reason for this phenomenon is that it ignores the difference between layers in the network and binarizes them in a unified manner. It is confirmed from the observation in the following analytic experiment. Take the YOLOv2 network \cite{Redmon and Farhadi 2016} for pedestrian detection as an example. We binarize different layers using the BWN method\footnote{All of the networks are trained and tested on Caltech \cite{Dollár et al. 2009} and INRIA \cite{Dalal and Triggs 2005} datasets. The initial parameters of these networks are obtained from the pre-trained real-valued model with 23.6\% and 11.4\% miss rate, respectively.} and show the variation of the miss rate with respect to different binarized layers in Table 1. We can find that binarizing the first few layers causes significant loss of accuracy about 1\% while binarizing the latter few layers has little effect. These observations reflect that 1) the contributions of different layers to network compression are inconsistent because the numbers of their weights are different — the latter layers generally contain more parameters due to the increment of channel numbers; 2) the loss of performance is widely varied when we binarize different layers — generally binarizing the latter layers has little negative effect on the performance.

To achieve a trade-off between the loss of performance and the compression ratio of model, therefore we binarize network layers with a priority strategy. Moreover, the proposed layer-wise priority is highly correlated with the depth of layer. In many cases, the latter layers should be assigned with higher priority because its binarization may lead to high compression ratio with ignorable loss of performance, and the binarization can be stopped at certain layer when the loss of performance is intolerable. Based on the observable layer-wise priority, we propose a novel and flexible network binarization method in the following section, which binarizes parameters in inverse order of their layer depth.

**Our approach**

**Layer-wise Network Binarization**

According to the above analysis, we propose a flexible network binarization method with a layer-wise priority strategy. Given an \( L \)-layer network, the priority of a layer is generally varied inversely with respect to its depth. In each binarization step, we select the layer with the highest priority of binarization in the current step to implement our binarization algorithm, minimizing the discrepancy between the original real-valued parameters and its binary approximations. In the following content, we therefore assume that the last \( i \) — 1 layers of the target network have been binarized before applying the \( i^{th} \) binarization step. It should be noted that even if the layer-wise priority is defined based on other metrics, rather than the depth of layer, our LWB algorithm is still feasible.

Denote the initial neural network in the \( i^{th} \) binarization step as \( \mathcal{W}^{(i)} = \{ \{ W^{(i)}_{jk} \}_{j=L-i+1}^{L}, \{ \alpha^{(i)}_{jk} B^{(i)}_{jk} \}_{j=L-i+1}^{L} \} \), where \( W^{(i)}_{jk} = \{ W^{(i)}_{jk} \}_{k=1}^{K_{j}} \) represents the set of real-valued parameters in the \( j^{th} \) layer and \( \alpha^{(i)}_{jk} B^{(i)}_{jk} = \{ \alpha^{(i)}_{jk} B^{(i)}_{jk} \}_{k=1}^{K_{j}} \) represents the set of binarized parameters in the \( j^{th} \) layer learned in previous binarization steps. Our LWB method further binarizes the parameters in the \( (L - i + 1)^{th} \) layer (the layer with the highest priority), i.e., \( \mathcal{W}^{L-i+1}_{L} \), and keeps the performance of the network by minimizing the loss function that is used to train the original network:
\[
\mathcal{W}^{(i+1)} = \underset{\mathcal{W}}{\arg \min} \text{loss}(\mathcal{W}; \mathcal{I}_{1}),
\]
\[s.t. \ W_{l} \text{ is binarized for } l = L - i + 1, \ldots, L,
\]
where \( \mathcal{I}_{1} \) is the training data, i.e., the input tensor of the 1\textsuperscript{st} layer, and the initial point of the problem is \( \mathcal{W}^{(i)} \).

To compress the network and keep the performance at the same time, we need to binarize the real-valued parameters in the layer and fine-tune the whole network as well. In particular, our LWB method can be viewed as a variant of SGD combined with BWN, which updates real-valued parameters and calculates their binary approximation alternatively. In each iteration of LWB, the forward propagation of activation and the backward propagation of gradients are calculated. For those binarized layers, both propagation can be accelerated because their operations merely involve additions and subtractions. Then, the real-valued parameters and binarized parameters are both updated using traditional gradient-
based methods (e.g., SGD, Adam, etc). Finally, we binarize the last $i$ layers by solving (2), the solution of which will be used in next iteration. Repeating the above iteration till the loss function converges, we can obtain a network with $i$ binarized layers.

**Flexible Binarization**

In practical applications, one challenging problem of network binarization is when to stop the binarization process. As aforementioned, binarizing the whole network (as BWN and other methods did) always cause obvious loss of performance. Moreover, these frameworks make their binarization processes uncontrollable. As compared with the existing network binarization methods, an advantage of our LWB method is its flexibility — we can define an explicit criterion and then stop our binarization process at a certain layer once the criterion is violated.

In this work, we apply a straightforward criterion to our LWB method and achieve a flexible network binarization result. Specifically, in each binarization step, we preserve the initial binarized network (learned in the previous step). After obtaining the new binarized network, we can go to the next binarization step if the loss of the learned network over the validation set is lower than a predefined threshold. Otherwise, we stop the whole binarization procedure.

Like the metric of priority, we can develop more sophisticated criteria or metrics to achieve a trade-off between the compression ratio of network and the loss of performance, e.g., the ratio of compression ratio to validation set’s loss. Fortunately, experimental results in the following section will show that applying the simple criterion mentioned above can help us to enhance the flexibility of our network binarization indeed. In summary, the scheme of our flexible LWB method is given in Algorithm 1.

### Algorithm 1 Flexible Layer-wise Binarization

**Input:** Training data $I_t$, validation data $I_v$, initial network $W^{(0)}$, and predefined metric of layer-wise priority and threshold $\tau$.

**Output:** Binarized network $W^*$.

1: $i = 0$
2: while loss$(W^{(i)}; I_v) < \tau$ do
3:   Preserve current network $W_{current} = W^{(i)}$
4:   while loss$(W^{(i)}; I_t)$ does not converge do
5:     for $l = 1 : L$ do
6:       if $W_{lk}^{(i)}$ is binarized as $B_{lk}^{(i)}$ then
7:         $I_{l+1} = \text{BinaryForward}(I_t, B_{lk}^{(i)})$
8:       else
9:         $I_{l+1} = \text{Forward}(I_t, W_{lk}^{(i)})$
10:        $B_{lk}^{(i)} = \text{BinaryBackward}(\frac{\partial \text{loss}}{\partial W_{lk}^{(i)}}, B_{lk}^{(i)})$
11:     end
12:     for the $(i+1)$ highest priority layers $\{I\}$’s do
13:       for $k = 1$ to $K_l$ do
14:         $B_{lk}^{(i)} = \frac{1}{\alpha_l} ||W_{lk}^{(i)}||_1$, $B_{lk}^{(i)} = \text{sign}(W_{lk}^{(i)})$
15:         $W_{lk}^{(i)} = \alpha_l B_{lk}^{(i)}$
16:     end
17:   end
18: $W^{(i+1)} = W^{(i)}$, $i = i + 1$.
19: end

The accuracy, we continue to binarize the parameters layer-by-layer. In this case, the number of the binarization steps is reduced from 20 to 6, and our LWB method is accelerated greatly. Moreover, it should be noted that compared with those methods that binarize all parameters simultaneously, our method requires more steps but converges much more quickly. As a result, our LWB method is at least comparable to its competitors like BWN on the runtime of the whole binarization process. More detailed experimental results and comparisons will be given in the following experimental section.

**Feasibility and Justifiability**

Network binarization is a complicated non-convex optimization problem, which requires effective algorithms that are able to converge quickly and avoid bad local optimums. Our LWB method is feasible and justifiable because the layer-wise binarization generally has better convergence than all-layer binarization. In fact, our LWB method takes advantage of a similar idea like alternating optimization and block coordinate descent (BCD) \cite{tseng2001}, which solves a complicated problem by solving a series of much simpler sub-problems. For each sub-problem, we can obtain a good initial point from the solution of the previous sub-problem, and we only need to binarize one more layer.

Empirically, denote the real-valued network obtained in each SGD step as $W_t$. If we binarize all layers simultaneously, the error between $W_t$ and its binary approximation will be aggregated and amplified through the forward propagation in the next SGD step. The large approximation error will finally affect the estimation of gradient through the
backward propagation. As a result, the learning trajectory
from the initial network \( W^{(0)} \) to the final binary network \( W^* \) will be very long and unstable. We have to choose very
small learning rate to avoid bad optimums or failures of con-
vergence. In contrast, our LWB method just binarizes one
more layer in each binarization step. In the \( i \)th step, we just
need to learn \( W^{(i+1)} \) from the initial point \( W^{(i)} \). This sub-
problem is much simpler because

1. Compared with the original problem, we have a goos ini-
tial point in this sub-problem because the distance be-
tween \( W^{(i+1)} \) and \( W^{(i)} \) is smaller than that between \( W_r \) and
\( W^{(0)} \) in general.

2. Because we don’t need to binarize all layers, the approxi-
imation errors involved in forward and backward propa-
gation in each SGD step is smaller than those in the original
problem.

In other words, the learning process of the sub-problem has
a much better initial point and more stable gradients. There-
fore, the binarization process of our LWB method is more
controllable and has much lower risk to fall into a bad local
optimum.

However, it should be mentioned that our method is not
alternating optimization or BCD because we do not fix any
variables in each sub-problem. As a result, it is hard for us
to prove the convergence of our LWB method same as BCD.
Fortunately, the following experimental results empirically
prove that our method has good convergence in practical ap-
lications.

Experiments

Implementation Details

In this work, we apply the proposed LWB method to binarize
YOLOv2 network and test its performance in pedestrian de-
tection task. Specifically, YOLOv2 is a generic object detec-
tion method, which formulates object detection problem as
a regression problem. It uses a single deep neural network to
predict the locations and class probabilities of object bound-
ing boxes. Meanwhile, the network is a full convolutional
network without fully connected layers. In Fig. 2, we present
the framework of YOLOv2 network. We use YOLOv2 in our
experiments because it achieves a trade-off between the de-
tection accuracy and speed and is one of the most important
detection methods.\(^2\)

However, the model size of YOLOv2 is up to 268 MB,
which is too large for embedded systems. We binarize the
YOLOv2 network by our LWB method and its competitors,
and evaluate these approaches on two famous pedestrian de-
tection benchmarks: Caltech and INRIA. A brief introduc-
tion of the two benchmarks is as follows:

- **Caltech.** The Caltech dataset (Dollár et al. 2009) consists
  of about 10 hours of 640 × 480 30Hz video sequences.
  Following (Hosang et al. 2015), we use Caltech10× set

\(^2\)By comparing the accuracy and speed of many state-of-the-art
object detection methods (Girshick 2015; Ren et al. 2015; Cai et
al. 2016; Redmon et al. 2016; Redmon and Farhadi 2016), we find
that YOLOv2 has the fastest detection speed.

- **INRIA.** We further use the INRIA dataset (Dalal and
  Triggs 2005) to verify the generality of our method. Since
  the size of images in this dataset is inconsistent, we
  roughly select an average size of 672 × 992. Due to the
  memory limitation, during the training process for this
dataset, we set the batch size to 32. For fair comparisons,
we follow the standard evaluation metric (Dollar et al.
2012) to get the log-average Miss Rate on False Positive
Per Image (FPPI) in \([10^{-2}, 10^{0}]\) range. The lower the miss
rate, the more accurate the detection method.

In particular, we compare the performance of our LWB
models against the real-valued models and BWN (Rastegari
et al. 2016). For fair comparison, we binarize all layers in
the network except the first and the last layers, just as same
as the BWN method. Our experiments are performed with
the Darknet framework, which is an open source neural net-
work framework (Redmon and Farhadi 2016), on a NVIDIA
GeForce Titan X GPU with CUDA 8.0 and cuDNN 5.0
configured.

In order to accelerate our LWB scheme, we propose to
binarize a batch of layers rather than a single layer in the
initial binarization step. According to the observable layer-
wise priority shown in Table 1 we can find that binariz-
ing the first 5 layers causes a significant increase of the
miss rate (with an average increment of about 1%) while
has an negligible contribution to the compression of net-
work. On the contrary, binarizing the latter layers (e.g.,
“conv10”, “conv15”, “conv19”) has no effect on or even re-
duces the miss rate. Therefore, in this work we assume that
the priority of layer is correlated with the depth of layer and
binarize a few of latter layers together in the beginning of our
binarization process. For pedestrian detection task, we first
binarize the latter 15 layers of the YOLOv2 network in the
initial binarization step and denote the binarized network as
“Pre6”. Then, we binarize a single remaining shallow layer
in each following binarization step and denote the binarized
network in each step as “Pre5”, “Pre4”, “Pre3”, “Pre2”, “Pre1”, accordingly. In

![Figure 2: The YOLOv2 framework. Conv represents
convolutional layers. Reorg layer distributes the features in each
2 × 2 grid into 4 channels. Route is a pass-through layer
concatenating two sets of feature maps in the direction of
channels.](image-url)
The average training loss on INRIA dataset was reported in Table 2. Specifically, the time required by each iteration of the three networks is consistent with our previous analysis. More important, the training of BWN network is more easier to converge due to the accuracy of our LWB method and the real-valued network only need about 30 iterations the learning rate is 0.0001, and then it is set to 0.001.

Observing the trend of the three lines, we can see that our LWB model is more easier to converge due to the accurate initial model “Pre4”. In contrast, the training of BWN model is unstable and need more iterations to converge. This phenomenon is consistent with our previous analysis. More specifically, the time required by each iteration of the three models is approximately same, about 5 seconds. Therefore, our LWB model can converge more quickly than its competitors.

Further, using the testing set of the Caltech dataset, we compare the three methods on their miss rate under different epoch. For each method, there are about 604 iterations in each epoch which takes about 55 minutes during the training process. In Fig. 5(c) we observe that the BWN requires more than 30 epochs to find the optimal solution. In contrast, our LWB method and real-valued network only need about 5 epochs to converge to the optimal state. Therefore, although the binarization of YOLOv2 network is divided into 6 steps (i.e., “Pre6”, “Pre5”, “Pre4” sequentially) by our LWB method, the total number of training epoch is almost same as that of the BWN. Moreover, the miss rate of BWN model can no longer be reduced even after more training epochs. Therefore, our LWB method achieves a good balance between the accuracy and the efficiency for network binarization.

**Comparisons on Performance**

Table 2 lists the loss of performance and the compression ratio obtained by different networks. For the Caltech dataset, the miss rate of the model “Pre6” is same as that of the original real-valued model, and for the INRIA dataset, its miss rate is only 0.8% higher than that of the real-valued model (“Ori”). At the same time, the model size of “Pre6” has also been greatly reduced from 268.2 MB to 10.5 MB. With the increase of binarized layers (from “Pre6” to “Pre1”), the miss rate increases consistently while the compression ratio increases only a little.

Figure 3: The comparison for various methods on their convergence.

Figure 4: The miss rate of networks trained by BWN and LWB on Caltech and INRIA datasets.

In this work, as the miss rate is very important for pedestrian detection task, we simply consider whether the increase of miss rate is tolerable and decide whether to continue binarization. This phenomenon is consistent with our previous analysis. More specifically, the time required by each iteration of the three models is approximately same, about 5 seconds. Therefore, our LWB model can converge more quickly than its competitors.

Further, using the testing set of the Caltech dataset, we compare the three methods on their miss rate under different epoch. For each method, there are about 604 iterations in each epoch which takes about 55 minutes during the training process. In Fig. 5(c) we observe that the BWN requires more than 30 epochs to find the optimal solution. In contrast, our LWB method and real-valued network only need about 5 epochs to converge to the optimal state. Therefore, although the binarization of YOLOv2 network is divided into 6 steps (i.e., “Pre6”, “Pre5”, “Pre4” sequentially) by our LWB method, the total number of training epoch is almost same as that of the BWN. Moreover, the miss rate of BWN model can no longer be reduced even after more training epochs. Therefore, our LWB method achieves a good balance between the accuracy and the efficiency for network binarization.

**Comparisons on Performance**

Table 2 lists the loss of performance and the compression ratio obtained by different networks. For the Caltech dataset, the miss rate of the model “Pre6” is same as that of the original real-valued model, and for the INRIA dataset, its miss rate is only 0.8% higher than that of the real-valued model (“Ori”). At the same time, the model size of “Pre6” has also been greatly reduced from 268.2 MB to 10.5 MB. With the increase of binarized layers (from “Pre6” to “Pre1”), the miss rate increases consistently while the compression ratio increases only a little.
Our LWB model achieves better results compared to BWN and LWB, respectively. Specifically, the real-valued network by the BWN method, the miss rates of those networks are higher than those obtained by our LWB method. For pedestrian detection task, the lower the miss rate as 2%, our LWB method can stop the binarization process.

Compared with the BWN method, which learns the final binarized networks from original ones by one-pass binarization, our LWB method can achieve better performance. Fig. 4 compare the increase of miss rate obtained by the BWN and our LWB on the two datasets. We can find that for “Pre5”, “Pre3”, if we learn them directly from original real-valued network by the BWN method, the miss rates of those networks are higher than those obtained by our LWB method. For pedestrian detection task, the lower the miss rate indicates the higher the accuracy of the model. The experimental result demonstrates that the networks obtained by our LWB method is better.

Finally, we compare the performances of real-valued YOLOv2, the BWN of YOLOv2 and the binarized YOLOv2 based on our LWB method with the state-of-the-art methods. As shown in Fig. 5(a) and Fig. 5(b) YOLOv2 indicates the real valued network. BWN and LWB represent the binarized YOLOv2 obtained by BWN and LWB, respectively. Our LWB model achieves 26.9% and 14% miss rate on Caltech and INRIA, which is superior to many existing methods. Although the network obtained by our LWB is less accurate than MS-CNN [Cai et al. 2016] and RPN+BF [Zhang et al. 2016], our binarized network achieves a speed of 66 FPS while detecting images of 672 x 512 pixels. In contrast, the speed of MSCNN is only 8 fps and RPN takes up to 0.5 seconds per image. Therefore, our detection rate is much higher than other detection algorithms. Furthermore, our LWB network only occupies 8.7 MB while many other methods rely on hundreds MB memory usage.

**Conclusion**

In this paper, we propose a novel and flexible network binarization method named as Layer-wise Binarization (LWB) scheme. Based on the concept of layer-wise priority, our method achieves comparable accuracy against full-precision network while resulting in $2 \times$ speed up and significant memory saving up to $32 \times$. We successfully apply our binarization method to pedestrian detection task and provide a fast and accurate pedestrian detector with network size of 8.7 MB and speed of 66 FPS. This pushes the possibility of deploying the CNN-based pedestrian detection algorithm on embedded devices. In the future work, we aim to develop more reasonable metric of priority and more effective criterion evaluation and control mechanism so as to achieve the best trade-off between compression ratio and the accuracy of the network.

**References**

[Bruna and Mallat 2011] Bruna, J., and Mallat, S. 2011. Classification with scattering operators. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 1561–1566. IEEE.

[Cai et al. 2016] Cai, Z.; Fan, Q.; Feris, R. S.; and Vasconcelos, N. 2016. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, 354–370. Springer.
[Dalal and Triggs 2005] Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In Advances in Neural Information Processing Systems, 3123–3131.

[Dollár et al. 2009] Dollár, P.; Wojek, C.; Schiele, B.; and Hoiem, D. 2009. Improved dense correspondence using oriented-patch matching. In Proceedings of the European Conference on Computer Vision, 2009. CVPR 2009, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 304–311. IEEE.

[Dollár et al. 2012] Dollár, P.; Wojek, C.; Schiele, B.; and Perona, P. 2012. Pedestrian detection: A benchmark. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 304–311. IEEE.

[Girshick 2015] Girshick, R. 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, 1440–1448.

[Gupta et al. 2015] Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 1737–1746.

[Han et al. 2015] Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, 1135–1143.

[Han, Mao, and Dally 2015] Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[Hassibi and Stork 1993] Hassibi, B., and Stork, D. G. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In Advances in neural information processing systems, 164–171.

[Hosang et al. 2015] Hosang, J.; Omran, M.; Benenson, R.; and Schiele, B. 2015. Taking a deeper look at pedestrians. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4073–4082.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, 1097–1105.

[Le Cun, Denker, and Solla 1989] Le Cun, Y.; Denker, J. S.; and Solla, S. A. 1989. Optimal brain damage. In Proceedings of the 2Nd International Conference on Neural Information Processing Systems, NIPS’89, 598–605. Cambridge, MA, USA: MIT Press.

[Park, Ahn, and Yoo] Park, E.; Ahn, J.; and Yoo, S. Weighted-entropy-based quantization for deep neural networks.

[Rastegari et al. 2016] 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.

[Redmon and Farhadi 2016] Redmon, J., and Farhadi, A. 2016. Yolo9000: better, faster, stronger. arXiv preprint arXiv:1612.08242.

[Redmon et al. 2016] Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788.

[Simpson and Zisserman 2014] Simpson, A., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[Tseng 2001] Tseng, P. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. Journal of optimization theory and applications 109(3):475–494.

[Ushio 1996] Ushio, T. 1996. Limitation of delayed feedforward control in nonlinear discrete-time systems. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 43(9):815–816.

[Xu et al. 2016] Xu, H.; Yan, J.; Persson, N.; Lin, W.; and Zha, H. 2016. Fractal dimension invariant filtering and its cnn-based implementation. arXiv preprint arXiv:1603.06036.

[Zhang et al. 2016] Zhang, L.; Lin, L.; Liang, X.; and He, K. 2016. Is faster r-cnn doing well for pedestrian detection? In European Conference on Computer Vision, 443–457. Springer.

[Zhou et al. 2016] 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. arXiv preprint arXiv:1606.06160.

[Zhou et al. 2017] Zhou, A.; Yao, A.; Guo, Y.; Xu, L.; and Chen, Y. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. arXiv preprint arXiv:1702.03044.