EFFICIENT OBJECT DETECTION USING BINARY NEURAL NETWORKS

Breno Skuk[1] *Dimitris Milioris*[2] *Joachim Wabnig*[3]

[1]Télécom Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France
[2]Nokia Bell Labs, Centre de Villarceaux, 91620 Nozay, France
[3]Nokia Bell Labs, 21 JJ Thomson Avenue, CB3 0FA Cambridge, United Kingdom

ABSTRACT

Recent advances in the field of neural networks with quantized weights and activations down to single bit precision have allowed the development of models that can be deployed in resource constrained settings, where a trade-off between task performance and efficiency is accepted. Here we design an efficient single stage object detector based upon the CenterNet architecture containing a combination of full precision and binarized layers. Our model is easy to train and achieves comparable results with a full precision network trained from scratch while requiring an order of magnitude less FLOP. Even in cases where the efficient models show lower task performance than their full precision counterparts, their use is still justified by the speedup of up to 58$\times$ that comes with performing convolutions of binary weights and binary activations. This opens the possibility of deploying an object detector in applications where time is of the essence and a GPU is absent. We study the impact of increasing the ratio of binarized MAC operations without significant increases in the total amount of MAC operations. Our study offers an insight into the the design process of binary neural networks for object detection and the involved trade-offs.

Index Terms— computer vision, object detection, binary neural networks

1. INTRODUCTION

To train Deep Neural Networks we exploit their graph-like structure representing simple MAC operations to perform parallel computations. Due to this, graphical processing units have become imperative for the training and inference of DNNs.

A new challenge emerges when adapting DNN models and techniques to be used without a GPU. This is currently a barrier on the widespread application of DNNs by the industry in settings where time is of the essence and hardware is limited. In this context, binary neural networks (BNNs) may be used as a solution [1]. A BNN is the extreme case of a quantized DNN, and achieves faster computation at inference time by constraining weights and activations to only two possible values (e.g. -1 or 1). Although this quantization reduces the representational capability of the network, it speeds up inference time on CPU by replacing multiply-accumulate (MAC) operations by XNOR operations and population counts. As MACs are among the most space and power hungry operations used in the implementation of neural networks [2], the increased speed and decreased model size can justify the decrease in a model's performance. In this study we binarize a one stage object detector and analyse the impact of this quantization technique on different parts of the architecture. As our goal is to depart from a full precision model and binarize it, our choice of object detection model is the CenterNet architecture proposed by [3]. Our results serve as a proof of concept of BNNs in computer vision, as a documentation of the binarization process itself and hopefully as an easy to use object detection pipeline for those aiming to apply BNNs to object detection tasks. Our code and the necessary documentation is available on GitHub [1].

2. RELATED WORK

In this article we address the task of 2D Object Detection, which ultimately consists of regressing the position of objects and assigning them a class label. Two main types of detectors are present in the literature.

Single-stage detectors rely on a single network to both locate the objects in the image and infer their class.

Two-stage detectors rely on two differentiated networks. One is responsible for generating regions of interest while the other is responsible for processing these regions and inferring the object class.

Although two stage detectors achieve overall better results on COCO and PascalVOC datasets, they tend to be slower and have a higher number of parameters [3]. We note that recent advances have produced fast two stage detectors as in [4], achieving state of the art performance among convolutional networks.

The current state of the art of object detection is dominated by transformer architectures but these models tend to

[1]https://github.com/brenoskuk/BinaryCenterNet

increase model size for improved results, which is contrary to the scope of this research [5].

To our knowledge, there are still no published attempts at binarized transformers applied to object detection, possibly due to the novelty of the technique. The state of the art in binary neural networks is achieved by [6] which employs a layer-wise search algorithm to train a 1-bit detector using a full precision teacher network. This approach manages to achieve a performance almost identical to that of the original model in full precision. Another approach to BNN object detection [7] apply the information bottleneck principle for redundancy removal to binarize architectures. Both this methods binarize a backbone such as VGG16 and use it on single stage or two stage detector. Figure 1 presents this results using the single stage detector SSD300[8].

In this work we binarize the CenterNet model. This network is an improvement to the approach adopted by CornerNet [9]. Both methods, CornerNet and CenterNet, produce heatmaps that approximate the probability of finding the corners and the center in the corresponding input image, respectively. We point out that there is another network called CenterNet [10], that employs a similar approach by detecting corners, centers and using a special pooling technique.

3. PROPOSED APPROACH

To leverage the speedup in inference time of binary layers and activations, state of the art binarized networks use the set $\{-1, 1\}$ as binary values. Other values could theoretically be used, for instance $\{0, 1\}$, as in the boolean multilayer perceptron [11].

The binarization of weights, a strong regularization, can be solved using latent weights during training [2]. Other techniques assume the discrete nature of the optimization landscape and implement custom optimizers such as [12].

The binarization of feature maps can be modeled as the application of a sign non-linearity on either the input or on the output of a linear layer. Since the output values of binary MAC operations are the result of the population count of a binary vector and therefore integer valued, state of the art binary methods prefer the placement of sign at the input of binary layers [2].

The sign mapping induces the problem of having an undefined derivative for use backpropagation. A working solution is to simply replace the derivative of the sign by other mappings such as the clip function, a technique called the straight-through estimator. The precise theoretical understanding of why this works or if there are better alternatives for this remains an open question [13].

Our binarization model uses a sign function at the input of binarized layers and restricts the kernel to binary values. During the training phase we use latent weights and the straight-through estimator for backpropagation.

The essence of CenterNet is to generate one heatmap for each class of objects to be detected. The network is then trained to predict the center of objects, the scale of the corresponding bounding box and the offset of the center from the input image to output heatmaps. The loss function, the preprocessing and postprocessing of images and bounding boxes are kept the same as in the original CenterNet method. We focus our binarization study on each individual part of our CenterNet architecture: the stem, the encoder, the decoder and the heads.

4. IMPLEMENTATION DETAILS

To implement the binarized layers we use the Larq library. Larq is an ecosystem of open-source Python packages for building, training and deploying BNNs, enabling efficient inference on mobile and edge devices [14].

We create a modular declaration for our CenterNet that allows for fine adjustments of the number of parameters of the architecture. Due to it's simplicity and effectiveness, a custom version of QuickNet [14] is used as our encoder and a series of full precision transposed 2D convolutions as decoder. We set the number of filters, the number of residual layers on each scale and the depth of the network to be adjustable. In the following items we introduce the notation used to represent the network at each module.

Stem: Given an input size z and a number of feature maps f_s, we declare our stem architecture as $st(z, f_s)$:

A (3×3) convolution layer with stride 2 that outputs $r/4$ feature maps followed by a batch-normalization and ReLU. Next a (3×3) depthwise 2d convolution with stride 2 followed by batch normalization is applied. The output goes through a (1×1) convolution that produces f_s feature maps followed again by batch-normalization. The output of this module should have one fourth of the input size and is directly fed to the encoder.

As f_s in our experiments is set as the same number of feature maps of the first encoder layer f_{e1}, we can make use of the adopted encoder name to simplify the notation. For instance, for QuickNet we have $st(z, f_{e1}) = qn(z)$.

Encoder: To compensate the quantization bottleneck the residual connection only bypasses one binarized (3×3) convolution and is followed by ReLU and batch-normalization with *momentum* $= 0.9$, $\epsilon = 1e - 5$, see [15]. To downsample the feature maps we use a transition layer composed of a ReLU followed by a (3×3) depthwise fixed blur kernel convolution and then a (3×3) full precision convolution followed by batch-normalization. The fixed blur kernel layer is used for its apparent property of inducing shift invariance as suggested by [16].

Given N_e scales with corresponding pairs of feature maps and number of residuals for each scale $\{(n_i, f_{ei})|i \in [1, N_e]$, we define the encoder $qn([n_1 \times f_{e1}, ...n_{N_e} \times f_{eN_e})\}$ by repeatedly applying a series of n_i residuals containing f_{ei} feature maps followed by a transition layer to a lower scale. And we

repeat for each i. The best results were obtained with $N_e = 4$.

Decoder: Given a kernel size k and N_d scales with corresponding number of feature maps for each scale $\{(f_{di})|i \in [1, N_d]\}$, we define the decoder $tp_{(k)}([n_1 \times f_{d1}, ...n_{N_d} \times f_{dN_d})$ by applying in series a $(k \times k)$ transposed 2d convolution with stride 2 that outputs f_{di} feature maps followed by batch normalization and ReLU. Given the composition of the modules the output of the decoder should have a scale of $s = (2^{N_d - N_e - 2})$ with respect to the input resolution.

Head: Given a number of feature maps f_h, we define the head $h(f_h)$ by separately applying three (3×3) convolutional layers that output f_h feature maps to the output of the decoder, creating three head branches. Each head branch then passes trough a batch-normalization and ReLU and another (1×1) convolutional layer. The two heads responsible for the offset and scale predictions output mappings of dimension $(z/s, z/s, 2)$ while the head that predicts heatmaps should have dimension $(z/s, z/s, c)$ where c denotes the number of classes the network has to predict.

4.1. Parameters, MACs, FLOP and Binarization Ratio

The speed of inference of a network depends not only on the number of parameters but also on the topology of the architecture and the device that will run it. The basic indicators of computational complexity in binary neural networks are the number of MAC operations M and the ratio of binarized MACs r_b.

Another measure present in the literature is that of equivalent FLOP, which assumes that the operations are executed on a 64 bit architecture where, theoretically, one FLOP can be used to compute 64 binary MACs [17]. Although not general, this assumption of the use of a 64 bit CPU allows a rough comparison of quantized and full precision architectures. We can easily obtain the equivalent FLOP as portrayed in Eq. 1.

$$\text{FLOP} = \frac{Mr_b}{64} + M(1 - r_b) \tag{1}$$

This will form the basis for our model comparisons.

5. EXPERIMENTS AND RESULTS

We train all of our networks with the same hyper-parameters with satisfying results. We use batch size 32, 200 epochs, and the ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e - 08$. The learning rate starts at 0.001 and learning rate decay is applied when there is a plateau of the train loss, the learning rate is set to 0.0001 at epoch 100. Data augmentation is used for the experiments consisting of random rotations, translations, scaling and shearing.

We design a series of architectures that are indexed in Table 1. Table 2 contains the comparative performance of the models evaluated on the PascalVOC dataset. Figure 1 offers a graphical overview of our results.

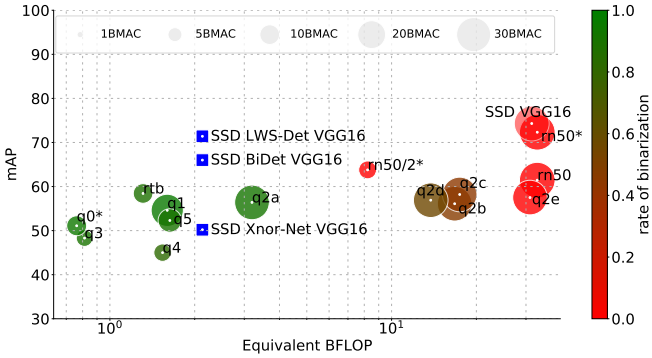

Fig. 1. Performance our experiments listed in Table 2. We also plot three different results for the different binarization techniques of the SSD object detector using the same design based on VGG16 as a backbone. The state of the art is given by LWS[6]. The area of the circles is proportional to the number of MAC operations of a model. The blue squares indicate that data on MAC and rate of binarization was not available.

Some of the insights of the study are highlighted below:

Input size: One can argue that reducing input size by a factor s reduces the number of parameters of the network by $\mathcal{O}(s^2)$. However this comes at a considerable decrease of mAP, around 10%, even on the full precision model. As we observe that the equivalent FLOP of a 512×512 input binary model is lower than that of a 256×256 full precision model with similar mAP, the use of the standard input of 512×512 is more than justified on binarized detectors.

Encoder depth: Depth can be used to linearly control the amount of parameters of the encoder with a somewhat consistent increase in mAP. Furthermore if the rest of the architecture is fixed we can increase the binarization ratio of the architecture in this way.

Decoder method: Our best results are obtained using full precision 2D transposed convolutions. Binary layers seem to critically decrease the performance of models when placed on transition layers where residual connections cannot be implemented without added layers or copying feature maps. This was already observed in binarized classifiers. We also try different kernel sizes for the transposed 2D convolutions and observe that although increasing the kernel size possibly increases the mAP, it also comes at the cost of more full precision MAC operations.

Decoder size: As the best results were obtained when using a full precision decoder, we show that despite this we can still achieve a high binarization ratio of 90%. For instance, we can reduce the number of feature maps needed to reconstruct the center of objects and infer their size. The reason for this asymmetry could be that the encoder needs to process a great variety of inputs while the decoder only needs to create feature maps with the center of objects and enough information such that size and offset are still retrievable.

Table 1. Specification of a selection of architectures we tested following the conventions established in Section 4. The architecture is determined by hyper-parameters given to the corresponding builder method and by the input size. r_b refers to the ratio of binarization. We note that $rn(256)$ refers to a non modified resnet50 stem module with a 256×256 input size.

Id	Input	Encoder	Decoder	Heads	BMACs	$\mathbf{r_b}$
rn50	$rn(512)$	Resnet50	$tp_{(4)}([256, 128, 32])$	$h(64)$	32.9	0
rn50/2	$rn(256)$	Resnet50	$tp_{(4)}([256, 128, 32])$	$h(64)$	8.22	0
q0	$qn(512)$	Quicknet	$tp_{(4)}([32, 16, 4])$	$h(64)$	10.3	0.9407
q1	$qn(512)$	$qn([4 \times 128, 4 \times 256, 4 \times 324, 4 \times 512])$	$tp_{(2)}([64, 64, 32])$	$h(32)$	26.8	0.9553
q2	$qn(512)$	$qn([2 \times 128, 4 \times 256, 8 \times 324, 10 \times 512])$	$tp_{(4)}([128, 64, 16])$	$h(64)$	31.0	0.9112
q3	$qn(512)$	$qn([4 \times 64, 4 \times 128, 4 \times 162, 4 \times 256])$	$tp_{(4)}([64, 32, 8])$	$h(32)$	3.56	0.8999
q4	$qn(512)$	$qn([4 \times 64, 4 \times 128, 4 \times 162, 4 \times 256])$	$tp_{(2)}([64, 64, 64])$	$h(32)$	7.84	0.8166
q5	$qn(512)$	$qn([8 \times 64, 8 \times 128, 8 \times 162, 8 \times 256])$	$tp_{(2)}([64, 64, 64])$	$h(32)$	14.2	0.8991
rtb	$rtb(512)$	real to binary [18]	$tp_{(4)}([32, 16, 4])$	$h(64)$	9.93	0.8817

Table 2. Comparative detection performance in terms of mean average precision (mAP) of the proposed experiments. The networks were trained using the routine specified in section 5. We highlight models that used transfer learning with an asterisk(*). Variations based on the same architecture are succeeded by a letter.

Id	Observation	FLOP	mAP
rn50*	Transfer Learning	32.9	72.3
rn50	Train from scratch	32.9	61.4
rn50/2*	Transfer Learning and 256 input	8.22	63.8
q0*	Transfer learning	0.76	53.3
q1	less features on decoder and head	1.60	54.6
q2a	increasing depth	3.19	56.4
q2b	Binarize first half of encoder	16.74	56.1
q2c	Binarize second half of encoder	17.44	58.2
q2d	Random (p=0.5) binarization of encoder	13.77	56.9
q2e	Removed binarization	31.0	57.5
q3	Same as qn5 but lighter	0.81	51.3
q4	Less params on encoder	1.53	40.0
q5	Same as qn4 but deeper	1.63	47.3
rtb	real to binary[18] as encoder	1.31	58.4

Transfer learning: We test the use of pretrained binarized encoders and verify that the available Quicknet pretrained on Imagenet does not show improved final mAP. On the other hand, pretraining and finetuning does contribute to a 10% increase in mAP for the full precision model based upon Resnet50.

Binarization and regularization: We verify that in the binary setting the introduction of dropout can be disadvantageous, possibly because it introduces an information bottleneck on training. Furthermore, binarizing only the second half of the encoder results in a better model compared to the same non binarized architecture, possibly due to the additional regularization from binarization.

5.1. PascalVOC Dataset

The chosen dataset for training and validation was the PascalVOC object detection dataset [19]. It contains images with bounding box ground truth labels of objects belonging to 20 classes.

The dataset is not the standard choice for evaluation for current state of the art methods but serves well to demonstrate the potential of architectures without the need of time consuming training routines.

We follow the practice of training with the union of Pascal VOC 2007trainval and PascalVOC 2012trainval and validating with Pascal 2007test [20].

The metric used to evaluate the dataset is the mean Average Precision (mAP) across all classes with IoU threshold 0.5.

6. CONCLUSION

This study serves as a proof-of-concept that showcases how designing binary neural networks object detectors requires modifications to compensate the expected performance issues. Some of these seem to be related with bottlenecks due to representational complexity, such as the need of full precision layers whenever downsampling or upsampling is performed.

Other modification, however, seem to increase the performance of binary neural networks on inference time with almost no added complexity in terms of total parameters and binarization ratio. Examples of this are the seemingly mandatory use of skip connections and the introduction of additional constraints on the binarization procedure as seen in [6]. These results points in the direction that the degraded performance of binary neural networks is not simply a limitation imposed by a reduced capacity of the network to represent data as discussed in [21]. Rather, it could be related to the instability during training time and non-robustness of current architectures and optimization algorithms in the BNN framework.

7. REFERENCES

[1] Chunyu Yuan and Sos S. Agaian, "A comprehensive review of binary neural network," *arXiv/2110.06804*, 2021.

[2] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Cham, 2016, pp. 525–542, Springer International Publishing.

[3] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl, "Objects as points," *arXiv/1904.07850*, 2019.

[4] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl, "Probabilistic two-stage detection," *arXiv/2103.07461*, 2021.

[5] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo, "Swin transformer v2: Scaling up capacity and resolution," *arXiv/2111.09883*, 2021.

[6] Sheng Xu, Junhe Zhao, Jinhu Lü, Baochang Zhang, Shumin Han, and David Doermann, "Layer-wise searching for 1-bit detectors," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5678–5687.

[7] Ziwei Wang, Ziyi Wu, Jiwen Lu, and Jie Zhou, "Bidet: An efficient binarized object detector," *arXiv/2003.03961*, 2020.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Cham, 2016, pp. 21–37, Springer International Publishing.

[9] Hei Law and Jia Deng, "Cornernet: Detecting objects as paired keypoints," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 642–656, Mar 2020.

[10] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian, "Centernet: Keypoint triplets for object detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6568–6577.

[11] D.L. Gray and A.N. Michel, "A training algorithm for binary feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 176–194, 1992.

[12] K. Helwegen, J. Widdicombe, L. Geiger, Ze. Liu, KT. Cheng, and R. Nusselder, "Latent weights do not exist: Rethinking binarized neural network optimization," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, vol. 32, Curran Associates, Inc.

[13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv/1308.3432*, 2013.

[14] Lukas Geiger and Plumerai Team, "Larq: An open-source library for training binarized neural networks," *Journal of Open Source Software*, vol. 5, no. 45, pp. 1746, Jan. 2020.

[15] Milad Alizadeh, Javier Fernández-Marqués, Nicholas D. Lane, and Yarin Gal, "A systematic study of binary neural networks' optimisation," in *International Conference on Learning Representations*, 2019.

[16] Richard Zhang, "Making convolutional networks shift-invariant again," in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 7324–7334, PMLR.

[17] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., Cham, 2018, pp. 747–763, Springer International Publishing.

[18] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," *arXiv/2003.11535*, 2020.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, June 2010.

[20] GluonCV, "Model zoo," `https://cv.gluon.ai/model_zoo/detection.html`, 2021, Accessed: 2022-02-22.

[21] Shilin Zhu, Xin Dong, and Hao Su, "Binary ensemble neural network: More bits per network or more networks per bit?," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4918–4927.