BinaryViT: Pushing Binary Vision Transformers Towards Convolutional Models (Supplementary Material)

Phuoc-Hoan Charles Le *
le.charles55@gmail.com

Xinlin Li
Huawei Noah's Ark Lab
xinlin.li1@huawei.com

	Output size	BinaryViT
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} C_1 = 64 \\ R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	$\begin{bmatrix} C_2 = 128 \\ R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 4$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} C_3 = 256 \\ R_3 = 1 \\ N_3 = 4 \\ E_3 = 4 \end{bmatrix} \times 8$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	$\begin{bmatrix} C_4 = 512 \\ R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 4$

Table 1. Architectural settings for BinaryViT where C_i is the base hidden dimension, R_i is the reduction ratio of the Bi-SR-MHA, N_i is the number of heads in the attention, and E_i is the expansion ration of the FFN in Stage i.

1. Architecture details

Table 1 shows the architectural settings of the BinaryViT. We designed the pyramid architecture in this way such that we still match the number of parameters of the DeiT-S model [10] and for the ReAcNet method [6] to be compatible with the transition from the 2nd to the 3rd stage. If the third stage has a base channel dimension of 320, we wouldn't be able to apply the concatenate operation properly as in [6], since 320 is not divisible by 128.

*This work was done when Phuoc-Hoan Charles Le was an intern at Huawei Noah's Ark Lab Montreal Research Center.

2. Experimental settings

We train the binary ViT from scratch mainly following the hyperparameters from DeiT [10], using PyTorch [9] and we train the model with Adam [5, 8] optimizer for 300 epochs with a batch-size of 512, a weight-decay of 0.0, warmup-epochs of 0, and with an initial learning rate of 5e-4 for the cosine learning rate decay scheduler. Also, we do not train with any data augmentation such as Rand-Augment [2], random erasing [15], stochastic depth [4], CutMix [13], Mixup [14], and repeated augmentation [1, 3]. However, we still use Random Resize Cropping and horizontal flipping. For knowledge distillation, we use a full-precision DeiT-S [10] as our teacher model. Our optimizer and scheduler are from the Timm library [11] and our training loop/pipeline is based on the DeiT repository [10].

3. Calculation of representational capability

Following [7, 12], we quantify the element-wise representational capability as the number of possible absolute values that each element in a tensor can have. We calculate the element-wise representational capability of these models by calculating the element-wise representational capability of the tensor that will be the input for the classifier layer or the final layer, following the steps from [7, 12] and we assume the first layer has binary weights only for simplicity.

3.1. Fully-binary DeiT-S

Following [7], for fully-binary DeiT-S, for the first layer, each output element can have a value range from [0, 195,840], since $P \cdot P \cdot C \cdot \max(x) = 16 \cdot 16 \cdot 3 \cdot 255 = 195,840$ where P is the patch size, C is the number of channels, and $\max(x) = 255$ is the max value an input image can hold. If we want to have a zero mean, then the range would be [-97,920, 97,920]. Therefore, the current representational capability at that point is 97,920. From [7], nor-

malization wouldn't affect the representational capability, since normalization is just an element-wise affine transformation. For each binary fully connected layer in the attention, for the query, key, value, and in the final projection layer, the element-wise representation capability for each of these layers is $D = 384$ which is the base hidden dimension for DeiT-S. For the matrix multiplication between the binarized attention probability and the binarized value matrix, the element-wise representation capability for this is $N = 196$ which is the sequence length for DeiT-S throughout the whole model. For the FFN, it would be $4 \cdot D = 4 \cdot 384 = 1,536$ for each binarized fully connected layer in the FFN. If we calculate the total element-wise representational capability of the fully-binary baseline ViT with DeiT-S backbone, $\mathbb{R}(\text{DeiT-S})$, it would be $(384 \cdot 4 + 196 + 4 \cdot 384 \cdot 2) \cdot 12 + 97,920 = 153,216$, considering the total number of fully connected layers per transformer block, and the total number of transformer blocks in DeiT-S.

3.2. Fully-binary ResNet-34

Following [7], for fully-binary ResNet-34, for the first layer, each output element can have a value range from [0, 37,485], since $K_1 \cdot K_1 \cdot C \cdot \max(x) = 7 \cdot 7 \cdot 3 \cdot 255 = 37,485$ where K_1 is the kernel size of the first convolutional layer. If we want to have a zero mean, then the range would be [-18,742, 18,742]. Therefore, the current representational capability at that point is 18,742. After the first convolutional layer, the feature map goes through max-pooling which has no effect on the element-wise representational capability of the model. ResNet-34 has 4 stages with each stage containing different feature map resolutions and hidden dimensions. For stage 1 in ResNet-34, each convolutional layer would have an element-wise representational capability of $3 \cdot 3 \cdot 64 = 576$, considering the kernel size and the hidden dimension of one weight filter. During the transition from stage 1 to stage 2, there will be an average pooling layer in the residual with a kernel size of 2×2 with a stride of 2×2, which can be seen as an information aggregation of 4 neighboring patches if we ignore the element-wise division involve in average pooling. Therefore, the element-wise representational capability would be multiplied by 4, so the current total element-wise representational capability of the model up to this point can be calculated as $(18,742 + 576 \cdot 3) \cdot 4 = 81,880$, considering the number of convolutional layers in that stage. For stage 2 in ResNet-34, each convolutional layer would have an element-wise representational capability of $3 \cdot 3 \cdot 128 = 1,152$. After the transition from stage 2 to stage 3, the current total element-wise representational capability of the model up to this point can be calculated as $(81,880 + 1,152 \cdot 4) \cdot 4 = 345,952$. For stage 3 in ResNet-34, each convolutional layer would have an element-wise representational capability of $3 \cdot 3 \cdot 256 = 2,304$. Therefore, after the transition from stage 3 to stage 4, the current total element-wise representational capability of the model up to this point can be calculated as $(345,952 + 2,304 \cdot 6) \cdot 4 = 1,439,104$. For stage 4 in ResNet-34, each convolutional layer would have an element-wise representational capability of $3 \cdot 3 \cdot 512 = 4608$. After the feature map is finished getting processed in stage 4, there will be a global average pooling layer that aggregates all the information from the remaining patches before getting processed at the final classifier layer. For a 224×224 input image, the feature map resolution at stage 4 of ResNet-34 will be 7×7, so the total element-wise representational capability of the fully binary ResNet-34, $\mathbb{R}(\text{ResNet-34})$, up until the final classifier layer, will be $(1,439,104 + 4,608 \cdot 3) \cdot 49 = 71,193,472$.

References

[1] Maxim Berman, Hervé Jégou, Vedaldi Andrea, Iasonas Kokkinos, and Matthijs Douze. MultiGrain: a unified image embedding for classes and instances. *arXiv e-prints*, Feb 2019. 1

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 1

[3] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8126–8135, 2020. 1

[4] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016. 1

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[6] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. ReActNet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision (ECCV)*, 2020. 1

[7] 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 *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018. 1, 2

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 1

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch:

An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[10] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers: distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021. 1

[11] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 1

[12] Yixing Xu, Xinghao Chen, and Yunhe Wang. BiMLP: Compact binary architectures for vision multi-layer perceptrons. In *Advances in Neural Information Processing Systems*, 2022. 1

[13] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 1

[14] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 1

[15] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 1