Training a Binary Weight Object Detector by Knowledge Transfer for Autonomous Driving

Jiaolong Xu**, Yiming Nie**, Peng Wang, Antonio M. López*
*Computer Vision Center, Universitat Autònoma de Barcelona, Spain
**Unmanned Systems Research Center, National Innovation Institute of Defense Technology, Beijing, China

***Abstract*—Autonomous driving has harsh requirements of small model size and energy efficiency, in order to enable the embedded system to achieve real-time on-board object detection. Recent deep convolutional neural network based object detectors have achieved state-of-the-art accuracy. However, such models are trained with numerous parameters and their high computational costs and large storage prohibit the deployment to memory and computation resource limited systems. Low-precision neural networks are popular techniques for reducing the computation requirements and memory footprint. Among them, binary weight neural network (BWN) is the extreme case which quantizes the float-point into just 1 bit. BWNs are difficult to train and suffer from accuracy deprecation due to the extreme low-bit representation. To address this problem, we propose a knowledge transfer (KT) method to aid the training of BWN using a full-precision teacher network. We built DarkNet- and MobileNet-based binary weight YOLO-v2 detectors and conduct experiments on KITTI benchmark for car, pedestrian and cyclist detection. The experimental results show that the proposed method maintains high detection accuracy while reducing the model size of DarkNet-YOLO from 257 MB to 8.8 MB and MobileNet-YOLO from 193 MB to 7.9 MB.**

I. Introduction

Autonomous driving requires object detectors to operate on embedded processors to accurately detect cars, pedestrians, cyclists, road signs, and other objects in real-time to ensure safety [1]. The state-of-the-art object detectors are trained with deep neural networks (DNNs) which have shown top accuracy for a wide range of computer vision tasks, such like image classification [2], semantic segmentation [3] and object detection [4], [5]. The success of deep convolutional neural networks relies on a large amount of labeled training data and powerful computing systems such as GPUs. Deep models generally have high computation cost and require large storage and memory footprints, which prohibits the deployment to resource constrained systems, *e.g.* embedded systems. One possible solution is to offload all computations to the cloud but this introduces latency and potentially privacy risks because data is processed remotely. Therefore, developing small size and energy efficient DNN object detector is an emergent task.

Network compression and prunning have attracted increasing research interests. Currently, there are mainly two categories of techniques to reduce the computational cost of DNNs. One approach is to prune the network by removing redundant weights. A typical way of pruning is to first remove weights with small magnitude and then the network is fine-tuned to recover the lost accuracy [6]. The second group of techniques is to use low-precision neural networks [7], [8], [9], [10], since conventional DNNs use float-point format, being power and storage inefficient. Weight quantization has become a popular technique that converts a baseline float-point model into a low-precision representation. The quantization algorithms can be classified into the following groups: fixed-point quantization [11], power-of-two quantization [12], ternary or binary quantization [9], [10]. Among them, binary quantization is the extreme case where the float-point weights are represented by only 1 bit. The binary weight neural networks (BWNs) have favorable largest compression ratio, *i.e.* $32\times$, but also sacrifice the most accuracy over the baseline full-precision networks [10].

How to improve the accuracies of BWNs has been a challenging problem. Focusing on developing better training strategies, recently, [13] proposed a layer-wise training method and [14] has done careful analysis on the training tricks, including learning rate, regularization, and activation approximation. Other studies try to compensate the accuracy loss by building more complex model structures to enhance the representation power [15], [16]. However, all these methods focus on the BWN itself but neglect its corresponding full-precision counterparts, *i.e.* the full-precision counterpart is not involved in the training. In this work, we consider to take the advantage of the high accuracy full-precision model to assist the training of the BWN.

Our method is inspired by Knowledge Distilling (KD) technique[17], which is originally applied to model compression and recently also to the training of low-precision networks [18]. In this work, we use a full-precision network as teacher network and a BWN as student network. We propose a knowledge transfer method which guides the BWN to mimic the responses of the intermediate layers of the teacher network during the training. As we will show in the experiments, such additional supervision actually improves the convergency of the BWN. As a result, it effectively avoids the accuracy drop in conventional BWN training. Note that the KD technique used in [17] and [18] is limited to image classification tasks, since it builds on the last layer of classification network, *i.e.* the Softmax layer. Compared to KD, our method is more flexible, since it transfers the knowledge of intermediate layers. As a result, our method can be applied to object detection as well as other tasks. Unlike [15], [16], our method increases neither the model complexity nor the computational cost, and moreover it is easy to implement.

In this work, we present the application of the proposed method to the state-of-the-art YOLO-v2 [5] object detector. However, our method is not limited to this specific detector. It can be applied to any single-stage (*e.g.* SSD [19]) or two-stage DNN-based detectors (*e.g.* Faster R-CNN[4]), or even for other tasks, *e.g.* semantic segmentation.

We use DarkNet and MobileNet [20] as the backbones in our binary weight YOLO-v2 detector. The former is the default architecture of YOLO-v2 [5] and the latter is a recent high-accuracy compact network which has much less parameters and is suitable for the deployment on mobile devices. We denote them by DarkNet-YOLO and MobileNet-YOLO respectively. We conduct experiments on KITTI dataset [21], a defacto benchmark designed for autonomous driving, for car, pedestrian and cyclist detection. Moreover, we also carried experiments on PASCAL VOC dataset [22] which contains 20 categories of general objects, to verify the generalization of the proposed method . The experimental results show that the proposed method significantly improves the accuracy of BWNs while reduces the model size of DarkNet-YOLO from 257 MB to 8.8 MB and MobileNet-YOLO from 193 MB to 7.9 MB.

II. Related work

Network quantization and binarization. Network quantization is an active research topic. Ternary [9] and binary quantization [10] aim at quantizing the network at the largest compression ratio. As a consequence, they usually suffer from accuracy degradation. BWN and XNOR-Net [10] are the most typical binary neural networks. Since XNOR-Net does not only binarize the weight but also the input, its accuracy is much worse than BWN. INQ [7] is an incremental network quantization method which progressively quantizes a full-precision network into a low-precision one whose weights are constrained to be either powers of two or zero. A layer-wise network binarization approach is studied in [14]. In our work, we also use a similar but more efficient stage-wise training strategy consisting of bainarizing groups of layers stage by stage. Compared to [14], our training strategy can reduce a lot of training iterations and meanwhile leads to a good accuracy.

Knowledge transfer methods. KT method for model compression could be dated to [23] where a compressed shallow model is trained with pseudo-data labeled by an ensemble of strong classifiers. Recently, [17] brings it back for DNNs and introduces knowledge distillation (KD). In [18], KD is also applied to the training of low-precision networks and several training schemes are studied. Inspired by KD, [24] proposes to improve the performance of the student CNN by forcing it to mimic the attention maps of the teacher network. In [24], it is further studied that KT can be treated as a distribution matching problem, which is similar to domain adaptation [25], [26]. A new KT loss function is devised to minimize the Maximum Mean Discrepancy (MMD) metric between the feature distributions of student and teacher [27]. Our work shares some characteristics of [24] and [27] in the sense that we also transfer knowledge from the intermediate layers. Our KT method is related to *curriculum learning*, as we transfer knowledge from easy tasks first and progressively increase the difficulty in the later stages.

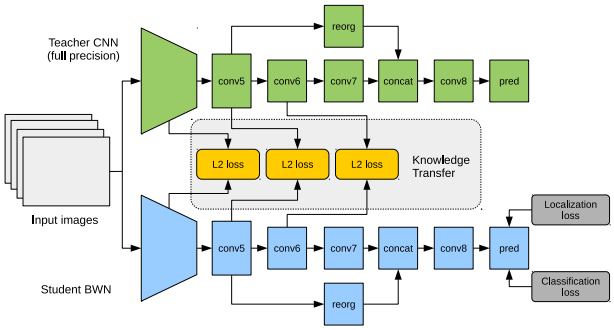

Fig. 1. The proposed knowledge transfer method for training binary weight YOLO object detector.

III. Proposed method

In this section, we first briefly revisit the binary weight neural networks of [10], which is the base of our work. Then we propose several straightforward schemes for efficient training of BWNs, including fine-tuning and state-wise binarization, which are served as strong baselines and can be combined with the proposed KT method. Finally, we elaborate the details of the proposed KT method for training high accuracy BWNs.

A. Binary weight neural network

BWN is the neural network with binary weights. Let c be the number of channels, w and h the width and the height of the filter respectively, the real valued filter $\mathbf{W} \in \mathcal{R}^{c\times w\times h}$ is estimated using a binary filter $\mathbf{B} \in \{+1, -1\}^{c\times w\times h}$ and a scaling factor $\alpha \in \mathcal{R}^{+}$, such that $\mathbf{W} \approx \alpha\mathbf{B}$. The convolution is thus approximated by $\mathbf{I} * \mathbf{W} \approx (\mathbf{I} \otimes \mathbf{B})\alpha$, where $\mathbf{I}$ is the input data, the symbol $*$ represents traditional convolution operation while $\otimes$ indicates the convolution operation only involving additions and subtractions as the weights of the filter are binary. The optimal estimation of α and $\mathbf{B}$ is obtained by solving the following optimization:

$$\alpha^*, \mathbf{B}^* = \underset{\alpha,\mathbf{B}}{\operatorname{argmin}} \|\mathbf{W} - \alpha\mathbf{B}\|_2^2 \tag{1}$$

The solution of Eq. (1) is:

$$\alpha^* = \frac{1}{n}\|\mathbf{W}\|_{l_1}, \mathbf{B}^* = \operatorname{sign}(\mathbf{W}), \tag{2}$$

where n is the number of elements in $\mathbf{W}$, $\|\cdot\|_{l_1}$ is the l_1 norm and sign is the sign function which is applied element-wise.

The training of BWN is similar to the ordinary CNNs. In each iteration, given real-valued weights from the previous iteration, the binarized weights are computed accoridng to Eq. 2, then the forward propagation of activations and backward propagation of gradients are calculated based on the scaled binary weights. Given the gradient of the scaled binary weights $\tilde{\mathbf{W}}$, the gradients of the real-valued weights are calculated by $\frac{\partial C}{\partial \mathbf{W}} = \frac{\partial C}{\partial \tilde{\mathbf{W}}}(\frac{1}{n} + \frac{\partial \operatorname{sign}}{\partial \mathbf{W}}\alpha)$. After that, the real-valued weights

are updated by gradient descent. For more details, please refer to [10].

B. Fine-tuning and stage-wise binarization

Before introducing the KT method, we first present several training schemes which can not only improve the training efficiency but also be combined with the KT method.

Although BWNs and XNOR-Net can be trained from scratch [10], fine-tuning with the pre-trained full precision network obtains a faster and better convergence. The effectiveness of such training strategy is also verified in [18]. In this work, we use the full-precision network to initialize the BWNs and fine-tune from the initialized BWNs.

As observed in [14], for BWN, the binarization of the first few layers causes significant accuracy loss while binarizing the last few layers has little effect. A layer-wise priority training strategy is studied in [14], where the weights are binarized in reverse order of the layer depth. In this work, we propose an analog but more efficient stage-wise training strategy. We first separate the layers into groups and then binarize the groups stage by stage. We also follow a reversed order, *i.e.* binarizing from the last group to the first group. Such stage-wise training can be very efficient. According to our experience, it only takes 1 or 2 epochs for the first stage training to achieve a comparable accuracy to the full-precision network. The stage-wise binarization can also be interpreted as curriculum learning. For curriculum learning, we first solve easy tasks and then gradually increase the difficulty. Binarizing the whole network from the beginning is much more difficult than stage-wise progressive binarization. The former converges much slower and may get stuck at very bad local minima.

However, fine-tuning and stage-wise binarization make very limited improvement on accuracy as they cannot provide additional supervision. In fact, we found that by carefully tuning the hyper-parameters, it is possible to train BWN from scratch to obtain the same accuracy as these strategies but it requires more iterations. In the next, based on the fine-tuning and stage-wise training, we introduce the proposed KT method which is the key to obtaining high accuracy in this work.

C. Intermediate layer knowledge transfer

Inspired by the knowledge distillation for model compression [17] and training low-precision networks [18], we propose to transfer knowledge from the intermediate layers of a full-precision network.

The overall idea is illustrated in Fig.1. The pre-trained full-precision teacher network is shown on the top of Fig.1, where *reorg* is the feature re-organization layer which stacks the neighborhood features along the channels in order to have the same spatial size as the concatenation layer. On the bottom is the student BWN, whose intermediate layers are connected to the teacher network. The knowledge transfer forces the student to output similar feature responses to the teacher. In this work, we assume the student and the teacher have exact the same network architecture. This allows the student network to be easily initialized by copying the pre-trained weights of the teacher network. However, our method can be extended to a more general case where the teacher and the student have different architectures but with only some layers in common. In such case, the student can be partially initialized from the corresponding layers of the teacher.

After initialization, we run stage-wise training to binarize the last few layers. As the first stage only takes a few epochs to converge without accuracy loss, we start the iteration of KT training based on the first stage BWN. In the forward propagation, the student network performs binarization according to Eq. (2), and intermediate feature responses of both teacher and the student are calculated. We use the simplest L_2 loss to measure the similarity of feature response between the teacher and the student. Given $\mathbf{F}_{t_i}$ and $\mathbf{F}_{s_i}$ the feature responses of the layer i in the teacher and student network respectively, the L_2 loss function minimizes the squared differences between the student (estimated) and teacher (target) features responses: $\mathcal{L}_2(\mathbf{F}_{t_i}, \mathbf{F}_{s_i}) = \|\mathbf{F}_{s_i} - \mathbf{F}_{t_i}\|_2^2$. Although other complex loss functions can also be employed, *e.g.* attention map transfer [24], or MMD [24], we find the simple L_2 loss works well in practice and leave the investigation of other loss functions as future work. The overall loss of the network can be written as follows:

$$\mathcal{L}(\mathbf{W}) = \lambda_1 \mathcal{L}_{cls}(\mathbf{y}_c, \mathbf{W}) + \lambda_2 \mathcal{L}_{loc}(\mathbf{y}_b, \mathbf{W}) + \lambda_3 \sum_{i \in K} \mathcal{L}_2(\mathbf{F}_{t_i}, \mathbf{F}_{s_i}), \tag{3}$$

where $\mathcal{L}_{cls}$ and $\mathcal{L}_{loc}$ are the classification and localization loss respectively for object detection, λ_1, λ_2 and λ_3 are the weights for each loss term, and K is the set of the indices of binary weight convolutional layers. In the backward propagation, the object detection loss, *i.e.* localization loss and classification loss, together with the feature matching L_2 loss are backward propagated in the student network to compute gradients of the weights. The weights of BWN are then updated using the standard solver *e.g.* SGD[28] or ADAM[29]. Note that the weights of the teacher network are fixed, thus the computation of backward propagation of the teacher network is not needed. The L_2 loss between teacher and student network forces the student to mimic the feature response of the full-precision network. In this way, the pre-trained full-precision network transfers knowledge to the student BWN. The transfered knowledge provides additional supervision to the training of BWN and guides the optimization of BWN along an optimal path.

IV. Experiments

We evaluate our proposed method on the KITTI dataset [21] which is a standard object detection benchmark designed for autonomous driving. The KITTI dataset contains three categories, which are car, pedestrian and cyclist. To evaluate the performance for more categories, we also carry out the experiments on PASCAL VOC dataset [22] which has 20 categories. The mean average precision (mAP) versus recall criterion is adopted to evaluate the detection performance.

Layer	Filter shape	*M0*	*M1*	*M2*	**KT**
Conv1	$3 \times 3 \times 32$				
Conv2	$3 \times 3 \times 32 \times 64$			✓	✓
Conv3_1	$3 \times 3 \times 64 \times 128$			✓	✓
Conv3_2	$1 \times 1 \times 128 \times 64$			✓	✓
Conv3_3	$3 \times 3 \times 64 \times 128$			✓	✓
Conv4_1	$3 \times 3 \times 128 \times 256$			✓	✓
Conv4_2	$1 \times 1 \times 256 \times 128$			✓	✓
Conv4_3	$3 \times 3 \times 128 \times 256$			✓	✓
Conv5_1	$3 \times 3 \times 256 \times 512$		✓	✓	✓
Conv5_2	$1 \times 1 \times 512 \times 256$		✓	✓	✓
Conv5_3	$3 \times 3 \times 256 \times 512$		✓	✓	✓
Conv5_4	$1 \times 1 \times 512 \times 256$		✓	✓	✓
Conv5_5	$3 \times 3 \times 256 \times 512$		✓	✓	✓
Conv6_1	$3 \times 3 \times 128 \times 256$		✓	✓	✓
Conv6_2	$1 \times 1 \times 256 \times 128$		✓	✓	✓
Conv6_3	$3 \times 3 \times 128 \times 256$		✓	✓	✓
Conv6_4	$3 \times 3 \times 128 \times 256$		✓	✓	✓
Conv6_5	$1 \times 1 \times 256 \times 128$		✓	✓	✓
Conv7_1	$3 \times 3 \times 1024$	✓	✓	✓	✓
Conv7_2	$3 \times 3 \times 1024 \times 1024$	✓	✓	✓	✓
Conv8_1	$3 \times 3 \times 1024 \times 1024$	✓	✓	✓	✓
Pred	$1 \times 1 \times 1024 \times 125$				
Size (MB)	257	82	12	8.8	8.8

TABLE I
THE BINARY WEIGHT LAYERS AND MODEL SIZES OF **DARKNET-YOLO** BASED MODELS.

Layer	Filter shape	*M0*	*M1*	*M2*	**KT**
Conv5_1_dw	$3 \times 3 \times 512$				
Conv5_1_sep	$1 \times 1 \times 512 \times 512$			✓	✓
Conv5_2_dw	$3 \times 3 \times 512$				
Conv5_2_sep	$1 \times 1 \times 512 \times 512$			✓	✓
Conv5_3_dw	$3 \times 3 \times 512$				
Conv5_3_sep	$1 \times 1 \times 512 \times 512$			✓	✓
Conv5_4_dw	$3 \times 3 \times 512$				
Conv5_4_sep	$1 \times 1 \times 512 \times 512$			✓	✓
Conv5_5_dw	$3 \times 3 \times 512$				
Conv5_5_sep	$1 \times 1 \times 512 \times 512$			✓	✓
Conv5_6_dw	$3 \times 3 \times 512$				
Conv5_6_sep	$1 \times 1 \times 512 \times 1024$		✓	✓	✓
Conv6_dw	$3 \times 3 \times 1024$				
Conv6_sep	$3 \times 3 \times 1024 \times 1024$		✓	✓	✓
Conv7_1	$3 \times 3 \times 1024$	✓	✓	✓	✓
Conv7_2	$3 \times 3 \times 1024 \times 1024$	✓	✓	✓	✓
Conv8_1	$3 \times 3 \times 1024 \times 1024$	✓	✓	✓	✓
Pred	$1 \times 1 \times 1024 \times 125$				
Size (MB)	193	19	13	7.9	7.9

TABLE II
THE BINARY WEIGHT LAYERS AND MODEL SIZES OF **MOBILENET-YOLO** BASED MODELS.

We use the state-of-the-art object detector YOLO-v2 [5] in our experiments for its efficiency and high accuracy. By default, YOLO uses DarkNet as its backbone network. In addition to that, we also use MobileNet [20] as the backbone, which is much more compact than DarkNet and meanwhile obtains similar accuracy on image classification. We denote by DarkNet-YOLO and MobileNet-YOLO for these two detectors. The filter sizes of the DarkNet-YOLO and MobileNet-YOLO are listed in Table I and Table II respectively, where the bold layer names are the candidate binary layers in our experiments. For each architecture, five types of models are compared, namely *FP*, *M0*, *M1*, *M2* and *KT*. The definition of the models are as follows. *FP*: the full-precision model; *M0*, *M1* and *M2*: the BWN with the 1st, 2nd and 3rd stage binarization; *KT*: the BWN initialized from *M0* and trained with knowledge transfer. The binary weight layers of each model as well as the model size are shown in Table I and Table II.

A. Implementation details

The proposed methods are implemented using MXNET [30]. Unless otherwise specified, we use following settings. We take ADAM [29] optimizer with the initial learning rate of $1e-4$. The default batch size is set to 10 for KITTI and 30 for PASCAL VOC. The training accuracy is measured by mAP. We train the model for around 500 and 300 epochs on KITTI and PASCAL VOC respectively, until the models converge. We use 5 anchors for the YOLO detectors for all experiments. As it is pointed out in previous literature [10], for BWNs, binarizing the first or the last layer will cause significant accuracy drop, thus we keep the first and the last layer in full-precision.

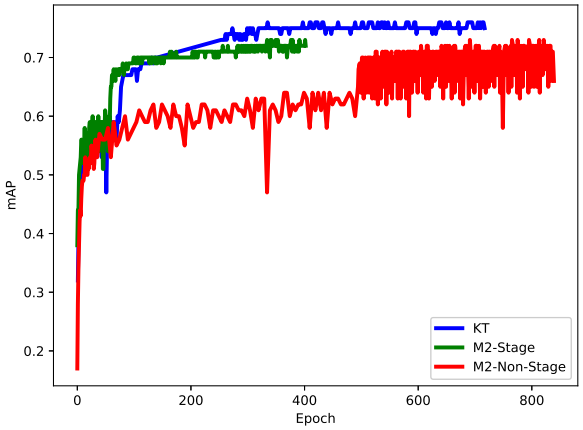

Fig. 2. Comparison of model convergence rate on KITTI.

Method	mAP	Pedestrian	Car	Cyclist	Size (MB)
FP	78	68	89	77	257
M0	78	67	89	76	82
M1	76	65	88	75	12
M2	72	58	87	72	8.8
KT	76	64	88	76	8.8

TABLE III
SUMMARY OF **DARKNET-YOLO** DETECTION ACCURACY AND MODEL SIZE ON KITTI OBJECT DETECTION BENCHMARK.

Method	mAP	Pedestrian	Car	Cyclist	Size (MB)
FP	78	64	89	77	193
M0	76	64	88	76	19
M1	73	60	86	72	13
M2	70	57	85	67	7.9
KT	72	58	87	72	7.9

TABLE IV
SUMMARY OF **MOBILENET-YOLO** DETECTION ACCURACY AND MODEL SIZE ON KITTI OBJECT DETECTION BENCHMARK.

B. KITTI object detection

The KITTI dataset contains 7381 training images. We randomly split it in half as training set and validation set. All images are scaled to canonical size of 1248×384. We report mean average precision on the validation set.

The results of DarkNet-YOLO and MobileNet-YOLO are shown in Table III and Table IV respectively. We report mAP across categories as well as the average precision (AP) of each category. The first row of each table shows the results of the full precision model. Both DarkNet and MobileNet based detectors obtain similar accuracy. The second row is the 1st stage BWN model *M0* which binarizes the $Conv\_7\_1$, $Conv\_7\_2$ and $Conv\_8$ layers. This model has equivalent accuracy as the *FP* but with significant smaller model size, *i.e.* 82 MB versus 257 MB and 19 MB versus 193 MB, which indicates that binarizing the last few layers has little effect to the accuracy. *M1* and *M2* further reduce the model size with more layers binarized. However, there is around 2 to 5 percentage points accuracy drop for *M1* and 6 to 8 for *M2*. The overall performance of DarkNet-YOLO looks more robust than MobileNet-YOLO which may due to that MobileNet is too compact to be further compressed. The result of the proposed *KT* is shown in the last row. *KT* has the same binarization level as *M2*, *i.e.* with the model size of 8.8 MB and 7.9 MB for DarkNet-YOLO and MobileNet-YOLO respectively. However, *KT* achieves better accuracy than *M2*, showing the effectiveness of the proposed method.

Fig. 2 depicts the model accuracy on different epochs of the training of DarkNet-YOLO. In this figure, we compare the convergence rate of *KT*, *M2-Stage* and *M2-Non-Stage*. *M2-Stage* is the *M2* BWN fine-tuned from *M1*, *i.e.* stage-wise training, while *M2-Non-Stage* is the BWN fine-tuned from *FP*, *i.e.* without stage-wise training. *KT* has much faster and better convergence rate than the other two models, which verifies the effectiveness of the additional supervision. *M2-Non-Stage* shows the worst convergence rate. In order to achieve the best accuracy with *M2-Non-Stage*, we have to carefully adjust the learning rate manually. The final accuracy is close to *M2-Stage* but the curve is still unstable, showing the difficulty of convergence.

Fig. 3 shows some typical failure cases of *KT* (right column) when comparing to *FP* (left column). We can see that *FP* achieves better detection accuracy for occluded cars and poorly illumination pedestrians which are typical difficult examples for object detectors. The per-category accuracy in Table III also shows that *KT* losses most of the accuracy on pedestrian detection, but achieves comparable accuracy on car and cyclist detection. The first row of Fig. 3 shows that *KT* even occasionally outperforms *FP* for the cyclist detection.

C. PASCAL VOC

In this section, we extend the proposed method for general object detection. We conduct the experiments on PASCAL VOC dataset, which contains 20 categories of common objects. Specifically, we train on VOC2007 trainval and VOC2012 trainval (16551 images) and test on VOC2007 test (4952 images). For these experiments, all images are re-scaled to canonical size of 416×416. Table V and Table VI present the detection accuracies of DarkNet-YOLO and MobileNet-YOLO models respectively. We obtain similar results as on KITTI dataset. The proposed *KT* method again outperforms *M2* by 3 percentage points, showing its effectiveness for general object detection.

V. Conclusion

In this work, we address the problem of how to train a compact binary weight object detector with high accuracy. First, we reveal that both fine-tuning from full-precision network and the stage-wise binarization are critical and efficient for training BWNs. Moreover, to further improve the model accuracy, we propose to transfer intermediate knowledge from the full-precision network. The experimental results show that the proposed method maintains a high detection accuracy while significantly reduces the model size with a compression rate around $30\times$. For the future work, we would like to combine other techniques to further improve the accuracy, *e.g.* attention map transfer [24], MMD [27] or domain adaptation [25].

Acknowledgment

This work is supported by National Natural Science Foundation of China (NSFC, NO. 61601508). Antonio M. López acknowledges the Spanish project TIN2017-88709-R (Ministerio de Economia, Industria y Competitividad) and the Spanish DGT project SPIP2017-02237, as well as the Generalitat de Catalunya CERCA Program and its ACCIO agency.

References

[1] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," *ArXiv e-prints*, 2016.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[5] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *CVPR*, 2017.

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

[7] Y. G. L. X. Y. C. Aojun Zhou, Anbang Yao, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *ICLR*, 2017.

[8] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *ArXiv e-prints*, 2016.

[9] N. Mellempudi, A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey, "Ternary neural networks with fine-grained quantization," in *NIPS*, 2017.

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," in *ICLR*, 2014.

[12] D. A. Gudovskiy and L. Rigazio, "ShiftCNN: Generalized Low-Precision Architecture for Inference of Convolutional Neural Networks," *ArXiv e-prints*, Jun. 2017.

[13] G. H. Wei Tang and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Fig. 3. Sample detections of **DarkNet-YOLO** on KITTI dataset (Left: **FP**, Right: **KT**).

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FP	70	74	77	68	59	41	78	79	84	49	76	70	79	79	75	73	42	70	70	83	71
M0	69	71	78	68	57	40	77	79	82	48	73	69	78	79	76	72	41	66	66	84	71
M1	66	69	75	65	57	42	75	76	78	46	67	63	71	77	77	72	37	69	61	81	71
M2	62	64	73	56	48	28	73	74	74	39	64	65	66	77	70	67	34	57	62	77	62
KT	65	67	74	60	52	37	74	76	77	44	67	65	70	76	74	70	38	61	63	79	67

TABLE V
RESULTS OF **DARKNET-YOLO** MODELS ON PASCAL VOC2007 TESTING SET.

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FP	69	71	78	70	56	41	75	76	82	42	73	69	77	79	74	71	43	65	62	81	70
M0	68	70	78	67	56	44	75	78	79	41	73	68	76	74	75	74	46	68	61	78	71
M1	65	68	77	67	50	39	73	76	80	42	70	60	73	74	74	72	43	66	59	79	69
M2	60	67	72	54	46	31	68	73	71	37	64	56	64	71	66	69	35	60	56	71	61
KT	63	66	75	60	52	36	71	75	72	42	72	63	68	71	69	71	33	62	60	73	64

TABLE VI
RESULTS OF **MOBILENET-YOLO** MODELS ON PASCAL VOC2007 TESTING SET.

[14] L. Zhuang, Y. Xu, B. Ni, and H. Xu, "Flexible Network Binarization with Layer-wise Priority," *ArXiv e-prints*, Sep. 2017.

[15] X. Lin, C. Zhao, and W. Pan, "Towards Accurate Binary Convolutional Neural Network," in *NIPS*, 2017.

[16] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao, "Performance guaranteed network acceleration via high-order residual quantization," in *ICCV*, 2017.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS 2014 Deep Learning Workshop*, 2014.

[18] A. Mishra and D. Marr, "Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy," *ArXiv e-prints*, Nov. 2017.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv e-prints*, 2017.

[21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, Washington, DC, USA, 2012.

[22] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[23] A. N.-M. Cristian Bucila, Rich Caruana, "Model compression," in *KDD*, 2006.

[24] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[25] M. Masana, J. van de Weijer, L. Herranz, A. D. Bagdanov, and J. M. Alvarez, "Domain-adaptive deep network compression," in *ICCV*, 2017.

[26] J. Xu, S. Ramos, D. Vázquez, and A. López, "Domain adaptation of deformable part-based models," *T-PAMI*, vol. 36, no. 12, pp. 2367–2380, 2014.

[27] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," in *NIPS*, 2017.

[28] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2014.

[30] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNET: A flexible and efficient machine learning library for heterogeneous distributed systems," in *NIPS Workshop on Machine Learning Systems*, 2015.