BinaryViT: Pushing Binary Vision Transformers Towards Convolutional Models

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

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

Abstract

With the increasing popularity and the increasing size of vision transformers (ViTs), there has been an increasing interest in making them more efficient and less computationally costly for deployment on edge devices with limited computing resources. Binarization can be used to help reduce the size of ViT models and their computational cost significantly, using popcount operations when the weights and the activations are in binary. However, ViTs suffer a larger performance drop when directly applying convolutional neural network (CNN) binarization methods or existing binarization methods to binarize ViTs compared to CNNs on datasets with a large number of classes such as ImageNet-1k. With extensive analysis, we find that binary vanilla ViTs such as DeiT miss out on a lot of key architectural properties that CNNs have that allow binary CNNs to have much higher representational capability than binary vanilla ViT. Therefore, we propose BinaryViT, in which inspired by the CNN architecture, we include operations from the CNN architecture into a pure ViT architecture to enrich the representational capability of a binary ViT without introducing convolutions. These include an average pooling layer instead of a token pooling layer, a block that contains multiple average pooling branches, an affine transformation right before the addition of each main residual connection, and a pyramid structure. Experimental results on the ImageNet-1k dataset show the effectiveness of these operations that allow a fully-binary pure ViT model to be competitive with previous state-of-the-art binary (SOTA) CNN models.

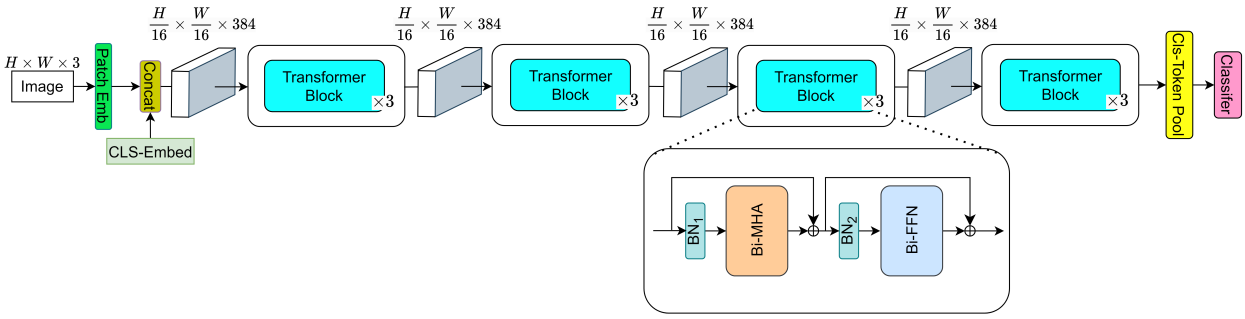
1. Introduction

Transformers [31] have attracted a lot of attention in natural language processing tasks such as BERT [6] and GPT [4]. Also, they are gaining a lot of attention in computer vision tasks [8, 29], since they have been able to outperform most CNNs when being pre-trained on large amounts of data with proper data augmentation and regularization.

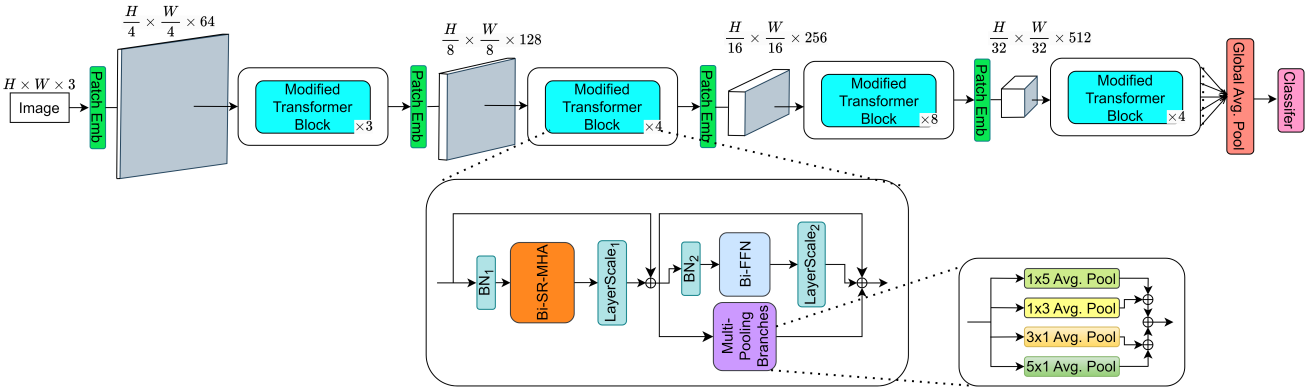
With the rising popularity and the increasing size of vision transformers (ViTs), there has been a rising interest in making them more efficient and less computationally costly to deploy them onto edge devices with limited resources such as smartphones, smart watches, etc. Therefore, model compression techniques such as quantization [15, 16, 21, 23, 39, 41], distillation [14, 23, 41], pruning [10, 27, 40], etc. have been actively studied to reduce the model size and computational cost of transformers. Among these compression methods, quantization can not only reduce the memory requirement of the model but can also replace expensive floating-point operations with simpler fixed-point operations. An extreme form of quantization is binarization. Binarization of weights and activations can facilitate the use of popcount operations to reduce the model size and reduce computational cost. However, matching the performance of a transformer with binary weights and activations with its full-precision counterpart is still a challenging task.

Previous works have shown that the drop in the performance can be mitigated with distillation methods from [23, 41] to encourage the binarized transformers to mimic the full-precision model. Also, a scaling factor can be applied to reduce the drop in the performance of DNNs due to binarization to minimize the quantization error [26]. The scaling factor can also be determined by setting it as a learnable parameter and training it to minimize the task loss [19]. For CNNs, [22] proposes Bi-RealNet a binary CNN that has extra residual connections to preserve more information to increase its representational capability to make it more accurate on tasks that have lots of classes like ImageNet-1k [5]. [20] proposes ReActNet which further improves upon the work Bi-RealNet [22] by adding a learnable threshold before the sign function and by adding the RPReLU activation function after each residual connection to help reshape the output distribution at near zero extra cost. For MLPs, BiMLP [36] tries to solve the limited representational capability of fully-connected layers by proposing the multi-branch block where they allow patch mixing and channel mixing to happen simultaneously.

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



(a) Baseline binary ViT with the DeiT-S structure



(b) Our BinaryViT with a global average pooling layer, Multi-Pooling Branches, LayerScale, and a pyramid structure

Figure 1. We start from the baseline binary ViT architecture in (a) and slowly change the architecture to BinaryViT in (b).

However, fully binarized pure ViTs suffer huge performance drop in tasks with large number of classes like ImageNet-1k such that they have lower performance than fully binarized CNNs when applying CNN binarization techniques or applying existing transformer/MLP-based binarization methods onto a pure vanilla ViT architecture since existing binarization methods do not go into details about how architectural or operational designs, besides additional extra residual connections, affect the performance of binary neural networks. BiMLP [36] only explores the limited representational capability of a binarized fully-connected layer itself and does not explore other architectural details that can affect the accuracy of a binary fully-connected-only-based model as this work does.

Therefore, in this work, after we design a baseline binary pure ViT using existing binarization techniques in Section 2, we analyze the differences between the architecture of a CNN model such as ResNet [11] and the architecture of a pure vanilla ViT such as DeiT. From our analysis in Section 3, we find that binary vanilla ViTs miss out on a lot of key architectural properties that CNNs have that allow CNNs to have much higher representational capability than binary vanilla ViTs. Therefore, we propose BinaryViT, in which inspired by the CNN architecture, we include operations from the CNN architecture into a pure ViT architecture in Section 3.1, 3.2, 3.3, and 3.4 to enrich the representational capability of a binary pure ViT without introducing convolutions and without significantly increasing the number of operations and the number of parameters. These include an average pooling layer instead of a token pooling layer to account for information from all tokens/patches; a block with multiple average pooling branches to compensate for the loss of the representational capability from a binarized fully-connected layer; an affine transformation right before the addition of each main residual branch to prevent the scale of each main residual branch from overwhelming the scale of each main branch; and a pyramid structure to allow binary features to be processed at higher resolution at the early stages without increasing the computational complexity.

To the best of our knowledge, these operations or architectural properties introduced in this work have already been explored in previous works relating to full-precision ViTs, but their effects on the accuracy of binary neural networks or on binary ViTs have not yet been explored. Also, to the best of our knowledge, with these operations, we are the first to outperform prior binary CNN models on the ImageNet-1k dataset in terms of accuracy and the number of operations (OPs), using a pure ViT architecture.

Method	Top-1 (%)
BinaryBERT [2]	1.4
BiBERT [23]	33.5
BiT [19]	45.7
+Remove FFN-Distill	46.5
+SGBERT [1]	47.4
+ReActNet [20]	48.5

Table 1. Top 1 accuracy on ImageNet with existing transformer-based binarization methods applied on the binarized DeiT-S.

2. Designing a fully binarized ViT baseline

First, we design a baseline binary pure ViT model using the DeiT [8, 29] backbone, since, to the best of our knowledge, there has never been works on a pure ViT model with binary weights and activations. In this section, we use existing binarization techniques applied on CNN, and on BERT [6] to design the baseline binary pure ViT model as shown in Figure 1a and 2a.

2.1. Binarized fully-connected layer

For the forward pass, for each matrix multiplication, a sign function is used for each input activation matrix, $\mathbf{X} \in \mathbb{R}^{N \times D_{in}}$, and weight matrix, $\mathbf{W} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$. A threshold vector, $\beta_{\mathbf{X}} \in \mathbb{R}^{D_{\text{in}}}$, can be applied to the real value inputs right before applying the sign function to allow these inputs to have some distributional shift. For the weights, the threshold, $\mu(\mathbf{W}) \in \mathbb{R}^{D_{\text{out}}}$, can be determined by computing the mean value of all elements inside the matrix as in [23–25]. For the activations, the threshold parameter can be determined using backpropagation to minimize the task loss as in [20, 37]. After applying the sign function, a scaling factor, $\alpha_{\mathbf{W}} \in \mathbb{R}$, where $\alpha_{\mathbf{W}} = \frac{1}{n}\|\mathbf{W}\|_1$, is applied as in [23]. Then, the matrix multiplication output, $\mathrm{Y}(\mathbf{X}) \in \mathbb{R}^{N \times D_{\text{out}}}$, can be calculated using popcount, $\otimes$, as

$$\mathrm{Y}(\mathbf{X}) = \alpha_{\mathbf{W}}\, \mathrm{Rsign}(\mathbf{X}) \otimes \mathrm{sign}(\mathbf{W} - \mu(\mathbf{W})) \tag{1}$$

where $\mathrm{Rsign} = \mathrm{sign}(\mathbf{X} + \beta_{\mathbf{X}})$ as in [20].

For each binary fully-connected layer, BiFC, in a binary transformer, we connect a residual connection, R, from the input, $\mathbf{X}$, to the output of the linear layer, $\mathrm{BN}(\mathrm{Y}(\mathbf{X}))$, as:

$$\mathrm{BiFC}(\mathbf{X}) = \mathrm{RPReLU}(\mathrm{BN}(\mathrm{Y}) + \mathrm{R}(\mathbf{X})) \tag{2}$$

to preserve the information from the previous layer as in [20, 22]. RPReLU activation function proposed by [20] is used after each residual connection. Also, we replace all layer normalization in the ViT model with batch normalization [13], BN, since all linear layers have a normalization layer after it, as in [38], to enable faster inference and also faster training compared to layer normalization. The residual function, R, from [20] can be an identity function if the input and output dimensions are the same. If the input dimension is smaller than the output channel by n times, the residual function will be the input concatenation with itself n times. If the input dimension is larger than the output channel by n times, the residual function will be the average pooling function with a stride of n and a window size of n. Therefore, $\mathrm{R}(\mathbf{X})$ can be defined as

$$\mathrm{R}(\mathbf{X}) = \begin{cases} \mathbf{X}, & c_{in} = c_{out} \\ \mathrm{Cat}([\mathbf{X}, \text{for i in range(n)}], \text{dim=2}), & nc_{in} = c_{out} \\ \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}(:,:, \frac{i-1}{n}d_{in} : \frac{i}{n}d_{in})) & c_{in} = nc_{out} \end{cases} \tag{3}$$

where Cat is the concatenation function.

Using a straight-through estimator [3], we approximate the derivative of sign with respect to an input as

$$\frac{\partial\, \mathrm{sign}(x)}{\partial x} \approx \begin{cases} 1, & |x| \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

2.2. Binarized vision transformer

A ViT uses N number of transformer encoder blocks and each transformer block contains one multi-head attention (MHA) module and one feed-forward network (FFN) module. Initially, the image, $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ gets split up into fixed-size patches, $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, in the embedding layer, where (H, W) is the image dimensions, C is the number of channels of the input image, (P, P) is the dimension of each image patch, and $N = HW/P^2$ is the number of patches. Then, a linear projection, $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$, is applied onto each of these patches before being appended with a cls-token embedding, $\mathbf{x}_{\text{class}} \in \mathbb{R}^D$, and summed with the position embeddings, $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$, as in Eq. (5)

$$\mathbf{H}_1 = [\mathbf{x}_{\text{class}};\ \mathbf{x}_p^1\mathbf{E};\ \mathbf{x}_p^2\mathbf{E};\ ...;\ \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{\text{pos}} \tag{5}$$

Like in previous works on binarization, the operations in the first layer are always kept in full precision.

The output of the embedding layer $\mathbf{H}_1 \in \mathbb{R}^{(N+1) \times D}$ then becomes the input for the first transformer block. In each transformer block, the input $\mathbf{H} \in \mathbb{R}^{(N+1) \times D}$ first gets

normalized by a pre-batch-normalization layer as $\hat{\mathbf{H}} = \text{BN}_1(\mathbf{H})$. Then, in the MHA module of the binarized transformer with N_H attention heads, the output of that batch-normalization is then used to calculate the query, $\mathbf{Q}_h \in \mathbb{R}^{(N+1)\times D_h}$; key, $\mathbf{K}_h \in \mathbb{R}^{(N+1)\times D_h}$; and value, $\mathbf{V}_h \in \mathbb{R}^{(N+1)\times D_h}$, for each head, h, as:

$$\begin{aligned}\mathbf{Q}_h &= \text{BiFC}_{\text{Q}_h}(\hat{\mathbf{H}}_h)\\ \mathbf{K}_h &= \text{BiFC}_{\text{K}_h}(\hat{\mathbf{H}}_h)\\ \mathbf{V}_h &= \text{BiFC}_{\text{V}_h}(\hat{\mathbf{H}}_h)\end{aligned} \tag{6}$$

using Eq. (1) and Eq. (2), where $D_h = D/N_H$. We then calculate the attention score as, $\mathbf{A}_h = \text{Rsign}(\mathbf{Q}_h)\text{Rsign}(\mathbf{K}_h^\top)$.

From [19], to get the binarized attention probability matrix of each head, $\mathbf{P}_h \in \mathbb{R}^{(N+1)\times(N+1)}$ we apply the softmax function on the attention score, $\mathbf{A}_h \in \mathbb{R}^{(N+1)\times(N+1)}$, and then apply the round-to-nearest-integer function, $\lceil\cdot\rfloor$,

$$\mathbf{P}_h = \alpha_{\mathbf{P}}\Big\lceil\sigma\Big(\frac{1}{\alpha_{\mathbf{P}}}\,\text{Softmax}\Big(\frac{\mathbf{A}_h}{\sqrt{D_h}}\Big), 0, 1\Big)\Big\rfloor \tag{7}$$

where $\alpha_{\mathbf{P}} \in \mathbb{R}$ is a learnable scaling factor trained using the method from [9] and the $\sigma(x, r_1, r_2)$ function keeps the output to be between r_1 and r_2. A boolean function from [23] can also be applied on the attention score, but in Table 1, we find that it performs 12% worse than applying the softmax function and then rounding the output.

To get the output of each head, $\text{head}_h \in \mathbb{R}^{(N+1)\times D_h}$, the binarized attention probability matrix, $\mathbf{P}_h$, will be multiplied by the binarized value matrix, $\bar{\mathbf{V}}_h = \text{Rsign}(\mathbf{V}_h)$. Also, to preserve the information of the query, the key, and the value, we also add the query, the key, and the value to the head output as shown below

$$\text{head}_h = \text{RPReLU}(\text{BN}_{\text{at}}(\mathbf{P}_h\bar{\mathbf{V}}_h) + \mathbf{Q}_h + \mathbf{K}_h + \mathbf{V}_h) \tag{8}$$

and as shown in Figure 2a.

The outputs from all heads are then concatenated with each other and used by the fully-connected layer, BiFC_{O}, to get the multi-head attention output. A main residual connection is then applied to the output of MHA as

$$\mathbf{F} = \text{BiFC}_{\text{O}}(\text{Cat}(\text{head}_1, ..., \text{head}_{N_H})) + \mathbf{H} \tag{9}$$

Then the residual output, $\mathbf{F} \in \mathbb{R}^{(N+1)\times D}$, gets normalized by a 2nd pre-batch-normalization layer, BN_2, as $\hat{\mathbf{F}} = \text{BN}_2(\mathbf{F})$ and goes through the binarized feed-forward network layer, BiFFN, which has two binary fully-connected linear (BiFC) layers. Finally, a second main residual connection is then applied to the FFN output to get $\mathbf{R}$

$$\mathbf{R} = \text{BiFFN}(\hat{\mathbf{F}}) + \mathbf{F} \tag{10}$$

which gets inputted to the next transformer block.

Model	$\mathbb{R}(...)$
DeiT-S	153,216
ResNet-34	71,193,472

Table 2. Element-wise representational capability $\mathbb{R}(...)$ for the fully-binary DeiT-S and for the fully-binary ResNet-34.

Like in previous works in binarization, the parameter biases, the parameters at the classifier, operations in the softmax, and normalization layers are kept in full precision.

To improve the performance of the binarized ViT, we distill the knowledge of a full precision model to a model with binary weights and activations by minimizing the soft cross-entropy loss between the student's logit and the teacher's logit as in [20]. We do not use distillation loss for the attention scores and output as the performance will significantly degrade just as shown in [19, 23]. Also, we do not distill the FFN outputs from [2, 19, 41] as it causes a slight loss of accuracy by 1.2%. Using the partially-random-initialization method from [1] where we initialize the first patch embedding layer from the full precision model, improves the performance by 0.9%.

3. What else do binary CNNs have that binary transformers do not have?

In Table 1, using all of the aforementioned techniques in Section 2 will only get us 48.5% top-1 accuracy on ImageNet-1k which is far below the accuracy for most SOTA binary CNNs. Therefore, in this section, we further analyze the details/properties of the CNN architecture that have not been explored in the context of binary neural networks and that can help pure ViTs with binary weights and binary activations improve their representational capability to improve their accuracy on a dataset with a large number of classes without introducing convolutions and without significantly increasing the number of operations and the number of parameters. From Table 4, even increasing the number of parameters by either increasing the width or depth, it has trouble surpassing SOTA binary CNN models.

Analysis of representational capability. First, let's analyze and compare the element-wise representational capability of a binary CNN such as a fully-binary ResNet34, $\mathbb{R}$(ResNet-34) versus a binary ViT such as a fully-binary DeiT-S, $\mathbb{R}$(DeiT-S) with both of these models using the ReActNet [20] design. As in [22, 36], we quantify the element-wise representational capability as the number of possible absolute values that each element in a matrix/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, following the steps from [22, 36].

Global Average Pooling	Multi-Pooling Branches	LayerScale	Pyramid Structure	ImageNet Top-1 (%)
✗	✗	✗	✗	48.5
✓	✗	✗	✗	56.4
✓	✓	✗	✗	60.2
✓	✓	✓	✗	61.8
✓	✓	✓	✓	67.7

Table 3. Results of introducing different operations to the binary ViT model.

From Table 2, for fully-binary DeiT-S, we calculate the element-wise representational capability to be $\mathbb{R}(\text{DeiT-S}) = 153,216$, whereas for fully-binary ResNet-34, we calculate the element-wise representational capability to be $\mathbb{R}(\text{ResNet-34}) = 71,193,472$ which is the order of magnitudes greater than the fully binary DeiT-S. We believe that the representational capability gap leads to the performance gap between binary ResNet-34 and binary ViT.

Calculating the representational capability. A detailed explanation of how the element-wise representational capability was calculated for binary DeiT-S and for binary ResNet-34 can be found in the Appendix. Generally, from [22, 36], one single binary linear layer contributes $D_{in} \cdot K^2$ to the element-wise representational capability of an output tensor, where D_{in} is the input dimension, K is the kernel size of the square-shaped weight filter. For an average pooling layer with a kernel size of 2×2 and a stride of 2×2, the element-wise representational capability of an output tensor would be equal to the element-wise representational capability of an input tensor multiplied by 4 such that $\mathbb{R}(\text{out}) = 4 \times \mathbb{R}(\text{in})$, since that average pooling layer can be seen as an information aggregation of 4 neighboring patches if we ignore the element-wise division involved in average pooling. For the global average pooling layer, the element-wise representational capability of an output tensor would be equal to the element-wise representational capability of an input tensor multiplied by the number of patches/tokens of that input tensor such that $\mathbb{R}(\text{out}) = N \times \mathbb{R}(\text{in})$, where N is the number of tokens/patches since the global average pooling layer can be seen as an information aggregation of all patches/token. For affine transformation such as batch-normalization [13], we ignore its effect on the representational capability according to [22].

Increasing the representational capability. To increase the representational capability of ViT architecture and achieve better binary ViT performance, we proposed three designs. (1) Adding global average pooling before the classifier layer. (2) Adding multiple average pooling branches (3) Bringing Pyramid structure from CNN to ViT. Also, we borrow from the successful design of ResNet and MobileNet and placed an affine transformation before the residual branch to prevent the scale of each main residual branch from overwhelming the scale of each main branch such as the MHA output and the FFN output. With these designs added onto the binary ViT, we name the resulting binary ViT, BinaryViT. While trying to improve the ViT with binary weights and activations, we try to keep the number of parameters and the number of OPs to be around the same as this baseline binary ViT or to be around the same as a ReActNet ResNet-34 [20]. We define our baseline ViT as the DeiT-S [29] architecture with the applied methods mentioned in Section 2. This baseline has 22.1M parameters and 1.29×10^8 operations (OPs).

3.1. Global average pooling before classifier layer

We noticed that most binary CNNs, such as binary ResNet-34 [11] and binary MobileNet [12] have an average pooling layer at right before the classifier fully-connected layer, whereas the pure vanilla ViT models such as DeiT [8, 29] have a single cls-token pooling layer before the classifier layer. Using a single cls-token pooling layer prevents the information from all tokens from being taken into account. Knowing from [22] that each output token right before the classifier layer has a very limited representational capability for binary networks compared to full-precision networks, it would be useful to have information from all tokens being taken into account through global average pooling so that the final classifier layer has more flexibility in adjusting its output during training. After the introduction of the global average pooling into the model right before the classifier, the total element-wise representational capability can be increased by up to $196\times$ since there are 196 patches/tokens in the DeiT-S throughout the whole model, increasing the total element-wise representational capability to $153,216 \cdot 196 = 30,030,336$.

We also remove the cls-token embedding from the model such that in our binary ViT the output from the first embedding layer, $\mathbf{H}_1 \in \mathbb{R}^{N \times D}$ is changed from Eq. (5) to

$$\mathbf{H}_1 = [\mathbf{x}_p^1\mathbf{E};\ \mathbf{x}_p^2\mathbf{E};\ ...;\ \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{\text{pos}} \tag{11}$$

where the position embedding is now $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{N \times D}$.

From Table 3, after replacing cls-token pooling with average pooling as shown in Figure 1b, we received an increase in the performance from 48.5% to 56.4% top-1 on

ImageNet-1k. Note that from [34], average pooling does not change the performance by a lot for a full-precision ViT. Also, the impact of the average pooling right before the final layer on the number of OPs is negligible.

3.2. More branches

We find that a binary convolution has more representational power than a binary fully connected layer if the number of parameters for the binary convolution and for the binary fully-connected layer are the same. Let's consider a fully-connected layer with a binary weight matrix of size 384×384 and a binary convolution weight filter of size $3 \times 3 \times 128 \times 128$. For that binary fully connected layer, the output tensor will have an element-wise representational capability of 384 whereas, for that binary convolutional layer, that output tensor will have an element-wise representational capability of 1152. As we can see, the element-wise representational capability of that binary convolutional layer is $3\times$ larger than the element-wise representational capability of that binary fully connected layer.

This huge element-wise representational capability from the binary convolutional layer allows it to be more flexible in adjusting its output. As a result of this huge output element-wise representational capability per layer, more information will be carried to the final layer through the residual connections, allowing the binary CNNs to adjust their output during training more easily compared to a binary ViT.

Therefore, we decided to add 4 branches right beside each FFN block as shown in Figure 1b. The 4 branch contains: an average pooling layer of kernel size 1×3, an average pooling layer of kernel size 3×1, an average pooling layer of kernel size 1×5, and an average pooling layer of kernel size 5×1. The average pooling branches that have a kernel size of $1 \times K$ apply the filter along the feature map width, whereas the average pooling branches that have a kernel size of $K \times 1$ apply the filter along the feature map height. This branch design is not necessarily optimal and how to better design the multi-branch structure to have a better trade-off between the number of operations and accuracy remains an open question.

The multi-branch structure increased the performance of the binary ViT from 56.4% to 60.2% in Table 3. Also, the number of OPs increased from 1.29×10^8 to 1.32×10^8, but the number of OPs for the current binary ViT is still below a ReActNet ResNet-34.

3.3. Scaling right before residual connection

We find that networks tested for binarization problems such as ResNet [11] and MobileNet [12] tend to have a batch-normalization layer [13] right before being added by a residual connection. We are not sure if that batch-normalization placement helps the signal that is propagating through the residual to be normalized as in [17] or if the affine transformation property of batch normalization prevents the signal of the main branch to be over-consumed by the residual branch as in [30, 35].

In [35], they showed that with pre-norm architectures like most ViTs, the scale of hidden states grows as we go deeper into the layers of these pre-norm models such that

$$(1 + \tfrac{l}{2})D \leq \mathbb{E}(||\mathbf{H}_l||_2^2) \leq (1 + \tfrac{3l}{2})D$$

where l is the layer index. Therefore, even though we mentioned that an element-wise affine transformation should have no effect on a binary model's representational capability, there is a possibility that the information from any main branches in the deeper layer area could be over-consumed by a residual branch, preventing information from any main branches in the deeper layer area to be passed down in the residual connection, such that the model's representational capability would be lower than what we would expect.

To test this, we test 3 configurations: one with the residual-post-norm connection (res-post-norm) as in ResNets; one with the residual-post-norm connection plus the pre-norm connection (sandwich-configuration [7]); and one with the pre-norm connection plus LayerScale [30].

From our experiments, using the res-post-norm configuration gets us to 61.4% top-1, using the sandwich configuration gets us to 61.8%, and using the third configuration gets us to 61.8%. However, having any kind of affine transformation right before the residual connection is better than no affine transformation at all. Using the third configuration, we get increased the performance of the binary ViT from 60.2% to 61.8% as shown in Table 3. Therefore, for each main residual connection in the transformer in the attention and in the FFN, its main branch would contain an affine transformation such that in our binary ViT Eq. (9) and Eq. (10) are changed to

$$\mathbf{F} = \alpha_1 \odot \text{BiFC}_\text{O}(\text{Cat}(\text{head}_1, .., \text{head}_{N_H})) + \beta_1 + \mathbf{H} \quad (12)$$

$$\mathbf{R} = \alpha_2 \odot \text{BiFFN}(\hat{\mathbf{F}}) + \beta_2 + \mathbf{F} \quad (13)$$

where $\alpha_1, \alpha_2 \in \mathbb{R}^D$ are scaling factors and $\beta_1, \beta_2 \in \mathbb{R}^D$ are the bias terms.

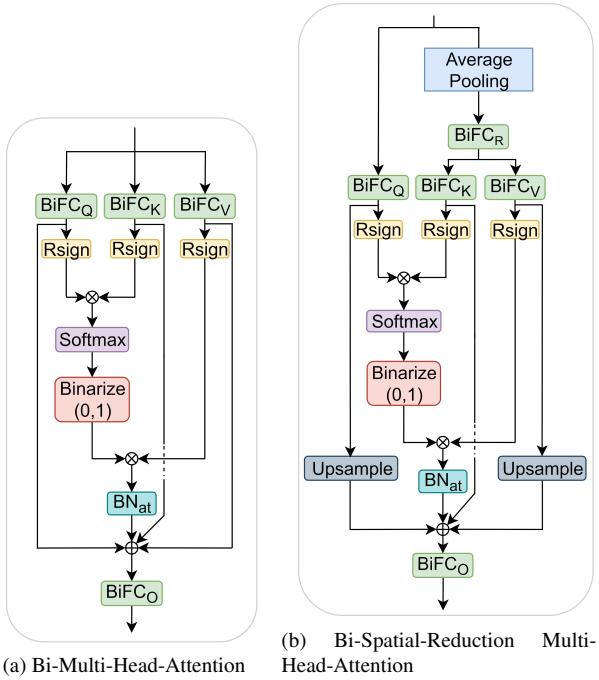
From [30, 34], the gain in accuracy for using res-post-norm or LayerScale for a full-precision ViT is very small compared to the gain in accuracy we get for a binary ViT.

3.4. Pyramid structure

Current SOTA binary CNN backbones such as ResNet [11] and MobileNet [12] have a pyramid structure where the feature map size progressively decreases from a high resolution to low resolution and the hidden dimension size progressively increases. Also, BiMLP with a Wave-MLP [28]

Methods	Model backbone	Global Average Pooling	Multi Branches	Pyramid Stucture	FLOPs ($\times 10^8$)	OPs ($\times 10^8$)	BOPs ($\times 10^9$)	Params ($\times 10^6$)	Top-1 Acc (%)
ReActNet [20]	ResNet-34	✓	✓	✓	1.39	1.93	3.53	21.8	67.5
ReActNet-B [20]	MobileNet	✓	✓	✓	0.44	1.63	4.69	29.3	70.1
BiMLP-S [36]	Wave-MLP-T	✓	✓	✓	1.21	1.56	**2.25**	**17.0**	70.0
Baseline-S	DeiT-S	✗	✗	✗	0.57	1.29	4.51	22.1	48.5
Baseline-S×1.5	DeiT-S×1.5	✗	✗	✗	0.57	1.70	7.15	34.9	48.7
Baseline-B	DeiT-B	✗	✗	✗	1.15	3.86	17.4	86.4	60.5
BinaryViT	-	✓	✓	✓	**0.19**	**0.79**	3.83	22.6	67.7
BinaryViT*	-	✓	✓	✓	0.95	1.54	3.75	22.6	**70.6**

Table 4. Comparing results of BinaryViT model with other SOTA binary models with different architecture backbones. * denotes that the patch embedding layers in the middle are in full precision. Floating-point operations (FLOPs) is the number of operations on full-precision layers, bit-operations (BOPs) is the number of operations on binarized layers, and from [20], OPs $=$ BOPs$/64 +$ FLOPs.



(a) Bi-Multi-Head-Attention

(b) Bi-Spatial-Reduction Multi-Head-Attention

Figure 2. Normal Bi-Multi-Head-Attention (Bi-MHA) (a) versus Bi-Spatial-Reduction Multi-Head-Attention (Bi-SR-MHA) (b).

backbone uses a pyramid structure, in which all SOTA binary neural networks with good performance have already, but previous works in binary neural networks do not explore the pyramid structure's effect on its performance.

However, we found that the pyramid structure can improve the element-wise representational capability of binary neural networks. Let's consider an example of a network containing 4 stages where each stage contains a different feature map resolution and base hidden dimension like the ResNet model [11], PVT [32] model, or the Swin [18] transformer. One binary fully connected layer with a hidden dimension of 64 in the first stage can contribute up to $64 \cdot 4 \cdot 4 \cdot 4 \cdot 49 = 200704$ of the element-wise representational capability, considering the downsampling layers for each transition between stages and the global average pooling layer right before the classifier layer. We can see that this binarized fully-connected layer with a low hidden dimension that is applied on a high-resolution feature map can contribute much more to the element-wise representational capability of a binary model than each binarized fully-connected layer with a high hidden dimension and a low feature-map resolution such as a binarized DeiT-S that has a global average pooling layer, hidden dimension of 384, and a feature map resolution of 14×14 in which each of its binarized fully-connected layer contributes 75,264 to the model's representational capability. Therefore, we find the pyramid structure can be useful to increase the element-wise representational capability of the model without increasing the computational complexity.

We design the pyramid architecture to be similar to PVT-S and/or ResNet-34. Our pyramid architecture contains 4 stages with base channel dimensions of 64-128-256-512 as in Figure 1b. The first stage contains 3 identical transformer blocks, the second stage contains 4 transformer blocks, the third stage contains 8 transformer blocks, and the last stage contains 4 transformer blocks. More details on the pyramid architecture can be found in the Appendix. We designed the pyramid architecture in this way such that we still match the number of parameters of the DeiT-S model [29] and for the ReAcNet method [20] 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 [20], since 320 is not divisible by 128.

For the first and second stages, applying attention to such a large sequence size of 3136 and 784, respectively, is very computationally costly. To remedy this, we apply a down-

sampling layer on the input right before the calculation of the key, and value matrix in the self-attention as in the PVT-v2 model [33] and as shown in Figure 2b. Therefore, Eq. (6) for the key and value would be changed to

$$\begin{aligned}\mathbf{K}_h &= \mathrm{BiFC}_{\mathrm{K}_h}(\mathrm{BiFC}_{\mathrm{R}}(\mathrm{AvgPool}(\hat{\mathbf{H}}))_h) \\ \mathbf{V}_h &= \mathrm{BiFC}_{\mathrm{V}_h}(\mathrm{BiFC}_{\mathrm{R}}(\mathrm{AvgPool}(\hat{\mathbf{H}}))_h)\end{aligned} \tag{14}$$

where AvgPool is an average pooling function with a kernel size of R and a stride of R and $\mathrm{BiFC}_{\mathrm{R}}$ is an additional binary linear projection after the average pooling function. Also the calculation of each head from Eq. (8) would be changed to

$$\mathrm{head}_h = \mathrm{RPReLU}(\mathrm{BN}_{\mathrm{at}}(\mathbf{P}_h\bar{\mathbf{V}}_h) + \mathbf{Q}_h + \mathrm{N}(\mathbf{K}_h) + \mathrm{N}(\mathbf{V}_h)) \tag{15}$$

where N is the nearest-neighbor interpolation function used to resize the resolution of the key and value feature back to its original resolution after it was downsampled in Eq. (14).

The accuracy gain on ImageNet [5] from introducing a pyramid structure into a full-precision ViT model as in PVT [32] is negligible. However, from Table 3, the pyramid structure increased the performance of the binary ViT from 61.8% to 67.7%. Also, the number of OPs decreased from 1.32×10^8 to 0.79×10^8.

4. Experiments

We evaluate our BinaryViT on the ImageNet [5] dataset which contains around 1.2M training images and 50,000 validation images from 1,000 classes. We evaluate our method on ImageNet because most of the SOTA binary CNNs works have been evaluated on that dataset. Our training procedure is described in the Appendix.

4.1. Experimental results

We compare BinaryViT to other SOTA binary models with different architectural backbones such as ReActNet [20] and BiMLP [36]. From Table 4, if we let the downsampling layers in the patch embedding be run in full precision, our BinaryViT can achieve a competitive performance of 70.6% top-1 accuracy while having less number of OPs and FLOPs compared to ReActNet CNNs. BinaryViT can achieve comparable performance with the MobileNet version of the ReActNet-B with having less parameters and less number of OPs. The MobileNet version of ReActNet does not use depth-wise convolutions like the original MobileNet, so it'll have 29.3M parameters.

Also, BinaryViT can outperform the ResNet-34 version of the ReAcNet, with having less number of FLOPs, having less number of OPs, and having approximately the same number of parameters. If the patch embeddings in the downsampling layers are binarized, it can outperform the ReAcNet ResNet-34, with having around 7× less number of FLOPs and having around 2× less number of OPs.

Comparing our BinaryViT with BiMLP-S [36] which uses a WaveMLP-T [28] architecture as its backbone, we can achieve competitive performance while having a lower number of FLOPs and having a lower number of OPs. This is due to the fact that our BinaryViT does not use an overlapping convolutional layer in each patch embedding layer which significantly increases the number of FLOPs like in WaveMLP [28] or in BiMLP. Also, the BiMLP is able to achieve competitive performance by just increasing the number of branches, because its backbone architecture, WaveMLP-T, already contains some operations from the CNN architecture, such as a global average pooling and a pyramid structure in which [36] is unaware of and hasn't explored their importance yet.

Overall, from Table 4, we can see that it is very important to have a global average pooling layer, some sort of multi branches layer, and a pyramid structure in which all of the SOTA binary models have in their architecture backbone to have a competitive accuracy.

4.2. Benefit of modifying the architecture

Model	DeiT-S	BinaryViT
FP32	79.9	79.9
1-bit	48.5	70.6

Table 5. Comparisons of the original and our proposed architectures on FP32 and binary settings on ImageNet-1k top-1 accuracy.

Table 5 reports the accuracy gaps between the FP32 and the binary version of the vanilla ViT such as DeiT-S and of the modified ViT. In FP32 setting, the modified ViT achieves roughly the same performance as the DeiT-S. However, in binary setting, our proposed BinaryViT architecture outperforms the DeiT-S significantly, which justifies the effectiveness of the proposed BinaryViT architecture.

5. Conclusion

In this work, we point out that binary vanilla ViTs with backbone architectures such as DeiT miss out on a lot of key architectural properties that CNNs have that allow binary CNNs to have much higher representational capability than binary vanilla ViTs. Thus, we introduce some operations from the CNN architecture into a pure ViT architecture to increase representational capability without the use of convolutions. These include an average pooling layer instead of a token pooling layer, a novel block that contains multiple average pooling branches, an affine transformation right before the addition of each main residual connection, and a pyramid structure. Experimental results on the ImageNet-1k dataset show the effectiveness of the proposed operations to outperform prior binary CNNs.

References

[1] Arash Ardakani. Partially-random initialization: A smoking gun for binarization hypothesis of BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2603–2612, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. 3, 4

[2] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online, Aug. 2021. Association for Computational Linguistics. 3, 4

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1, 8

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 1, 3

[7] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. CogView: Mastering text-to-image generation via transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19822–19835. Curran Associates, Inc., 2021. 6

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1, 3, 5

[9] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020. 4

[10] Mitchell Gordon, Kevin Duh, and Nicholas Andrews. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online, July 2020. Association for Computational Linguistics. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 5, 6, 7

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5, 6

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 3, 5, 6

[14] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online, Nov. 2020. Association for Computational Linguistics. 1

[15] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-ViT: Accurate and fully quantized low-bit vision transformer. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 1

[16] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. FQ-ViT: Post-training quantization for fully quantized vision transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 1173–1179, 2022. 1

[17] 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. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 7

[19] Zechun Liu, Barlas Oguz, Aasish Pappu, Lin Xiao, Scott Yih, Meng Li, Raghuraman Krishnamoorthi, and Yashar Mehdad. BiT: Robustly binarized multi-distilled transformer. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave,

and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 1, 3, 4

[20] 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, 3, 4, 5, 7, 8

[21] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 1

[22] 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, 3, 4, 5

[23] Haotong Qin, Yifu Ding, Mingyuan Zhang, Qinghua Yan, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. BiBERT: Accurate fully binarized bert. In *International Conference on Learning Representations (ICLR)*, 2022. 1, 3, 4

[24] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *IEEE CVPR*, 2020. 3

[25] Haotong Qin, Xiangguo Zhang, Ruihao Gong, Yifu Ding, Yi Xu, and Xianglong Liu. Distribution-sensitive information retention for accurate binary neural network. *International Journal of Computer Vision*, pages 1–22, 2022. 3

[26] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing. 1

[27] Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020. 1

[28] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Yanxi Li, Chao Xu, and Yunhe Wang. An image patch is a wave: Phase-aware vision mlp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10935–10944, 2022. 6, 8

[29] 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, 3, 5, 7

[30] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021. 6

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[32] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, October 2021. 7, 8

[33] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. PVT v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):1–10, 2022. 8

[34] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[35] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020. 6

[36] 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, 2, 4, 5, 7, 8

[37] Zhe Xu and Ray C. C. Cheung. Accurate and compact convolutional neural networks with trained binarization. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 19. BMVA Press, 2019. 3

[38] Zhuliang Yao, Yue Cao, Yutong Lin, Ze Liu, Zheng Zhang, and Han Hu. Leveraging batch normalization for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 413–422, 2021. 3

[39] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8BERT: Quantized 8bit BERT. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pages 36–39, 2019. 1

[40] Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754*, 2021. 1

[41] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. TernaryBERT: Distillation-aware ultra-low bit BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 509–521, Online, Nov. 2020. Association for Computational Linguistics. 1, 4