Boosting Binary Neural Networks via Dynamic Thresholds Learning

Jiehua Zhang, Xueyang Zhang, Zhuo Su, Zitong Yu, Yanghe Feng, Xin Lu, Matti Pietikäinen, Li Liu

Abstract—Developing lightweight Deep Convolutional Neural Networks (DCNNs) and Vision Transformers (ViTs) has become one of the focuses in vision research since the low computational cost is essential for deploying vision models on edge devices. Recently, researchers have explored highly computational efficient Binary Neural Networks (BNNs) by binarizing weights and activations of Full-precision Neural Networks. However, the binarization process leads to an enormous accuracy gap between BNN and its full-precision version. One of the primary reasons is that the Sign function with predefined or learned static thresholds limits the representation capacity of binarized architectures since single-threshold binarization fails to utilize activation distributions. To overcome this issue, we introduce the statistics of channel information into explicit thresholds learning for the Sign Function dubbed DySign to generate various thresholds based on input distribution. Our DySign is a straightforward method to reduce information loss and boost the representative capacity of BNNs, which can be flexibly applied to both DCNNs and ViTs (i.e., DyBCNN and DyBinaryCCT) to achieve promising performance improvement. As shown in our extensive experiments. For DCNNs, DyBCNNs based on two backbones (MobileNetV1 and ResNet18) achieve 71.2% and 67.4% top-1 accuracy on ImageNet dataset, outperforming baselines by a large margin (i.e., 1.8% and 1.5% respectively). For ViTs, DyBinaryCCT presents the superiority of the convolutional embedding layer in fully binarized ViTs and achieves 56.1% on the ImageNet dataset, which is nearly 9% higher than the baseline.

Index Terms—Binary neural network, Vision transformer, Model compression, Object recognition

I. INTRODUCTION

In the past decade, Deep Neural networks (DNNs), especially earlier Deep Convolutional Neural Networks (DCNNs) [1]–[4] and more recent Vision Transformer (ViT) [5], [6], have brought tremendous progress in numerous computer vision tasks including image classification [1], [5], [7]–[9], object detection [10]–[12], image segmentation [13]–[15], and video classification [16]–[18], etc., and are still developing dizzyingly fast. Progress in vision, however substantial, has been achieved by increasingly large and complex DCNNs [5], [19] that are computationally expensive, environmentally unfriendly (due to the massive carbon footprints), and cannot be deployed in resource-constrained edge devices like the Internet of Things, and portable devices. Therefore, to mitigate these challenges, efficient processing of DNNs [20], [21] has attracted enormous attention, and numerous methods have been proposed including model quantization [22]–[24], network pruning [25]–[27], model distillation [7], [28], compact networks [29], [30] and efficient architecture design [31]–[34].

Among the aforementioned mainstreams for DNN compression, model quantization [35] aims at reducing the number of bits per weight and (or) activation for saving memory storage and computation via various forms of data quantization. As an extreme case of model quantization, Binary Neural Networks (BNNs) [36]–[38] binarize model weights and activations, which can drastically save computing resources and lead to considerable speedup. Therefore, BNNs have appeared as one of the most promising solutions for implementing deep models on edge devices. Despite the great computational efficiency, the binarization results in severe information loss and more difficulty during model optimization. These lead to constrained representational capacity of BNNs and big accuracy drops in comparison with their real-valued versions. Hence, how to train highly accurate BNNs remain open. If successful, accurate BNNs will enable ubiquitous vision applications at the edge near the sensor.

Existing studies on how to optimize BNNs can be grouped into the following categories [39]:

- **Minimize the binarization error.** A common practice to minimize the quantization error for binarized weights and activations is to approximate full-precision parameters as closely as possible since directly using 1-bit (-1/+1) severely impairs the performance of BNNs. Existing researches [37], [40]–[43] apply a scale factor for binary weights and regularize activation distribution. These methods can benefit the training of BNNs and reduce information loss in each layer to achieve a significant performance improvement.

- **Improve loss function.** Minimizing quantization error focuses on optimizing BNNs in each layer and omits the impact of the final output. Some works [44], [45] regard binarization as a training objective and design a specific loss function associated with binary weights and activations to guide the learning of BNNs globally.

- **Reduce the gradient error.** Existing BNNs are trained with the straight-through estimator (STE) technique to deal with the gradients for the Sign function. This method has two problems: 1) gradient mismatch between Sign and STE; 2) parameters not in [-1, +1] are unable to be updated. These problems seriously affect the optimization of BNNs. To solve these problems, some works [46]–[48] attempt to modify the...
binarization function in backward or forward propagation or calibrates the gradient to reduce the gradient error.

Despite the considerable research efforts, the limitations of existing research on BNNs include:

- ignoring the gap in the distribution of diverse samples.
- focusing on binarizing DCNNs (BCNNs) rather than ViTs, GANs, etc.
- focusing on image classification rather than other vision tasks like object detection, semantic segmentation, etc.

In this paper, we focus on improving the first two limitations. Many existing BNNs apply the same threshold value (like 0) to binarize all feature maps of each input image. But this might not be appropriate in all cases, e.g., activation feature maps highlight different information of the input image, and have different feature distributions. This one-for-all (one threshold for all feature maps and all images) binarization has serious limitations such as failing to adapt to feature map distributions. It is easy to realize [43] that the binarized feature map is sensitive to the threshold value used for binarization, as the fluctuation above or below the threshold can cause the binarized feature map to have a different appearance. In this case, adaptive thresholding is beneficial. Therefore, Liu et al. [43] proposed ReActNet that learns a threshold for each feature map channel by a generalization of Sign function (RSign) and produces higher accuracy. However, the learnable thresholds in the RSign are image-agnostic and still neglect activation variations of diverse samples, which makes ReActNet only adapt to feature channels, not input images. For various input images, the image-agnostic threshold fails to preserve representative features since considerable variations in their feature distributions. As shown in Fig. 1, the thresholds in RSign fit with images in the first two columns but drop valuable object information and introduce excess background noise features for other images, which limits representation learning of BNNs.

To address this issue, the learning of thresholds should be based on statistics of channel information in each layer, which is equivalent to learning sample-independent thresholds. The sample-independent thresholds can shrink the gap between binarized and full-precision features by reshaping activation distributions based on input. To achieve this goal, we take the global channel information as a statistical criterion for generating sample-independent thresholds. A lightweight computational unit is incorporated with the Sign function to assign thresholds for each channel based on the statistical criterion. We name this novel Sign function as DySign, which can adaptively perform activation distributions adjustment. Our DySign has two advantages: 1) generates diverse thresholds automatically according to feature distributions without human design; 2) can be flexibly transferred to other BNN architectures. Compared with RSign, the outputs of DySign remain informative features and suppress noise as shown in the last row of Fig. 1.

Existing BNNs in vision tasks are Binary CNNs (BCNNs). ViTs perform remarkable progress in vision tasks. However, this quadratic complexity limits ViTs to model high spatial-resolution images and deploy them on portable devices, which is crucial for vision tasks. Thus, we attempt to build a fully binarized ViT to reduce huge computational costs. We first observe that replacing the linear layer in the patch embedding layer with a convolutional block significantly improves the performance of binary ViTs since local information can be well preserved.
Therefore, we then construct a binarized ViT baseline based on Convolutional Transformer [49]–[52]. However, the binarization error in features seriously affects the self-attention module to capture the relationship between tokens. Thus, we adopt the DySign on both the self-attention layer and the feed-forward layer in each binarized transformer block to emphasize valuable information in each token.

This paper presents a substantial extension of our previous short conference paper [53] in ICASSP 2022. In addition to the contribution Dynamic Binary Neural Network (DyBCNN) to generate diverse thresholds based on the activation distribution of the input itself in our conference paper, this paper presents the following additional technique contributions: 1) we conduct a first study to develop binary ViTs by proposing BinaryCCT building upon Compact Convolutional Transformer (CCT) [49] which is a fully binarized transformer architecture; 2) we analyze the impact of binarization for standard ViTs and Convolutional Transformers; 3) we present DyBinaryCCT by introducing the DySign to input tokens in BinaryCCT to improve the learning ability of BinaryCCT.

The main contributions can be summarized as follows:

- We pioneer to propose a fully binarized transformer-based architecture for computer vision tasks. Compared with standard ViT, inserting a convolution based patching method in the tokenization layer can preserve local information between image patches and mitigate information loss caused by binarization. Thus, we build the baseline based on CCT, named BinaryCCT.
- Based on the standard Sign function, we propose Dynamic Sign (DySign), a simple and effective method to enhance BNNs performance. DySign aggregates global activation distribution to produce an embedding, allowing information from the global receptive field of feature maps to be used to adaptively assign the thresholds for each channel. In addition to the performance advantages, DySign can be flexibly migrated to other BNN frameworks. With BCNN and BinaryCCT, we propose DyBCNN and DyBinaryCCT, replacing the original Sign with DySign. The results suggest the superiority of our method.
- Extensive experimental evaluations on the commonly used datasets for image classification. For DyBCNN, we demonstrate the effectiveness of on the ImageNet dataset. The DyBCNN is built on two networks (ReActNet and ReActNet based on ResNet18), achieving 71.2% and 67.4% top1-accuracy, outperforming strong baselines 1.8% and 1.5% by a large margin, respectively. For DyBinaryCCT, we evaluated on CIFAR10, CIFAR100, and ImageNet dataset. The DyBinaryCCT can achieve 84.98% on CIFAR10, 62.10% on CIFAR100, and 56.09% on ImageNet, outperforming average 7% over baseline. The proposed DySign significantly advances the performance of BNNs and flexibly migrate to various BNN frameworks.

The remainder of this paper is organized as follows. Section II reviews the related works of binary neural networks and vision transformer. Section III formulates the proposed DySign, and the architectures of DyBCNN and DyBinaryCCT. Section IV provides rigorous ablation studies and evaluates the performance of the proposed DyBCNN and DyBinaryCCT on the image classification task, respectively. Finally, the conclusion and future work are presented in Section V.

II. RELATED WORK

A. Binary Neural Networks

The binary neural networks (BNN) serves as a favorable technique for deploying deep learning models on portable devices. Soudry et al. [54] claimed that good performance could be achieved when all neurons and weights are binarized. They proposed the expectation backpropagation (EBP) method to train the network with discrete values. Hubara et al. [36] binarized the weights and activations in networks and illustrated its success on the small datasets. While BNN largely saves storage and computation, the accuracy gap between BNN and full-precision model limits its application, especially when evaluated on large-scale dataset, like ImageNet. This issue is caused by the severe information loss and difficulty to the optimization.

In recent years, lots of works attempted to improve the accuracy by introducing extra feature information to reduce quantization error and information loss. Rastegari et al. [37] proposed XNOR-Net, which incorporated scaling factors to multiply with binary kernel weights to reduce binarized error. Based on this, Liu et al. [38] introduced an extra shortcut operation between continuous binary convolutional blocks to enhance representational capacity and reduce information loss of BNNs. Liu et al. [43] also demonstrated that BNNs were sensitive to activation shift. They proposed simple activation distribution shift functions and built BNN based on MobileNetV1, resulting in a significant accuracy improvement. Martinez et al. [55] introduced a learning branch to compute the scale factors to rescale the activations. Besides, several methods also attempt to alleviate the optimization difficulty by using approximate gradient. IR-Net [46] enhances the performance of BNNs by retaining both forward and backward information. Xu et al. [47] estimated the gradient of sign function in the frequency domain and proposed frequency domain approximation (FDA), which can better maintain the main direction of factual gradient. We considered to improve representational power of BNNs by replacing previous fixed and static thresholds with dynamic thresholds.

B. Vision Transformer

Transformer [56] is firstly proposed to model sequential data for the machine translation task. Recently, the Vision Transformer (ViT) [5] is proposed by splitting an image into patches and providing the sequence of linear embedding of these patches as an input to the transformer-based architecture. Many other ViT variants [6], [12], [28], [57]–[60] are then proposed and achieve great performance. The challenges of ViT involve: 1) The ViT is data-hungry and fails to generalize well on small datasets due to the lack of inductive bias of CNN, which can help model effectively process image; 2) The ViT requires huge memory and computation, which is intolerable for edge devices.

To address the first problem, recent efforts attempt to introduce the CNN inductive bias to enhance Transformer. Touvron et al. [7] proposed a Data-efficient Transformer (DeiT) to free ViT
from dependency on large-scale training data. Except for existing training strategies for data augmentation and regularization, a distillation strategy is applied for adjective representation learning. Researchers also combine the transformer with CNN [34], [49]–[52], [61]–[64] to leverage the strength of both CNNs and Transformers. These methods replace the linear patch project with the convolution stacks. Extensive experiments illustrate that the locality and translation invariance of CNN can effectively boost the performance of ViT across multiple datasets. Besides, a local attention mechanism is proposed to augment the local image information extraction ability [12], [58], [65]–[67]. One of the representative methods is the Swin Transformer [12], which conducts a shifted window to model the global and local features in the spatial dimension. The shifted windows bridge the windows of the preceding layer, providing connections among them that significantly enhance modeling power. TNT proposed by Han et al. [58] aggregates patch-level and pixel-level representation by two blocks: an inner block to capture interaction within each patch and an outer block to extract global information.

For the second problem, several works have focused on lightweight model design [68]–[71] and quantization [22], [23]. MobileViT [68] attempts to incorporate attention into MobileNetV2 to improve mobile CNNs. presents a different perspective on the global processing of information with transformers. Huang et al. [71] proposed the LightViT based on a pure transformer without convolution. For quantization, Liu et al. [23] adopted the post-training quantization algorithm to compress transformer architectures into 8-bit to reduce storage memory and computational cost. Lin et al. [22] proposed FQ-ViT to achieve 4-bit ViT with comparable accuracy degradation on fully quantized ViT. In the NLP field, researchers have compressed the transformer model to 1-bit. Bai et al. [72] compressed the weight and embedding of BERT to 1-bit. Qin et al. [73] and Liu [74] proposed fully binarized BERT and provided theoretical formulation and empirical observation for the degradation of the information of attention weight. In this paper, we focus on binarizing the vision transformer model for the visual recognition task.

III. PROPOSED METHOD

In this section, we will first introduce the basic knowledge of binary neural networks and Compact Convolutional Transformer (CCT). Then, we present detail information about the proposed DySign. Based on this, then we introduce architectures of DyBCNN and DyBinaryCCT, which can be observed in Fig. 2. The detailed approach for each aspect will be introduced following.

A. Preliminary

Binary Neural Networks. BNNs utilize the Sign function to constrain weights and activations to +1 or -1. For explaining the operation of binarized convolution, we denote $W^\ell$ and $X^\ell$ as the weights and input features in the $\ell$-layer. The input of the $\ell + 1$-layer can be defined as:

$$X^{\ell+1} = \phi^\ell(\text{Sign}(W^\ell) \otimes \text{Sign}(X^\ell)),$$

where

$$\text{Sign}(x) = \begin{cases} +1, & x > 0, \\ -1, & x \leq 0, \end{cases}$$

This equation avoids the multiplication operation and uses the Sign function to convert the values into binary numbers.
where $\otimes$ denotes convolutional operation, the $\phi^l(\cdot)$ denotes the nonlinear operation in the $l$-layer (ReLU, PReLU, and BN layer, etc.). To enhance the performance of BNNs, the ReActNet [43] utilized the learnable channel-wise thresholds to shift feature maps for retaining valuable information, which can be expressed as below:

$$R\text{Sign}(x_i) = \begin{cases} +1, & x_i > a_i, \\ -1, & x_i \leq a_i, \end{cases}$$  \hspace{1cm} (3)

where $x_i$ denotes an element of $i$-th channel in input feature map, $a_i$ denotes the threshold of $i$-th channel, which illustrates that the threshold can vary for different channels. The PReLU function is also processed by this operation to reshape the feature map, which can be expressed as:

$$\text{RPReLU}(x_i) = \begin{cases} x_i - \gamma_i + \zeta_i, & x_i > \gamma_i, \\ \beta_i(x_i - \gamma_i) + \zeta_i, & x_i \leq \gamma_i, \end{cases}$$  \hspace{1cm} (4)

where $\gamma_i$ and $\zeta_i$ denote the learnable shift parameters for reshaping the distribution. By introducing the activation distribution shift and reshape, ReActNet achieves a significant accuracy increase in image classification.

**Compact convolutional transformer.** Vision Transformer (ViT) [5] is the first work to construct a pure transformer architecture for computer vision tasks. ViT consists of several components: Image Tokenization, Positional Embedding, Classification Token, the Transformer Encoder, and a Classification Head. The motivation of ViT is that it takes a sequence of image patches as the input and utilizes the self-attention block to capture the long-range relationship. However, ViT is only applicable when trained on a large-scale dataset, and it generalizes poorly when trained on insufficient amounts of data due to the lack of inductive biases.

To reduce the dependency upon data, the Compact Convolutional Transformer (CCT) [49] introduces a convolutional based patching method and achieves prominent performance on the small dataset. CCT is composed of several parts: Convolution Block, Transformer Encoder, Sequence Pooling, and a Classification Head. The position embedding is optional in CCT. Detailed information on these components is presented below:

1) **Convolution block:** To improve the generalization ability of vision transformers on the limited training data, CCT replaces the image patching and projection layers in the ViT with a simple convolutional block. The convolutional block consists of a single convolutional layer, ReLU activation, and a max pooling layer. Compared with ViT directly patching on images, the convolutions can embed the image into a latent representation, which is more efficient. Furthermore, the convolution block can inject inductive bias into the model and preserve local spatial information. With these characteristics, CCT can provide competitive results on a small dataset.

2) **Transformer encoder:** The transformer encoder layer remains the same as the original Transformer [56] and ViT [5], consisting of a Multi-head attention (MHSA) layer and a Feed-Forward (FFN) layer. It utilizes Layer Normalization [75], GELU activation, and dropout. The positional embedding is defaulted as learnable.

3) **Sequence pooling:** Sequence Pooling (SeqPool) aims at pooling the sequential based information from transformer encoder layers and can eliminate the need for the classification token [49]. SeqPool can be regarded as the mapping transformation: $\mathbb{R}^{b \times n \times d} \rightarrow \mathbb{R}^{k \times d}$. It makes CCT weigh the sequential embeddings of the latent space from the transformer encoder and capture the relationship between the input data.

**B. DySign: Dynamic Sign Function**

Although previous works have achieved remarkable progress in BNNs, the feature information loss still severely degrades representation learning of BNNs. The ReActNet focuses on activation distribution and adopts static channel-wise thresholds on the input feature maps of each layer. However, these image-agnostic thresholds neglect the difference between various images and fail to remain informative features for all cases. As observed Fig. 1, the RSign can preserve object information for the samples in the blue box but still eliminate informative features for the samples in the red box. To address this issue, BNNs requires statistical channel information to adjust thresholds adaptively when processing different input. This motivates us to insert a computational unit in the Sign function to calculate thresholds based on the activation distribution of input feature maps. We name this novel Sign function as DySign.

We first introduce the DySign in BCNNs. The DySign can be regarded as a feature transformation function with learnable parameters $f(X)$, mapping an input feature tensor $X \in \mathbb{R}^{C \times H \times W}$ with spatial size $H \times W$ and $C$ channels to binarized feature tensor $X_{\text{binary}} \in \mathbb{R}^{C \times H \times W}$ with more informative features. The function $f(X)$ aggregates information from the global receptive field of the network and then computes the thresholds $\alpha_{1:C}$ for each channel in feature map $X$. The calculation of $\alpha$-th element of $\alpha$ can be written as:

$$\alpha_c = f(X_c) = W_2(W_1(\frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_c(i,j)))$$  \hspace{1cm} (5)

where $W_1 \in \mathbb{R}^{C \times C}$ and $W_2 \in \mathbb{R}^{C \times C}$ denote two fully-connected (FC) layer. Based the output $\alpha_{1:C}$ from $f(X)$, DySign can binarize the input feature maps $X$ as:

$$\text{DySign}(X_i) = \begin{cases} +1, & X_i > \alpha_i, \\ -1, & X_i \leq \alpha_i, \end{cases}$$  \hspace{1cm} (6)

where $\alpha_i$ denotes the threshold of $i$-th channel, which is the $i$-th element of output vector from $f(X)$. The computational unit of DySign can be observed in Fig. 3. The DySign adopts a SEBlock [8] to learn a set of channel-wise thresholds from the input feature maps for Sign function. The process is “input $\rightarrow$ GAP $\rightarrow$ FClayer $\rightarrow$ FClayer”. To avoid over-fit and reduce the extra computational cost, the reduction ratio $\gamma$ is set between two FC layers.

For BinaryCCT, the embedding image feature tensor $X \in \mathbb{R}^{C \times H \times W}$ is split into a long sequence $X \in \mathbb{R}^{N \times D}$ with sequence length $N$ and embedding dimension $D$ and then passes through transformer blocks. Compared with BCNNs, BinaryCCT suffers more serious performance degradation since binarization error in each token makes the MSA layer
fail to capture distinguishable attention. Thus, DySign can be applied to eliminate this shortcoming. The small difference in the computational unit in DySign between DyBCNN and DyBinaryCCT is that we add the GELU function among two FC layers. In BinaryCCT, we consider generating dynamic thresholds in token-wise. The token-wise threshold $\alpha \in \mathbb{R}^{1 \times N}$ is determined by the global information of each token. We apply the DySign in both MSA and FFN layers. Although this function introduces extra FLOPs and memory cost, it can significantly improve the performance of binarized transformers in vision tasks, which will be discussed in Section IV-C.

DySign intrinsically introduces dynamic thresholds conditioned on the input. This hyperfunction maps the global information of input $X$ to its binarization distribution with more informative features. Based on the simple computational unit in DySign, channel dependencies are explicitly modeled to assign appropriate thresholds for each channel, which can enhance the representation learning of BNNs. In the experiment section, we will show that dynamic learning distribution is an effective and straightforward way to boost the performance of BNNs.

C. The Architecture of DyBCNN

Except for the DySign, we also handle the RPReLU function [43] in the same way. The shift parameters $\gamma^{1:C}$ and $\xi^{1:C}$ are learnable based on input feature maps (DyPReLU). The hyperfunction $f(\cdot)$ generates the shift parameters for each channel in input feature maps. This process can dynamically shift and reshape feature distribution, which is an effective and simple way to increase model capacity.

Essentially, the DySign can learn the most suitable thresholds $\alpha^{1:C}$ to binarize the input feature map. The threshold parameters can be dynamically adjusted for different input feature maps, which can effectively limit the feature information loss after binarization. For DyPReLU, the $\gamma^{1:C}$ and $\xi^{1:C}$ can be easily understood as these parameters are dynamically generated to obtain better output distribution. By introducing these functions, the aforementioned problem risen by static parameters can be eliminated. The BCNNs can retain more object information and learn more meaningful features. In the experiment section, we will show that adaptive thresholds learning is an effective and straightforward way to boost the performance of BCNNs.

For model architecture, The ResNet18 [1] and MobileNetV1 [76] are built as the backbones following ReActNet [43]. In the ReActNet (Shown as in Fig. 2), the $3 \times 3$ depthwise and $1 \times 1$ pointwise convolution are replaced by standard $3 \times 3$ and $1 \times 1$ convolution. The duplication and concatenation operations are designed for addressing the channel number difference. In our case, we simply replace the RSign and RPReLU functions in ReActNet with our DySign and DyPReLU. The ReActNet can be regarded as the baseline in our experiment.

D. The Architecture of DyBinaryCCT

We start with introducing the basic components of BinaryCCT. The tokenization layer and classification head maintain full precision. In each fully binarized transformer block, the matrix multiplication with bitwise XNOR and Popcount operation in the binarized linear layer is presented in Fig. 4.

For the weights in all binarized linear layers, the scale factor is applied to minimize quantization errors and zero-mean is utilized for retaining representation information [73]. The activation is only utilized Sign Function to binarize. The complete computational process can be expressed as:

$$\text{BiLinear}(X) = \alpha_w (\text{Sign}(X) \otimes \text{Sign}(W - u))$$

(7)

where $X$ and $W$ denote full-precision activations and weights, $u$ is the mean value of weight $W$, $\alpha_w = \frac{1}{n} \|W\|_{l^1}$, denotes the scaling factor for weight.

In the BinaryCCT, the input data first passes through a full-precision convolutional block to generate tokens before being fed into multiply transformer encoder layers. The calculation between queries $Q$, keys $K$ and values $V$ consumes huge computational cost, which can be mitigated in binarized version. For one MHSA layer, the computation in an attention head can be expressed as:
where the OPs can be total operations (OPs), which consists of binary operations (BOPs) and floating point operations (FLOPs). The OPs can be obtained as:

\[ \text{OPs} = \frac{\text{BOPs}}{64} + \text{FLOPs} \]  

Taking DyBCNN as an example, we do not introduce extra binary convolutional operations. Thus, the BOPs are the same as ReActNet. The increased computational consumption mainly comes from floating-point operations in DySign and DyPReLU, including one global average pooling layer and two fully-connected layers. For reducing the introduces computational cost, we set the reduction ratio between two fully-connected layers as 16, which can limit the introduced model parameters and float operations. We denote the size of the input feature map as \( C \times H \times W \). The FLOPs for each RSign and RPReLU increase \( C + \frac{C}{2} \) and \( 2 \times (C + \frac{C}{8}) \). The extra computational cost is only a small fraction of the total cost, which is above 2% for DyBCNN and DyBinaryCCT.

### IV. Experiments

#### A. Datasets and Implementation details

To evaluate the performance of our proposed method, we conduct experiments on the image classification task and evaluate three datasets: CIFAR10, CIFAR100, and ImageNet dataset [82]. The CIFAR-10 dataset consists of 60000 32x32 images, 50000 images for training, and 10000 for testing. The ILSVRC12 ImageNet classification dataset contains 1.2 million training images and 50,000 validation images across 1000 classes, which is more challenging than small datasets. In this section, we first introduce the training details for DyBCNN and DyBinaryCCT. Then we provide the experimental analysis and performance of the two models separately.

**DyBCNN implementation details.** For DyBCNN, we evaluated our method on the ImageNet dataset and compared it with recent state-of-the-art methods. The training strategy utilizes the two-step training strategy described in [55]. In the first step, the network is trained from scratch with binary activations and weights. In the second step, the network inherits the weight from the first step and is trained with binary activations and weights. For both steps, we follow the training scheme in [43]. We use Adam optimizer and a linear learning rate decay to optimize the model. The initial learning rate is set to 5e-4, batchsize is set to 256, and \( \gamma \) in DySign is set to 16. Besides, we also train a quick version of DyBCNN with the one-step training strategy.

**DyBinaryCCT implementation details.** We conducted extensive experiments on three datasets: CIFAR10, CIFAR100, and ImageNet. The training hyper-parameters follow the CCT [49]. All reported accuracy is best out of 4 runs. For CIFAR10 and CIFAR100, the initial learning rate is set to 1e-3 with cosine annealing decay strategy, and weight decay is set to 1e-4. The batchsize is set to 128. The training stage is run for 300 epochs on CIFAR10 and CIFAR100. AutoAugment [83] is used.
for CIFAR-10 and CIFAR-100. For the ImageNet dataset, the training stage is run for 300 epochs with a batchsize of 64. The initial learning rate is set to 5e-4 with a cosine annealing decay strategy. The weight decay is set to 1e-4. The model is warmed up for 25 epochs. The γ in DySign is set to 4.

B. Experiment on DyBCNN

We compare the DyBCNN with other state-of-the-art binarization methods in Table I. Compared with ReActNet and ReActNet-ResNet18, the DyBCNN and DyBCNN_ResNet18 achieve 1.8% and 1.5% increase, respectively. The superior improvement further proves the importance of activation distribution for BNNs. We also provide the test accuracy of each training step, which can be observed in Table II. Due to the introduced global average pooling layer and fully-connected layers, the OPs increase slightly. For DyBCNN, Extra OPs from DySign only account for 2% of the total computational cost, which is acceptable. With the limited computational cost increased, the DyBCNN can outperform previous methods by a large margin, which illustrates the effectiveness of dynamic learnable shift parameters in BCNNs. Besides, another advantage of our proposed method is that it is straightforward and can be easily transferred to any other binarized architecture.

| Network            | distillation | OPs             | Acc Top-1(%) |
|--------------------|--------------|-----------------|--------------|
| ReActNet           |              | 1.0             | 70.1         |
| DyBCNN             |              | 1.0             | 71.2         |

To further analyze our proposed method, we conduct following ablation studies for DyBCNN.

1) One-step Training: The DyBCNN utilizes the two-step training strategy with 512 epochs in total, which is time-consuming. To simplify the training process, we also evaluate the quick version of DyBCNN in one-step training. The result can be observed in Tabel III. We evaluate DyBCNN under the following strategies: 1) distillation with distribution loss in [43]; 2) no distillation with CrossEntropy loss. The training epoch is set to 90. Following other training details described at the beginning of Section IV-A, DyBCNN achieved 2.3% and 2.0% higher than ReActNet under the two training strategies mentioned above, which illustrated that DyBCNN is an effective and straightforward way to boost the performance of BNNs. We also observe that DyBCNN achieves higher accuracy and faster convergence (See Fig. 5).

| Network           | distillation | OPs             | Acc Top-1(%) |
|-------------------|--------------|-----------------|--------------|
| DyBCNN-ResNet18   |              | 0.97 × 10^8     | 66.3         |
| DyBCNN-MobileNet  |              | 0.99 × 10^8     | 64.3         |

2) Impacts of DySign on DyBCNN: In this section, we analyze the individual effect of DySign. We conduct the experiment in the two-step training strategy following Section IV-A. The experimental result is shown in Table IV. DyBCNN achieves 70.1% top-1 accuracy on the ImageNet without DyPReLU. Compared with ReActNet, the accuracy increases by 0.7%, illustrating the effectiveness of our proposed DySign. DySign promotes model performance by reducing feature information loss. Furthermore, DyPReLU increases model expression ability by reshaping the activation distribution of feature maps, which further improves model accuracy.

| Network           | DySign | DyPReLU | Acc Top-1(%) |
|-------------------|--------|---------|--------------|
| ReActNet          |        |         | 69.4         |
| DyBCNN            | ✓      |         | 70.1         |
| DyBCNN            | ✓      | ✓       | 71.2         |

We also visualize the thresholds learned by DySign when processing different samples. As shown in Fig. 6, the diverse thresholds can be applied to binarize features when processing the sample dog and duck. We observe that the shallower layer shares similar learned patterns since they tend to catch the basic and less abstract image features, while the deeper layer diverges different images into different thresholds. The question is raised whether a similar result would be achieved if DySign was only added to deeper layers. Thus, We design an ablation study to verify the effectiveness of our DySign in each layer. We utilize ReActNet based on ResNet18 and conduct experiments on CIFAR100. ResNet18 can be divided into four layers according to the number of channels. We replace RSign with DySign from the last layer forward. The result can be observed in Tabel V. The performance of the model improves as the number of layers incorporated with DySign increases. Compared with adding DySign only to the deepest layer, the accuracy of full use can be improved by 1.3%. Thus, this flexible method of handling different sample features is helpful for the representation learning of BNNs from shallow to deep layers.

To further evaluate the transfer ability of DySign, we apply our method to two different architectures: WRN22 [84] and VGG16.
Shallow layer
Deep layer
Fig. 6. The learning thresholds of DySign for duck and dog samples in shallow and deep layers. The similar patterns are learned in the shallow layers, and diverse thresholds are generated for processing duck and dog features in deep layers.

| Layer 1 | Layer 2 | Layer 3 | Layer 4 | Accuracy |
|---------|---------|---------|---------|----------|
| x       | x       |         |         | 66.7%    |
| x       |         | x       |         | 68.5%    |
|         | x       |         | x       | 68.8%    |
|         |         | x       |         | 69.1%    |
|         |         |         | x       | 69.6%    |

The ✓ denotes DySign applied in this layer and x denotes RSign.

The! denotes DySign applied in this layer and % denotes RSign.

[2]. WRN is a variant on ResNets that introduces a new depth factor $k$ to adjust the feature map depth expansion through 3 stages, where the spatial dimension of the features remains the same. VGG is a standard deep CNN architecture with multiple layers. In our experiment, we set the $k$ as 2 in WRN22. Thus, the kernel stages for WRN22 and VGG16 are 32-32-64-128 and 64-128-256-512-512, respectively. The DySign is added before each binarized convolutional kernel. We evaluate our method on CIFAR100 shown in Fig. 7. Compared with the baseline, DySign can improve performance by 1.62% and 4.33% for binarized WRN22 and VGG16, respectively. The result suggests that our method can flexibly transfer to other binarized CNN architectures and enhance performance.

C. Experiments on DyBinaryCCT

In this section, we experimentally analyze the BinaryCCT on both small and large scale datasets (CIFAR10, CIFAR100, and ImageNet datasets). We first compare the accuracy degradation between two different transformer architectures: Vision transformer and convolutional transformer. Then we report the accuracy of our models and implementation details. At the end of this section, we provide the ablation studies to analyze the impact of the introduced DySign.

1) ViT or CCT under binarized step: In this part, we experimentally analyzed two architectures (vision transformer and convolutional transformer) to verify which one is suitable for binarized situations. For the vision transformer, we adopted binarized ViT-Lite introduced from [49]. The ViT-Lite variants with their patch size. For instance, ViT-Lite-7/4 denotes a ViT-Lite with a 7-layer transformer encoder with $4 \times 4$ patches. For the convolutional transformer, we selected our baseline, binarized compact convolutional transformer (CCT). The CCT is specified by the number of layers, as well as the kernel size for convolutional layers. For instance, CCT-6/3 $\times$ 1 denotes the model has 6 transformer encoder layers and 1 convolutional block with $3 \times 3$. We binarized the ViT-Lite-7/8, ViT-Lite-7/4, ViT-Lite-6/8, ViT-Lite-6/4, CCT-6/3 $\times$ 1, and CCT-7/3 $\times$ 1.

We evaluate these different models on CIFAR10. The comparison of performance degradation between the two architectures can be observed in Fig. 8. The binary operation can significantly influence the module to distinguish attention weights between patches, which suppresses the expression ability of the transformer. The binarized ViT-Lite models suffer catastrophic accuracy degradation (nearly 50% on CIFAR10 and CIFAR100). For binarized CCT, the accuracy drop is less severe than that of binarized ViT-Lite. This experiment demonstrates that early...
The convolution can effectively boost the performance of binarized vision transformer architecture. For convolutional transformers, setting early visual processing of ViT as convolutions can introduce the inductive bias of CNN, which can enhance the generalization ability of ViT and improve the representation learning ability of transformer blocks. In the binarized setting, a convolution based patching method in the tokenization layer can preserve local relationship information between image tokens. Due to this property, BinaryCCT can remain more valuable information than binarized ViT during binarization. What’s more, the low capacity of a binarized linear layer is unable to directly process complicated image information. As a result, the accuracy of BinaryCCT exceeds BiViT-Lite by a large margin, revealing its superiority in the binarized transformer-based module. Thus, we select the convolutional transformer as the binarized module.

### Results on Image Classification

In this part, we discuss the performance of BinaryCCT on the image classification task. The experiments are conducted on CIFAR10, CIFAR100, and ImageNet. The

#### Results on small datasets

Firstly, we build our fully binarized transformer baseline BinaryCCT based on CCT-6/3x1 and CCT-7/3x1, named BinaryCCT_6 and BinaryCCT_7. The BinaryCCT When evaluated on BinaryCCT_6, the accuracy drops 16% on CIFAR10 and more than 20% on CIFAR100 (See Table VI). This severe performance degradation derives from indistinguishable attention in binarized transformer blocks according to the observation in Fig. 9. The training loss curves (See Fig. 10) also present the sluggish convergence of BinaryCCT. Besides, due to low model capacity, the loss curve is unable to continue to fall especially on the large-scale dataset ImageNet.

To enhance the performance of BinaryCCT, we apply DySign on the baseline models. The learnable thresholds have been proved as an effective method to enhance BNN accuracy [43], [53]. As shown in Table VI, the accuracy of DyBinaryCCT_6 is improved by nearly 6% on CIFAR10 and nearly 8% on CIFAR100. The DySign is a straightforward and effective way to reduce information loss and boost the representational ability of BinaryCCT. The detail analysis can be observed in Section IV-C3.

#### Results on large datasets

Except the small dataset, we also explore traits of BinaryCCT on large-scale dataset ImageNet. Due to the limitation of computing resources, we selected CCT-7/7×2 as our baseline model, which consists of 7-layer transformer block and 2-layer convolutional embedding layer. The experiment is shown in Table VII. Compared with baseline, BinaryCCT only achieves 35.65% top-1 accuracy, accuracy loss of approaching 40%. Based on the observation in Fig. 10, the sluggish rate of descent loss and convergence speed demonstrate that BinaryCCT is unable to distinguish property for complicate object categories due to low-capacity. We attempt to improve the capacity from two perspectives: depth and width. For depth, we increase the transformer blocks, eg. 14-layer. For width, we wide the dimension in each transformer block, eg. 384 and 512.

In this paper, we utilized CCT-14/7×2, increasing number of transformer block to 14 to evaluate the impact of depth. And we set the embedding dimension as 256, 384, 512 to evaluate the impact of width. The experimental results observed in Table VII suggest increasing embed dimension of transformer block is an effective way to boost performance of BinaryCCT. When the dimension is extended from 256 to 384, the accuracy of the model increased by 6.03%. For adding transformer block, the accuracy of 14-layer BinaryCCT is 3.4% lower than 7-layer BinaryCCT. Due to the severe loss of semantic information, the model is unable to capture the correct relationship between tokens. Therefore, deepening binarized transformer layers fails to enhance the representation ability of the model and confuses classifier by indistinguishable attention. Besides, the matrix multiplication
in transformer blocks decompose to XNOR PopCount operation. Thus, increasing width of BinaryCCT only introduces negligible amount of computational cost. As shown in Table VII, the red denotes the FLOPs of full-precision model and blue denotes the OPs after binarization. When extending embedded dimension from 256 to 512, OPs of BinaryCCT only increases 0.04G while achieving significant performance improvement.

The experiment of DyBinaryCCT is shown in Tabel VIII. We set the embedding dimension to 512. With DySign, DyBinaryCCT can achieve 56.09% top-1 accuracy, nearly 9% higher than the baseline. The result suggests the importance of retaining token information for the binarized transformer performance. The single-threshold leads to the MHSA layer being unable to capture the relationship between each token since excessive noise and information loss exist in binary tokens. DySign reduces the negative effects of binary operation by adjusting the distribution of tokens so that the attention module can capture the accurate dependencies between tokens.

For the computational cost, although floating-point operations of the convolutional tokenization layer occupy almost 70% of the FLOPs and affect the overall calculation consumption, the computational cost of binarized transform blocks is only 2% of the full-precision version. Therefore, deepening or widening the BinaryCCT to further improve the performance will not introduce too much computation.

### V. Conclusions

In this paper, we first propose a simple and effective computational unit to adaptively binarize features in BNNs, named as DySign. DySign explicitly models the channel dependencies to assign appropriate thresholds for each channel, which can effectively reduce the feature information loss and enhance the representation learning of BNNs. Furthermore, we proposed a fully binarized transformer architecture for vision tasks, named BinaryCCT. We evaluate the proposed DySign on BCNN and BinaryCCT, two popular backbones in the visual recognition tasks. The extensive results suggest that our learnable thresholds can effectively improve the representation ability of CNN by eliminating the information loss in the binarized operation, especially on CIFAR100. For RSign, accuracy improvement on CIFAR100 is about 2% compared with the baseline, which is 6% lower than DySign. The comparison in Tabel IX demonstrates that the static threshold is not sufficient to compensate for the accuracy gap between full-precision model due to the information loss and low capacity of binarized linear layers. The dynamic threshold learning module adaptively generates the thresholds based on input tokens, which effectively reduce information loss and boost the representational ability of binarized transformer block.

Furthermore, We compare the performance of applying DySign channel-wise and token-wise. Different from token-wise DySign (thresholds $\alpha \in \mathbb{R}^{1 \times N}$), the channel-wise thresholds $\alpha \in \mathbb{R}^{1 \times D}$ is determined by the average information in each embedding dimension of input tokens. As shown in Tabel IX, the performance of the two methods is almost the same on CIFAR10. When evaluated on CIFAR100, the token-wise dynamic thresholds can significantly enhance the performance of BinaryCCT, which is 2.42% higher than channel-wise dynamic thresholds on DyBinaryCCT. The experimental result illustrates that token-wise thresholds are suitable for the transformer block to process feature information and retain the representation of each token. Thus, we select token-wise dynamic thresholds in our method.

#### TABLE VII

| Number of transformer block | embed dim | full-precision | binarized | OPs |
|-----------------------------|-----------|----------------|-----------|-----|
|                            |           | top-1          | top-5     |     |
| CCT-7/7x2                   | 7         | 70.17%         | 90.06%    | 35.65% | 60.11% | 1.80G/0.95G |
| CCT-7/7x2                   | 7         | 77.62%         | 93.83%    | 41.62% | 66.11% | 2.78G/0.97G |
| CCT-7/7x2                   | 7         | 80.35%         | 95.03%    | 47.72% | 72.02% | 4.09G/0.99G |
| CCT-14/7x2                  | 14        | 80.83%         | 95.28%    | 38.22% | 63.89% | 5.40G/1.01G |

Red denotes the FLOPs of full-precision model and blue denotes the OPs after binarization.

#### TABLE VIII

| Model       | Top-1 Acc | Top-5 Acc | BOPs  | FLOPs  | OPs    |
|-------------|-----------|-----------|-------|--------|--------|
| Full-precision | 80.35%     | 95.03%    | -     | 4.10G  | 4.10G  |
| BinaryCCT   | 47.72%     | 72.02%    | 3.15G | 0.94G  | 0.99G  |
| DyBinaryCCT | 56.09%     | 79.14%    | 3.15G | 0.95G  | 0.99G  |

#### TABLE IX

|                     | static_channel | static_token | dynamic_channel | dynamic_token | CIFAR10 | CIFAR100 |
|---------------------|----------------|--------------|-----------------|---------------|---------|----------|
| DyBinaryCCT_6       | ✓              | ✓            | ✓               |               | 78.50%  | 53.59%   |
|                     |                |              |                 |               | 80.52%  | 55.92%   |
|                     |                |              |                 |               | 85.10%  | 60.52%   |
|                     |                |              |                 |               | 84.94%  | 61.60%   |
| DyBinaryCCT_7       | ✓              | ✓            | ✓               |               | 78.67%  | 52.42%   |
|                     |                |              |                 |               | 80.65%  | 56.98%   |
|                     |                |              |                 |               | 85.37%  | 59.68%   |
|                     |                |              |                 |               | 84.98%  | 62.10%   |

3) The impact of DySign in Transformer block: In this section, we provided a detail analysis of the DySign. The ablation study is conducted on CIFAR10 and CIFAR100. To demonstrate the superiority of architecture, we also compared with RSign introduced from [43]. As shown in Table IX, the
method can significantly improve the performance of BCNN and BinaryCCT with a limited computational cost increase.

**Future works.** During our experiment, we observe two points: 1) binarizing $1 \times 1$ convolution and linear layer results in significant degradation of model accuracy. 2) The knowledge of the full-precision teacher model cannot effectively transfer to the binarized student model. We assume that 1) low-capacity $1 \times 1$ convolution and linear layer fail to process complicated semantic features; 2) the binarized student model is hard to mimic the distribution of a full-precision network. In the future, we will consider improving the performance of binarized vision architecture from these two points.

**References**

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[4] Y. Li, J. Wei, Y. Liu, J. Kauttunen, and G. Zhao, “Deep learning for micro-expression recognition: A survey,” IEEE Transactions on Affective Computing, 2022.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in ICLR, 2021.

[6] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu et al., “A survey on vision transformer,” IEEE transactions on pattern analysis and machine intelligence, 2022.

[7] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, “Training data-efficient image transformers & distillation through attention,” in ICML, 2021.

[8] J. Hu, L. Shen, and G. Sun, “Squeeze-and-Excitation networks,” in CVPR, 2018.

[9] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: analysis, applications, and prospects,” IEEE transactions on neural networks and learning systems, 2021.

[10] N. Carion, F. Massa, C. Munteanu, N. Usunier, A. Kirillov, and S. Zhou, “SegFormer: simple and efficient design for semantic segmentation with weights and activations constrained to $\pm 1$,” arXiv:1602.02830, 2016.

[11] [Z. Dai, H. Liu, Q. Le, and M. Tan, “CoAttNet: marrying convolution and activations constrained to $\pm 1$,” arXiv:2107.01378, 2021.

[12] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikäinen, and L. Liu, “Pixel difference networks for efficient edge detection,” in ICCV, 2021.

[13] H. Zhang, W. Hu, and X. Wang, “ParC-Net: position aware circular convolution with merits from convnets and transformer,” in ECCV, 2022.

[14] Y. Chen, X. Dai, D. Chen, M. Liu, X. Dong, L. Yuan, and Z. Liu, “Mobileformer: Bridging mobilenet and transformer,” in CVPR, 2022.

[15] Y. Tay, D. Bahri, D. Metzler, D. C. Juan, Z. Zhao, and C. Zheng, “Synthesizer: rethinking self-attention for transformer model,” in ICLR, 2021.

[16] S. Zhang, W. Talbott, N. Srivastava, C. Huang, H. Goh, R. Zhang, and J. Susskind, “An attention free transformer,” arXiv preprint arXiv:2105.14103, 2021.

[17] Q. Zhang and Y. Yang, “ResT: an efficient transformer for visual recognition,” in NeurIPS, 2021.

[18] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” arXiv preprint arXiv:2103.13630, 2021.

[19] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural network: Training deep neural networks with weights and activations constrained to $\pm 1$,” arXiv:1602.02830, 2016.

[20] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: imagerent classification using binary convolutional neural networks,” in ECCV, 2016.

[21] Z. Liu, W. Luo, B. Wu, X. Yang, W. Liu, and K. Cheng, “Bi-Real net: binarizing deep network towards real-network performance,” International Journal of Computer Vision, pp. 1–18, 2020.

[22] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, “Binary neural networks: A survey,” Pattern Recognition, vol. 105, p. 107281, 2020.

[23] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients,” arXiv:1606.01610, 2016.

[24] J. Wang, Y. Wang, Z. Yang, L. Yang, and Y. Guo, “Bi-GCN: binary graph convolutional network,” in CVPR, 2021.

[25] R. Ding, T.-W. Chin, Z. Liu, and D. Marcelescu, “Regularizing activation distribution for training binarized deep networks,” in CVPR, 2019.

[26] Z. Liu, Z. Shen, M. Savvides, and K. T. Cheng, “ReActNet: towards precise binary neural network with generalized activation functions,” ECCV, 2020.

[27] L. Hou, Q. Yao, and J. T. Kwok, “Loss-aware binarization of deep networks,” in ICLR, 2017.

[28] Y. Shang, D. Xu, Z. Zong, and Y. Yan, “Network binarization via contrastive learning,” in ECCV, 2022.

[29] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. Song, “Forward and backward information retention for accurate binary neural networks,” in CVPR, 2020.

[30] Y. Xu, K. Han, C. Xu, Y. Tang, C. Xu, and Y. Wang, “Learning frequency domain approximation for binary neural networks,” in NeurIPS, 2021.

[31] P. Yin, S. Zhang, J. Lu, S. Osher, Y. Qi, and J. Xin, “Blended coarse gradient descent for full quantization of deep neural networks,” Research in the Mathematical Sciences, vol. 6, no. 1, pp. 1–23, 2019.

[32] A. Hassan, S. Walton, N. Shah, A. Abduweili, J. Li, and H. Shi, “Escaping the big data paradigm with compact transformers,” arXiv:2104.05704, 2021.

[33] Z. Dai, H. Liu, Q. Le, and M. Tan, “CoAtNet: marrying convolution and attention for all data sizes,” in NeurIPS, 2021.

[34] Y. Guo, K. Han, H. Wu, C. Xu, Y. Tang, C. Xu, and Y. Wang, “CMT Convolutional neural networks meet vision transformers,” in CVPR, 2022.

[35] T. Xiao, M. Singh, E. Mintun, T. Durrell, P. Dollár, and R. Girshick, “Early convolutions help transformers see better,” in NeurIPS, 2021.
