BHViT: Binarized Hybrid Vision Transformer

Supplementary Material

1. Hybrid Vision Transformer

Recent advancements in ViT architectures have explored the integration of convolutional layers, departing from the original design that relied solely on fully connected layers for processing. Notably, works like Pyramid Vision Transformer (PVT) [14] and FastViT [12] have introduced convolutional layers into the ViT model, leading to enhanced model performance and capabilities. Moreover, according to Meta-former [15], the outstanding performance of ViT is attributed more to its architectural characteristics rather than the introduction of self-attention modules. This finding further reinforces the effectiveness of the hybrid ViT architecture. Therefore, when designing model architectures based on ViT, using different structures as alternatives to the self-attention module for token mixing in specific scenarios is possible. The computational complexity of the attention matrix demonstrates quadratic growth concerning the number of tokens, and the acquisition of a binarized attention matrix introduces notable computational redundancy. Therefore, when dealing with a large number of tokens, replacing the attention module with specialized convolutional structures can reduce computational complexity and decrease the corresponding number of parameters, which is an effective solution to address the issue caused by an excessive number of tokens.

2. The detailed illustration of three observations

Observation 1. *Avoiding excessive numbers of tokens is beneficial for Binary ViT.*

Detailed illustration. For a vector $\mathbf{x}$ containing k elements, $[x_1, x_2, \cdots, x_k]$, represents as one row of the attention matrix before softmax, which is the similarity vector between a token and the rest of the tokens.

As shown in Bibert [10], we assume the $\mathbf{x}$ is the m row of the attention matrix before softmax, and the element of $\mathbf{x}$ can be obtained by the following,

$$\begin{aligned} x_i &= \sum_{l=1}^{d} B_a\left(\mathbf{Q}, a_1, b_1\right)^{m,l} \times B_a\left(\mathbf{K}^T, a_2, b_2\right)^{l,i}, \\ \mathrm{B}_a\left(\mathbf{M}, a, b\right) &= sign\left(\frac{\mathbf{M}-b}{a}\right), \end{aligned} \tag{1}$$

where $\mathrm{B}_a\left(\mathbf{M}, a, b\right)$ is the binary process of $\mathbf{M}$. a and b are scale factor and bias, respectively. l and d are the index and number of channels of $\mathbf{Q}$ and $\mathbf{K}$, respectively.

Let $\gamma = B_a\left(\mathbf{Q}, a_1, b_1\right)^{m,l} \times B_a\left(\mathbf{K}^T, a_2, b_2\right)^{l,i}$, thus γ is a binary random variable taking 1 or -1, which is subject to a Bernoulli distribution with the probability of p (when $\gamma = 1$). Based on the binary process in Eq. 2, p near 0.5. Then, the probability of x_i, p_{x_i}, can be expressed as a binomial distribution.

$$p_{x_i}\left(x_i\right) = C_d^t p^t \left(1-p\right)^{d-t} \tag{2}$$

where x_i takes the value of $2t-d$, referring to Fig. 1. Following the DeMoivre–Laplace theorem [13], x_i can be well approximated by the normal distribution $\mathcal{N}\left(\mu, \sigma^2\right)$ when d is large enough, shown in Fig. 2. In our case, d is no less than 256 (the number of channels), and the DeMoivre–Laplace theorem can be applicable very well. As the information entropy of one-dimensional Gaussian distribution is

$$\begin{aligned} H_G(x) &= -\int_{-\infty}^{+\infty} p_G\left(x\right) \ln\left(p_G\left(x\right)\right) dx \\ &= -\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \cdot \ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \mathrm{d}x \\ &= -\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \cdot (-\ln\sqrt{2\pi\sigma^2} - \frac{(x-\mu)^2}{2\sigma^2}) \mathrm{d}x \\ &= \ln\sqrt{2\pi\sigma^2} + \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \cdot \frac{(x-\mu)^2}{2\sigma^2} \mathrm{d}x \\ &= \ln\sqrt{2\pi\sigma^2} + \frac{1}{\sqrt{\pi}} \int_{-\infty}^{+\infty} e^{-\rho^2} \cdot \rho^2 d\rho \\ &= \ln\sqrt{2\pi\sigma^2} + \frac{1}{2} \\ &= \frac{1}{2}\ln\left(2\pi e\sigma^2\right), \end{aligned} \tag{3}$$

where $p_G\left(x\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. Assuming the elements of $\mathbf{x}$ are independent and identically distributed, the information entropy of $\mathbf{x}$ is

$$H_G\left(\mathbf{x}, k\right) = \frac{k}{2}\ln\left(2\pi e\sigma^2\right). \tag{4}$$

From Eq. 4, we can find the information entropy of $\mathbf{x}$ is proportional to the token number k. Therefore, as the number

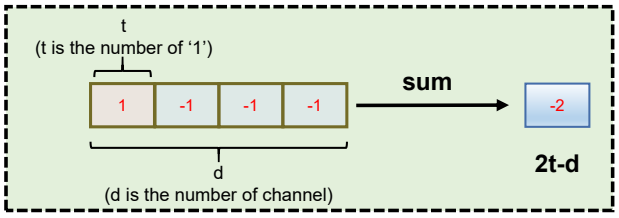

Figure 1. The Schematic diagram of the process of computing x_i, referring to Eq. 2.

of tokens k increases, the information entropy of $\mathbf{x}$ would continuously increase.

The *softmax* operation transfers vector $\mathbf{x}$ to the normalized vector $\mathbf{p}_{sof}$, as shown in Eq. 5,

$$\mathbf{p}_{sof}^{i} = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}}, \tag{5}$$

where e^{x_i} means exponential mapping for x_i. $\mathbf{p}_{sof}^{i}$ represent the probabilities of the corresponding element x_i in $\mathbf{x}$.

The entropy of the vector $\mathbf{x}$ after *softmax* is represented by

$$\begin{aligned} H_s(\mathbf{x},k) &= -\sum_{i=1}^{k} \mathbf{p}_{sof}^{i} \ln(\mathbf{p}_{sof}^{i}) \\ &= -\sum_{i=1}^{k} \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \ln\left(\frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}\right) \\ &= -\sum_{i=1}^{k} \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \left(x_i - \ln\left(\sum_{j=1}^{k} e^{x_j}\right)\right) \\ &= \ln\left(\sum_{j=1}^{k} e^{x_j}\right) - \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}, \\ &= \ln\left(k \times \frac{1}{k} \sum_{j=1}^{k} e^{x_j}\right) - \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}, \\ &= \ln(k) + \ln\left(\frac{1}{k} \sum_{j=1}^{k} e^{x_j}\right) - \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}, \end{aligned} \tag{6}$$

where $\frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}$ is the expectation value of vector $\mathbf{x}$. $\frac{1}{k}\sum_{j=1}^{k} e^{x_j}$ is the expectation of variable e^{x_j} and e^{x_j} follows the log-normal distribution, a continuous probability distribution of a random variable whose logarithm is normally distributed [1]. Therefore, we could obtain

$$\begin{aligned} H_s(\mathbf{x},k) &= \ln(k) + \ln\left(\frac{1}{k} \sum_{j=1}^{k} e^{x_j}\right) - \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}, \\ &= \ln(k) + \ln\left(e^{\mu + \frac{\sigma^2}{2}}\right) - \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}, \end{aligned} \tag{7}$$

where x_j and x_i follow the same Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Let $\mu_s = \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}}$, and we have

$$H_s(\mathbf{x},k) = \ln(k) + \mu + \frac{\sigma^2}{2} - \mu_s, \tag{8}$$

The μ_s is the weighted sum of x_i and the sum of weights is 1. As k increases, there is an upper bound on the value of μ_s.

$$\mu_s = \frac{\sum_{i=1}^{k} e^{x_i} \cdot x_i}{\sum_{j=1}^{k} e^{x_j}} < \frac{\sum_{i=1}^{k} e^{x_i} \cdot d}{\sum_{j=1}^{k} e^{x_j}} = d, \tag{9}$$

where d is the channel number of x_i. Combining the Eq. 8 and Eq. 9, the information entropy of $\mathbf{x}$ after *softmax* also increases with a larger k. Meanwhile, as k increases, we have

$$\begin{aligned} \lim_{k\to\infty} \mathbf{p}_{sof}^{i} &= \lim_{k\to\infty} \frac{e^{x_i}}{\sum_{j=1}^{k-1} e^{x_j} + e^{x_i}}, (i \neq j) \\ &\approx \frac{e^{x_i}}{(k-1)\, e^{x_i} + e^{x_i}} = \frac{1}{k}, \end{aligned} \tag{10}$$

where the difference between e^{x_i} and each e^{x_j} can be ignored when $k \to \infty$.

Therefore, the probability distribution vector of $\mathbf{x}$ gradually approximates a uniform distribution with an increasing number of tokens. An illustrative example is shown in Fig. 3. Three pictures describe different numbers of samples (20, 200, and 2000) from the same Gaussian distribution, respectively. From Fig. 3, it is observed that the distribution is gradually approximating uniform with the number of data increasing. It is well known that a uniform distribution for the attention matrix implies that all tokens are treated equally, which undermines the effectiveness of the attention mechanism.

From another perspective, as the number of tokens increases, the scaling factor a of the binary attention matrix may become too small. This is because $\mathbf{A}_{tt}$ becomes too small when the number of tokens is large, and thus a should be very small to make $\frac{\mathbf{A}_{tt}-b}{a}$ aligned to the range [0,1]. However, as shown in Eq. 11, a very small scale factor a reduces the value of the final binarization result during forward propagation and results in gradient disappearance during the back-propagation process.

$$\begin{aligned} &\textbf{Forward}: \mathrm{B}_{att}(\mathbf{A}_{tt}, a, b) = \\ &\qquad a \cdot clip\left(round\left(\frac{\mathbf{A}_{tt}-b}{a}\right), 0, 1\right), \\ &\textbf{Backward}: \frac{\partial L}{\partial \mathbf{A}_{tt}} = \begin{cases} a\frac{\partial L}{\partial \hat{\mathbf{A}}_{tt}} & b \leqslant \mathbf{A}_{tt} < a+b \\ 0 & otherwise \end{cases}, \end{aligned} \tag{11}$$

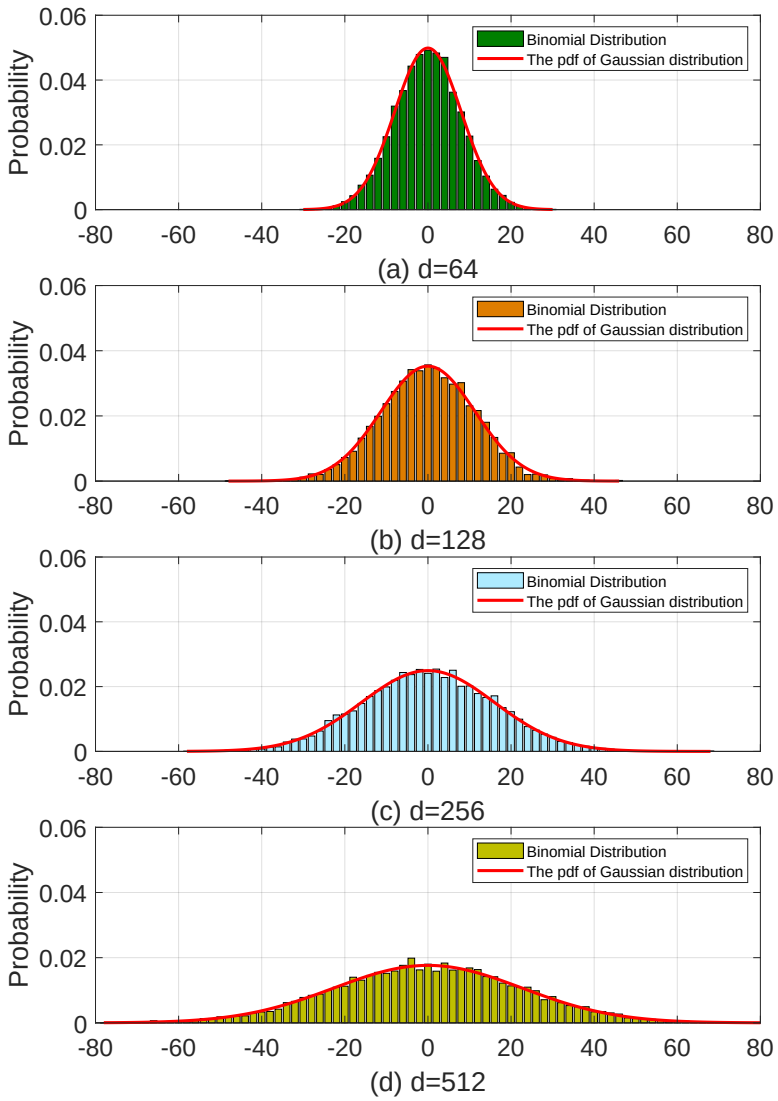

Figure 2. The Schematic diagram of DeMoivre–Laplace theorem with different d. The histograms display the Binomial distributions with the same $p = 0.5$ and different d. The red lines are the corresponding fitted Gaussian distributions. When d increases, the Binomial distribution can be better approximated by the Gaussian distribution.

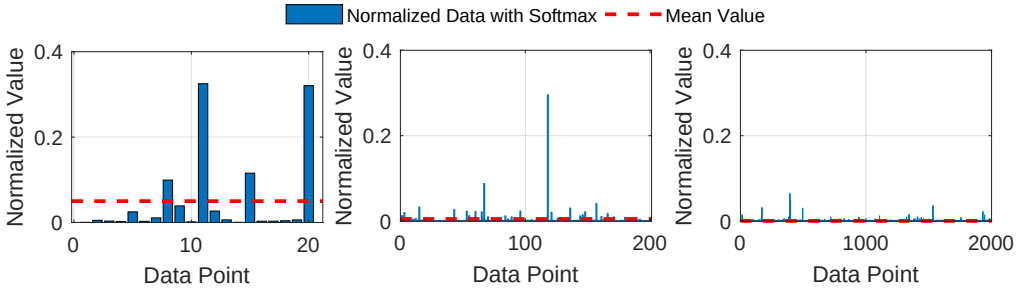

Figure 3. The distribution of the sample data sets with different numbers of data. Three sets are randomly sampled from the same Gaussian distribution. (a) 20 Sample data, (b) 200 Sample data, (c) 2000 Sample data.

where B_{att} represents the binary function for the full precision attention matrix $\mathbf{A}_{tt}$, and $\hat{\mathbf{A}}_{tt}$ denotes the corresponding binary attention matrix. $clip\left(x,0,1\right)$ truncates values that fall below 0 to 0 and those above 1 to 1, effectively ensuring that the output remains within the range [0, 1]. $round$ operation maps the input to the nearest integer.

To summarize the above observation, avoiding using many tokens for the binary attention module is advisable.

Observation 2. *Adding a residual connection in each binary layer is beneficial for binary ViT.*

Detailed illustration. Layer-by-layer residual connection refers to adding a residual connection for each binarization layer in a model. The essence is that applying layer-by-layer residual connections can effectively alleviate the disappearance of activation gradients caused by the continuous superposition of gradient truncation in multiple binary layers. Meanwhile, binarization functions inherently lead to information loss in activation values, and layer-by-layer residual connections help mitigate this information loss [7]. The current binary ViT algorithms [2, 6] only retain the residual connection outside the MLP and multi-head attention modules. Consequently, the gradient might not be fully exploited across all layers within each module of the binary ViT models.

Fig. 4 shows one attention module. in the gradient backpropagation of current binary ViT models [2, 6], the Jacobian of the output Y is calculated with respect to the weight of the linear layer. We take the weight W_q for the Q tensor as an example, as shown in Eq. 12, the element of Y is represented by Y^{n_l,c_i} and the element of W_q is represented by $W_q^{c_i,c_j}$, we have

$$\begin{aligned}\frac{\partial Y^{n_l,c_i}}{\partial {W_q}^{c_i,c_j}} = &\frac{\partial Y^{n_l,c_i}}{\partial B\left(A^{n_l,\boldsymbol{n}}\right)}\cdot\frac{\partial B\left(A^{n_l,\boldsymbol{n}}\right)}{\partial A^{n_l,\boldsymbol{n}}}\cdot\frac{\partial A^{n_l,\boldsymbol{n}}}{\partial M^{n_l,\boldsymbol{n}}}\cdot\\ &\frac{\partial M^{n_l,\boldsymbol{n}}}{\partial B\left(Q^{n_l,c_i}\right)}\cdot\frac{\partial B\left(Q^{n_l,c_i}\right)}{\partial Q^{n_l,c_i}}\cdot\\ &\frac{\partial Q^{n_l,c_i}}{\partial B\left(W_q^{c_i,c_j}\right)}\cdot\frac{\partial B\left(W_q^{c_i,c_j}\right)}{\partial W_q^{c_i,c_j}}\end{aligned} \tag{12}$$

where $\boldsymbol{n}\in\mathbb{R}^t, l\&k\in[1,t], i\&j\in[1,d]$. t means the token number and d is the channel number. We omit each activation's batch size and head dimension for simplicity of description. $B()$ means binarization function. M is the attention matrix before the *softmax* operation and $\sqrt{d}$ scaling process. A is the attention matrix after the *softmax* process.

Through the forward propagation path of the attention module, we can deduce the specific values of the gradients of each part of the chain rule. As shown in Eq. 13:

$$\begin{aligned}&\frac{\partial Y^{n_l,c_i}}{\partial B\left(A^{n_l,\boldsymbol{n}}\right)} = B\left(V^{\boldsymbol{n},c_i}\right),\\ &\frac{\partial B\left(A^{n_l,\boldsymbol{n}}\right)}{\partial A^{n_l,\boldsymbol{n}}} = \mathbf{1}_{0.5\leqslant A^{n_l,\boldsymbol{n}}\leqslant 1},\\ &\mathbf{1}_{0.5\leqslant A^{n_l,\boldsymbol{n}}\leqslant 1}^{n_l,n_k} = \begin{cases}1 & 0.5\leqslant A^{n_l,n_k}\leqslant 1\\ 0 & others\end{cases},\\ &\frac{\partial A^{n_l,\boldsymbol{n}}}{\partial M^{n_l,\boldsymbol{n}}} = \frac{A^{n_l,\boldsymbol{n}}\otimes\left(1-A^{n_l,\boldsymbol{n}}\right)}{\sqrt{d}},\\ &\frac{\partial M^{n_l,\boldsymbol{n}}}{\partial B\left(Q^{n_l,c_i}\right)} = \sum_{j=1}^{t} B\left(K^{c_i,n_j}\right),\\ &\frac{\partial B\left(Q^{n_l,c_i}\right)}{\partial Q^{n_l,c_i}} = \begin{cases}1 & |Q^{n_l,c_i}|\leqslant 1\\ 0 & others\end{cases},\\ &\frac{\partial Q^{n_l,c_i}}{\partial B\left(W_q^{c_i,c_j}\right)} = B\left(X^{n_l,c_j}\right),\\ &\frac{\partial B\left(W_q^{c_i,c_j}\right)}{\partial W_q^{c_i,c_j}} = \begin{cases}1 & |W_q^{c_i,c_j}|\leqslant 1\\ 0 & others\end{cases},\end{aligned} \tag{13}$$

where K is the K tensor and $B\left(X\right)$ is the binary input tensor of attention module. $\otimes$ denotes Hadamard product. So, we can get the specific expression of $\frac{\partial Y^{n_l,c_i}}{\partial {W_q}^{c_i,c_j}}$ as shwon in Eq. 14.

$$\begin{aligned}&\frac{\partial Y^{n_l,c_i}}{\partial {W_q}^{c_i,c_j}} = G\cdot\frac{\partial B\left(Q^{n_l,c_i}\right)}{\partial Q^{n_l,c_i}}\cdot B\left(X^{n_l,c_j}\right)\cdot\frac{\partial B\left(W_q^{c_i,c_j}\right)}{\partial W_q^{c_i,c_j}},\\ &G = \sum_{k=1}^{t}\left(B\left(V^{n_k,c_i}\right)^T\cdot 1_{0.5\leqslant A^{n_l,n_k}\leqslant 1}\cdot H_k\right),\\ &H_k = \frac{A^{n_l,n_k}\otimes\left(1-A^{n_l,n_k}\right)}{\sqrt{d}}\cdot B\left(K^{c_i,n_k}\right)\end{aligned} \tag{14}$$

As demonstrated in Eq. 14, the superposition of multiple binarized functions with large null range of the gradient results in vanishing gradient. To address this issue, similar to previous works [5, 7], we add a residual connection for each binary layer and attention module to avoid insufficient optimization caused by the vanishing gradients.

For example, as shown in Fig. 4, when we introduce a residual connection linking the Q tensor and the output Y, the gradient of the element Y^{n_l,c_i} with respect to the element $W_q^{c_i,c_j}$ is shown in Eq. 15. Due to the existence of the residual link, the gradient from Y^{n_l,c_i} to Q^{n_l,c_i} is increased by 1, which effectively avoids the gradient disappearance problem.

$$\begin{aligned}\frac{\partial Y^{n_l,c_i}}{\partial {W_q}^{c_i,c_j}} = &\left(1+G\cdot\frac{\partial B\left(Q^{n_l,c_i}\right)}{\partial Q^{n_l,c_i}}\right)\cdot B\left(X^{n_l,c_j}\right)\\ &\cdot\frac{\partial B\left(W_q^{c_i,c_j}\right)}{\partial W_q^{c_i,c_j}},\end{aligned} \tag{15}$$

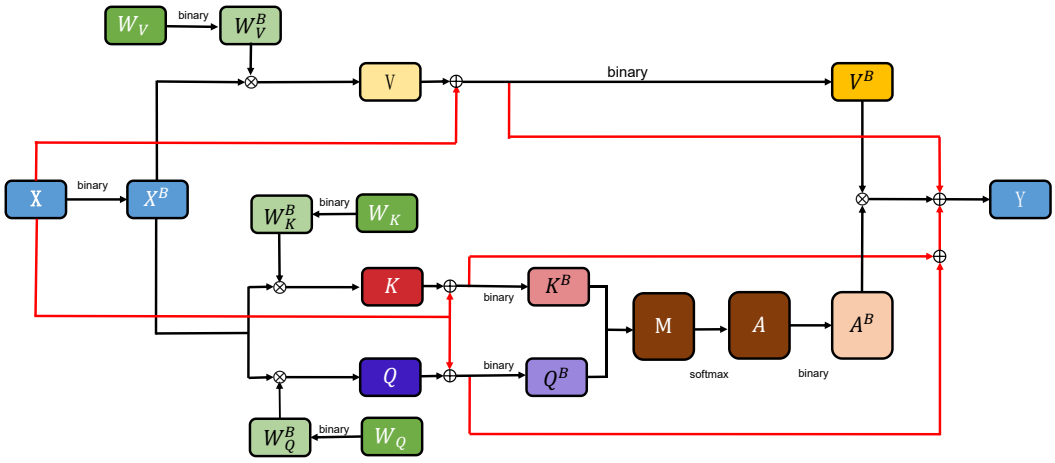

Figure 4. The information flow of the binary attention of ViT. The path with the black line and arrow refers to the information flow of binary MHSA with the original architecture. The red line and arrow path denote the added residual branch, which can solve the vanishing gradient problem caused by the superposition of truncated functions without introducing too much computation.

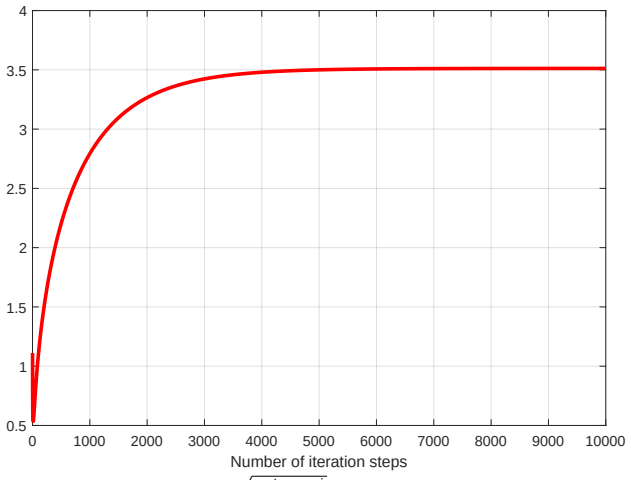

Figure 5. The value of $\frac{\sqrt{\sum_{i=1}^{t}\beta_2^i}}{\sum_{i=1}^{t}\beta_1^i}$ with respect to the number of iteration steps i.

Observation 3. *The Adam optimizer enlarges the weight oscillation of binary networks in the later stages of the training process, failing to update numerous parameters effectively.*

Detailed illustration. The previous work [9] asserts that the regularization effect of second-order momentum in Adam is beneficial for reactivating deactivated weights, which is more effective than SGD. However, because a significant proportion of elements in latent weights are close to zero, weight oscillation becomes a common issue in binary networks during the later stages of model training. To shed light on the underlying reason why the Adam optimizer is not well-suited for the binary network towards the end of training, we explain this phenomenon by re-examining the Adam algorithm. The computational operations involved in Adam are defined by Eq.16 and Eq.17 [4].

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\, g_t, \\ s_t &= \beta_2 s_{t-1} + (1-\beta_2)\, g_t^2, \end{aligned} \tag{16}$$

where m_t is first-order momentum, a weighted average of the 1st-order gradients (g_t) over time. t is the number of iteration steps. s_t is second-order momentum. β_1 and β_2 are two proportional coefficients 0.9 and 0.999, respectively. Then the first- and second-order momenta after coefficient correction, $\hat{m}_t$ and $\hat{s}_t$, and the final gradient $g_t^{'}$ which involved in the weight update (including learning rate η) are shown in Eq. 17 [4],

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{s}_t = \frac{s_t}{1-\beta_2^t}, g_t^{'} = \frac{\eta \hat{m}_t}{\sqrt{\hat{s}_t + \varepsilon}}, \tag{17}$$

where $\varepsilon = 10^{-8}$. β_1^t and β_2^t are the t power of β_1 and β_2, respectively. Then we put Eq. 16 into Eq. 17 and get the simplified form of $g_t^{'}$, as shown in Eq. 18,

$$g_t^{'} = \eta \frac{\frac{\sum_{i=1}^{t}\left((1-\beta_1)\beta_1^{t-i} g_i\right)}{1-\beta_1^t}}{\sqrt{\frac{\sum_{i=1}^{t}\left((1-\beta_2)\beta_2^{t-i} g_i^2\right)}{1-\beta_2^t}} + \varepsilon}, \tag{18}$$

Based on the properties of geometric sequence that $\sum_{i=1}^{t}\beta^i = \frac{\beta(1-\beta^t)}{(1-\beta)}$, Eq. 18 can be further simplified to

Eq. 19.

$$
\begin{aligned}
g_t^{'} &= \eta \frac{\frac{\sum_{i=1}^{t}\left(\beta_1^{t-i+1} g_i\right)}{\sum_{i=1}^{t} \beta_1^i}}{\sqrt{\frac{\sum_{i=1}^{t}\left(\beta_2^{t-i+1} g_i^2\right)}{\sum_{i=1}^{t} \beta_2^i}}+\varepsilon} \\
&= \eta \frac{\sqrt{\sum_{i=1}^{t} \beta_2^i}}{\sum_{i=1}^{t} \beta_1^i} \cdot \frac{\sum_{i=1}^{t}\left(\beta_1^{t-i+1} g_i\right)}{\sum_{i=1}^{t}\left(\beta_2^{t-i+1} g_i^2\right)+\sqrt{\sum_{i=1}^{t} \beta_2^i} \varepsilon},
\end{aligned}
\tag{19}
$$

As t increases, the term $\sqrt{\sum_{i=1}^{t} \beta_2^i} \varepsilon$ gradually increase. The value of $\frac{\sqrt{\sum_{i=1}^{t} \beta_2^i}}{\sum_{i=1}^{t} \beta_1^i}$ according to the number of iteration steps is shown in Fig. 5. The value of $\frac{\sqrt{\sum_{i=1}^{t} \beta_2^i}}{\sum_{i=1}^{t} \beta_1^i}$ approximately equals 3.51 when the number of iterations i is larger than 5000. So the $g_t^{'}$ is further simplified to Eq. 20.

$$
g_t^{'} \approx 3.51 \eta \times\left(\frac{\sum_{i=1}^{t}\left(\beta_1^{t-i+1} g_i\right)}{\sqrt{\sum_{i=1}^{t}\left(\beta_2^{t-i+1} g_i^2\right)}+\sqrt{\sum_{i=1}^{t} \beta_2^i} \varepsilon}\right),
\tag{20}
$$

where the value of $g_t^{'}$ is determined by the learning rate η, β_1, β_2, and the g_i. When the weight oscillation happens, the sign of gradient g_i changes frequently, causing the numerator of Eq. 20 in different iteration steps to cancel each other out, while the denominator of Eq. 20 keeps growing (Note that the decay rate of β_2^t is much smaller than that of β_1^t). As a result, many parameters close to 0 are deactivated in the later stages of model training. To solve this problem, we add a regularization loss function to constrain the distribution of weights to keep them away from zero.

3. Experiment

3.1. Ablation study

Architecture Details The hyper-parameters of our BHViT can be summarized in Tab. 1.

Table 1. Hyper-parameters of BHViT (n is 64).

Parameter	BHViT-tiny	BHViT-small
The number of blocks	[2,2,6,2]	[3,4,8,4]
The dimension of activation	[n,2n,4n,8n]	[n,2n,4n,8n]
The expand ratio of MLP	[8,8,4,4]	[8,8,4,4]
The number of attention head	[4,8]	[4,8]

Binary atrous convolution layer As shown in Fig. 6, due to the introduced "0" states, the binary atrous convolution layer is not suitable for deployment on binary devices. To solve this problem, we could use the shift operation proposed in section 4 to obtain the feature at the position that the corresponding weight of atrous convolution is nonzero. Then, the select feature and corresponding weight are reshaped to one dimension to apply the "xnor" and "popcount" operations instead of the multiplication between binary vectors.

In another way, we could apply the max pooling layer (with no additional FLOPs) coordinated with standard 3×3 convolution to implement the convolution with different receptive fields. As shown in Tab. 2, we conduct a performance comparison between the token mixer using dilated convolutions and the version using max pooling.

Table 2. The performance of BHVIT with different version token mixer.

Network	Token mixer	Top1(%)
BHViT	Dilated Convolution	70.1
BHViT	Max pooling	69.8

According to Tab. 2, each version of the token mixer has its advantages. Dilated convolutions obtain higher classification accuracy, but deploying this setting requires preprocessing for the activation. Deploying the token mixer with max pooling is relatively easy, but obtaining a relatively lower accuracy.

The ablation study about the latency To obtain a latency result comparison between the full precision BHViT and the corresponding binary version, we first transfer the Pytorch code of BHViT to the ONNX version. Then, we utilize the BOLT toolbox [3] to implement our method to the edge device based on an ARM Cortex-A76 CPU (without cuda). The result is shown in the Tab. 3. Due to the lack of optimization and deployment methods for the specific modules in the ViT structure, the acceleration results of BHViT cannot achieve an ideal acceleration state the same as the BNN. Therefore, further deployment techniques must be developed to show the full advantages of binary vision transformers on edge devices.

Table 3. The latency result of the full precision BHViT and binary BHViT.

Network	W/A (bit)	Latency (ms)
BHViT	32/32	612
BHViT	1/1	157

The impact of different architecture: In this subsection, we compare the performance differences of three variants

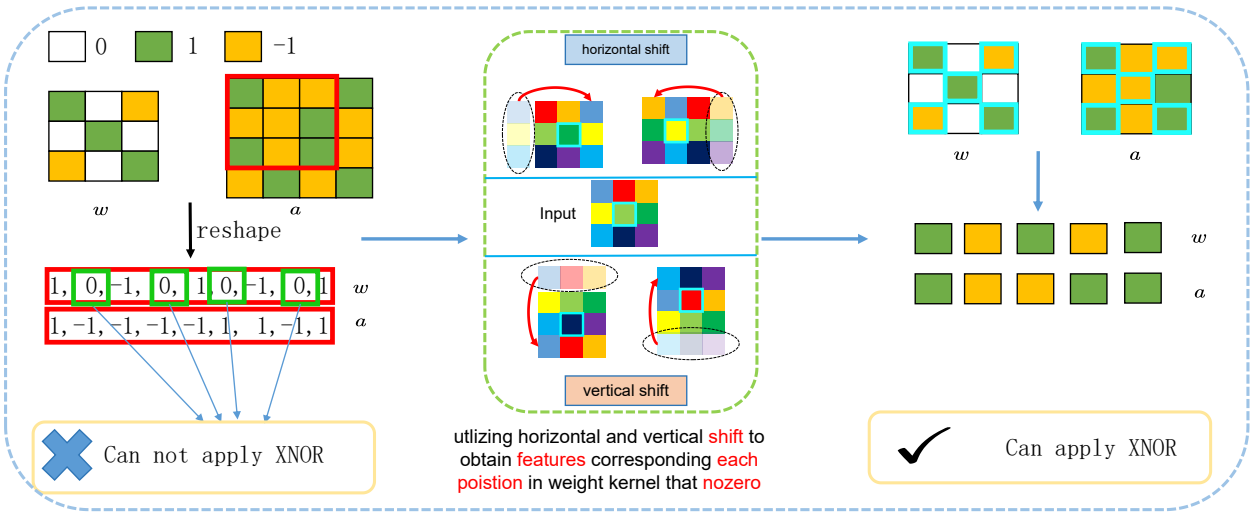

Figure 6. The process of applying shift operation for binary atrous convolution layer to approximate the vector multiplication by Xnor and popcount.

of ViT architectures before and after the binarization process. As shown in Tab. 4, The accuracy differences of the three network structures before and after binarization are 30.4%, 12.2%, and 10.9%, respectively. Compared with DeiT-Small and BinaryViT, the network architecture of the proposed BHViT is more suitable for binarization.

Table 4. The performance difference of three variants of ViT.

Network	Binary method	W/A	Top1(%)
DeiT-Small [11]	ReActNet [8]	32-32 / 1-1	79.9 / 49.5
BinaryViT [5]	BinaryViT [5]	32-32 / 1-1	79.9 / 67.7
Ours	Ours	32-32 / 1-1	79.3 / 68.4

References

[1] J Aitchison and JAC Brown. The lognormal distribution. university of cambridge, department of applied economics monograph 5, 1957. 2

[2] Yefei He, Zhenyu Lou, Luoming Zhang, Weijia Wu, Bohan Zhuang, and Hong Zhou. Bivit: Extremely compressed binary vision transformer. *arXiv preprint arXiv:2211.07091*, 2022. 4

[3] HuaWei-Noah. Bolt, 2020. 6

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

[5] Phuoc-Hoan Charles Le and Xinlin Li. Binaryvit: Pushing binary vision transformers towards convolutional models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition workshop*, pages 4664–4673, 2023. 4, 7

[6] Yanjing Li, Sheng Xu, Mingbao Lin, Xianbin Cao, Chuanjian Liu, Xiao Sun, and Baochang Zhang. Bi-vit: Pushing the limit of vision transformer quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3243–3251, 2024. 4

[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*, pages 722–737, 2018. 4

[8] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Proceedings of the European Conference on Computer Vision*, pages 143–159. Springer, 2020. 7

[9] Zechun Liu, Zhiqiang Shen, Shichao Li, Koen Helwegen, Dong Huang, and Kwang-Ting Cheng. How do adam and training strategies help bnns optimization. In *International conference on machine learning*, pages 6936–6946. PMLR, 2021. 5

[10] 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*, 2022. 1

[11] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 7

[12] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceed-*

ings of the IEEE/CVF International Conference on Computer Vision, pages 5785–5795, 2023. 1

[13] Helen M Walker and M Helen. De moivre on the law of normal probability. *Smith, David Eugene. A Source Book in Mathematics, Dover*, pages 64690–4, 1985. 1

[14] 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*, pages 568–578, 2021. 1

[15] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022. 1