Scalable Vision Transformers with Hierarchical Pooling

Zizheng Pan   Bohan Zhuang[†]   Jing Liu   Haoyu He   Jianfei Cai

Dept of Data Science and AI, Monash University

Abstract

The recently proposed Visual image Transformers (ViT) with pure attention have achieved promising performance on image recognition tasks, such as image classification. However, the routine of the current ViT model is to maintain a full-length patch sequence during inference, which is redundant and lacks hierarchical representation. To this end, we propose a Hierarchical Visual Transformer (HVT) which progressively pools visual tokens to shrink the sequence length and hence reduces the computational cost, analogous to the feature maps downsampling in Convolutional Neural Networks (CNNs). It brings a great benefit that we can increase the model capacity by scaling dimensions of depth/width/resolution/patch size without introducing extra computational complexity due to the reduced sequence length. Moreover, we empirically find that the average pooled visual tokens contain more discriminative information than the single class token. To demonstrate the improved scalability of our HVT, we conduct extensive experiments on the image classification task. With comparable FLOPs, our HVT outperforms the competitive baselines on ImageNet and CIFAR-100 datasets. Code is available at https://github.com/MonashAI/HVT.

1. Introduction

Equipped with the self-attention mechanism that has strong capability of capturing long-range dependencies, Transformer [37] based models have achieved significant breakthroughs in many computer vision (CV) and natural language processing (NLP) tasks, such as machine translation [10, 9], image classification [11, 36], segmentation [43, 39] and object detection [3, 48]. However, the good performance of Transformers comes at a high computational cost. For example, a single Transformer model requires more than 10G Mult-Adds to translate a sentence of only 30 words. Such a huge computational complexity hinders the widespread adoption of Transformers, especially on resource-constrained devices, such as smart phones.
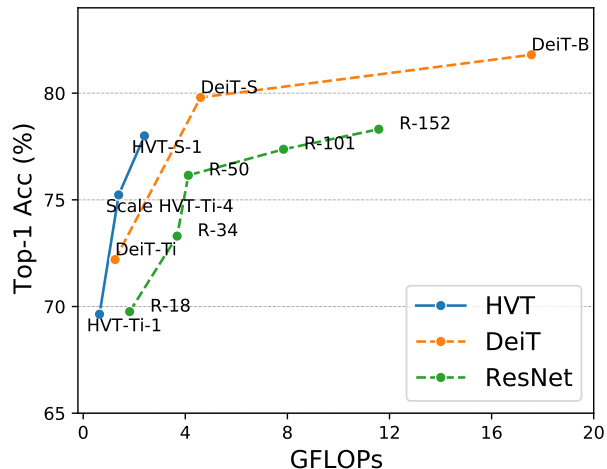

Figure 1: Performance comparisons on ImageNet. With comparable GFLOPs (1.25 vs. 1.39), our proposed Scale HVT-Ti-4 surpasses DeiT-Ti by 3.03% in Top-1 accuracy.

To improve the efficiency, there are emerging efforts to design efficient and scalable Transformers. On the one hand, some methods follow the idea of model compression to reduce the number of parameters and computational overhead. Typical methods include knowledge distillation [19], low-bit quantization [29] and pruning [12]. On the other hand, the self-attention mechanism has quadratic memory and computational complexity, which is the key efficiency bottleneck of Transformer models. The dominant solutions include kernelization [20, 28], low-rank decomposition [41], memory [30], sparsity [4] mechanisms, etc.

Despite much effort has been made, there still lacks specific efficient designs for Visual Transformers considering taking advantage of characteristics of visual patterns. In particular, ViT models maintain a full-length sequence in the forward pass across all layers. Such a design can suffer from two limitations. Firstly, different layers should have different redundancy and contribute differently to the accuracy and efficiency of the network. This statement can be supported by existing compression methods [35, 23], where each layer has its optimal spatial resolution, width and bitwidth. As a result, the full-length sequence may contain

---

[†]Corresponding author. Email: bohan.zhuang@monash.edu

huge redundancy. Secondly, it lacks multi-level hierarchical representations, which is well known to be essential for the success of image recognition tasks.

To solve the above limitations, we propose to gradually downsample the sequence length as the model goes deeper. Specifically, inspired by the design of VGG-style [33] and ResNet-style [14] networks, we partition the ViT blocks into several stages and apply the pooling operation (*e.g.*, average/max pooling) in each stage to shrink the sequence length. Such a hierarchical design is reasonable since a recent study [7] shows that a multi-head self-attention layer with a sufficient number of heads can express any convolution layers. Moreover, the sequence of visual tokens in ViT can be analogous to the flattened feature maps of CNNs along the spatial dimension, where the embedding of each token can be seen as feature channels. Hence, our design shares similarities with the spatial downsampling of feature maps in CNNs. To be emphasized, the proposed hierarchical pooling has several advantages. (1) It brings considerable computational savings and improves the scalability of current ViT models. With comparable floating-point operations (FLOPs), we can scale up our HVT by expanding the dimensions of width/depth/resolution. In addition, the reduced sequential resolution also empowers the partition of the input image into smaller patch sizes for high-resolution representations, which is needed for low-level vision and dense prediction tasks. (2) It naturally leads to the generic pyramidal hierarchy, similar to the feature pyramid network (FPN) [24], which extracts the essential multi-scale hidden representations for many image recognition tasks.

In addition to hierarchical pooling, we further propose to perform predictions without the class token. Inherited from NLP, conventional ViT models [11, 36] equip with a trainable class token, which is appended to the input patch tokens, then refined by the self-attention layers, and is finally used for prediction. However, we argue that it is not necessary to rely on the extra class token for image classification. To this end, we instead directly apply average pooling over patch tokens and use the resultant vector for prediction, which achieves improved performance. We are aware of a concurrent work [6] that also observes the similar phenomenon.

Our contributions can be summarized as follows:

- We propose a hierarchical pooling regime that gradually reduces the sequence length as the layer goes deeper, which significantly improves the scalability and the pyramidal feature hierarchy of Visual Transformers. The saved FLOPs can be utilized to improve the model capacity and hence the performance.

- Empirically, we observe that the average pooled visual tokens contain richer discriminative patterns than the class token for classification.

- Extensive experiments show that, with comparable FLOPs, our HVT outperforms the competitive baseline DeiT on image classification benchmarks, including ImageNet and CIFAR-100.

2. Related Work

Visual Transformers. The powerful multi-head self-attention mechanism has motivated the studies of applying Transformers on a variety of CV tasks. In general, current Visual Transformers can be mainly divided into two categories. The first category seeks to combine convolution with self-attention. For example, Carion *et al*. [3] propose DETR for object detection, which firstly extracts visual features with CNN backbone, followed by the feature refinement with Transformer blocks. BotNet [34] is a recent study that replaces the convolution layers with multiheaded self-attention layers at the last stage of ResNet. Other works [48, 18] also present promising results with this hybrid architecture. The second category aims to design a pure attention-based architecture without convolutions. Recently, Ramachandran *et al*. [27] propose a model which replaces all instances of spatial convolutions with a form of self-attention applied to ResNet. Hu *et al*. [17] propose LR-Net [17] that replaces convolution layers with local relation layers, which adaptively determines aggregation weights based on the compositional relationship of local pixel pairs. Axial-DeepLab [40] is also proposed to use Axial-Attention [16], a generalization form of self-attention, for Panoptic Segmentation. Dosovitskiy *et al*. [11] first transfers Transformer to image classification. The model inherits a similar architecture from standard Transformer in NLP and achieves promising results on ImageNet, whereas it suffers from prohibitively expensive training complexity. To solve this, the following work DeiT [36] propose a more advanced optimization strategy and a distillation token, with improved accuracy and training efficiency. Moreover, T2T-ViT [45] aims to overcome the limitations of simple tokenization of input images in ViT and propose to progressively structurize the image to tokens to capture rich local structural patterns. Nevertheless, the previous literature all assumes the same architecture to the NLP task, without the adaptation to the image recognition tasks. In this paper, we propose several simple yet effective modifications to improve the scalability of current ViT models.

Efficient Transformers. Transformer-based models are resource-hungry and compute-intensive despite their state-of-the-art performance. We roughly summarize the efficient Transformers into two categories. The first category focuses on applying generic compression techniques to speed up the inference, either based on quantization [47], pruning [26, 12], and distillation [32] or seeking to use Neu-
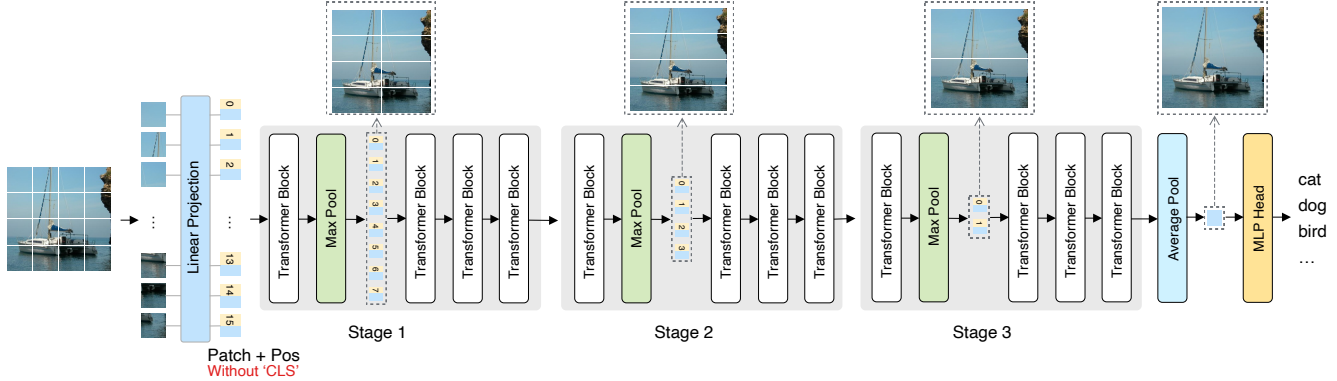
Figure 2: Overview of the proposed Hierarchical Visual Transformer. To reduce the redundancy in the full-length patch sequence and construct a hierarchical representation, we propose to progressively pool visual tokens to shrink the sequence length. To this end, we partition the ViT [11] blocks into several stages. At each stage, we insert a pooling layer after the first Transformer block to perform down-sampling. In addition to the pooling layer, we perform predictions using the resultant vector of average pooling the output visual tokens of the last stage instead of the class token only.

ral Architecture Search (NAS) [38] to explore better configurations. Another category aims to solve the quadratic complexity issue of the self-attention mechanism. A representative approach [5, 20] is to express the self-attention weights as a linear dot-product of kernel functions and make use of the associative property of matrix products to reduce the overall self-attention complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. Moreover, some works alternatively study diverse sparse patterns of self-attention [4, 21], or consider the low-rank structure of the attention matrix [41], leading to linear time and memory complexity with respect to the sequence length. There are also some NLP literatures that tend to reduce the sequence length during processing. For example, Goyal *et al*. [13] propose PoWER-BERT, which progressively eliminates word tokens during the forward pass. Funnel-Transformer [8] presents a pool-query-only strategy, pooling the query vector within each self-attention layer. However, there are few literatures targeting improving the efficiency of the ViT models.

To compromise FLOPs, current ViT models divide the input image into coarse patches (*i.e.*, large patch size), hindering their generalization to dense predictions. In order to bridge this gap, we propose a general hierarchical pooling strategy that significantly reduces the computational cost while enhancing the scalability of important dimensions of the ViT architectures, *i.e.*, depth, width, resolution and patch size. Moreover, our generic encoder also inherits the pyramidal feature hierarchy from classic CNNs, potentially benefiting many downstream recognition tasks. Also note that different from a concurrent work [42] which applies 2D patch merging, this paper introduces the feature hierarchy with 1D pooling. We discuss the impact of 2D pooling in Section 5.2.

3. Proposed Method

In this section, we first briefly revisit the preliminaries of Visual Transformers [11] and then introduce our proposed Hierarchical Visual Transformer.

3.1. Preliminary

Let $\mathcal{I} \in \mathbb{R}^{H \times W \times C}$ be an input image, where H, W and C represent the height, width, and the number of channels, respectively. To handle a 2D image, ViT first splits the image into a sequence of flattened 2D patches $\mathbf{X} = [\mathbf{x}_p^1; \mathbf{x}_p^2; ...; \mathbf{x}_p^N]$, where $\mathbf{x}_p^i \in \mathbb{R}^{P^2 C}$ is the i-th patch of the input image and $[\cdot]$ is the concatenation operation. Here, $N = HW/P^2$ is the number of patches and P is the size of each patch. ViT then uses a trainable linear projection that maps each vectorized patch to a D dimension patch embedding. Similar to the class token in BERT [10], ViT prepends a learnable embedding $\mathbf{x}_{cls} \in \mathbb{R}^D$ to the sequence of patch embeddings. To retain positional information, ViT introduces an additional learnable positional embeddings $\mathbf{E} \in \mathbb{R}^{(N+1) \times D}$. Mathematically, the resulting representation of the input sequence can be formulated as

$$\mathbf{X}_0 = [\mathbf{x}_{cls}; \mathbf{x}_p^1 \mathbf{W}; \mathbf{x}_p^2 \mathbf{W}; ...; \mathbf{x}_p^N \mathbf{W}] + \mathbf{E}, \qquad (1)$$

where $\mathbf{W} \in \mathbb{R}^{P^2 C \times D}$ is a learnable linear projection parameter. Then, the resulting sequence of embeddings serves as the input to the Transformer encoder [37].

Suppose that the encoder in a Transformer consists of L blocks. Each block contains a multi-head self-attention (MSA) layer and a position-wise multi-layer perceptron (MLP). For each layer, layer normalization (LN) [1] and residual connections [14] are employed, which can be for-

mulated as follows

$$\mathbf{X}'_{l-1} = \mathbf{X}_{l-1} + \text{MSA}(\text{LN}(\mathbf{X}_{l-1})), \qquad (2)$$

$$\mathbf{X}_l = \mathbf{X}'_{l-1} + \text{MLP}(\text{LN}(\mathbf{X}'_{l-1})), \qquad (3)$$

where $l \in [1, ..., L]$ is the index of Transformer blocks. Here, a MLP contains two fully-connected layers with a GELU non-linearity [15]. In order to perform classification, ViT applies a layer normalization layer and a fully-connected (FC) layer to the first token of the Transformer encoder's output $\mathbf{X}_L^0$. In this way, the output prediction $\mathbf{y}$ can be computed by

$$\mathbf{y} = \text{FC}(\text{LN}(\mathbf{X}_L^0)). \qquad (4)$$

3.2. Hierarchical Visual Transformer

In this paper, we propose a Hierarchical Visual Transformer (HVT) to reduce the redundancy in the full-length patch sequence and construct a hierarchical representation. In the following, we first propose a hierarchical pooling to gradually shrink the sequence length and hence reduce the computational cost. Then, we propose to perform predictions without the class token. The overview of the proposed HVT is shown in Figure 2.

3.2.1 Hierarchical Pooling

We propose to apply hierarchical pooling in ViT for two reasons: (1) Recent studies [13, 8] on Transformers show that tokens tend to carry redundant information as it goes deeper. Therefore, it would be beneficial to reduce these redundancies through the pooling approaches. (2) The input sequence projected from image patches in ViT can be seen as flattened CNN feature maps with encoded spatial information, hence pooling from the nearby tokens can be analogous to the spatial pooling methods in CNNs.

Motivated by the hierarchical pipeline of VGG-style [33] and ResNet-style [14] networks, we partition the Transformer blocks into M stages and apply downsampling operation to each stage to shrink the sequence length. Let $\{b_1, b_2, \ldots, b_M\}$ be the indexes of the first block in each stage. At the m-th stage, we apply a 1D max pooling operation with a kernel size of k and stride of s to the output of the Transformer block $b_m \in \{b_1, b_2, \ldots, b_M\}$ to shrink the sequence length.

Note that the positional encoding is important for a Transformer since the positional encoding is able to capture information about the relative and absolute position of the token in the sequence [37, 3]. In Eq. (1) of ViT, each patch is equipped with positional embedding $\mathbf{E}$ at the beginning. However, in our HVT, the original positional embedding $\mathbf{E}$ may no longer be meaningful after pooling since the sequence length is reduced after each pooling operation. In this case, positional embedding in the pooled sequence

needs to be updated. Moreover, previous work [8] in NLP also find it important to complement positional information after changing the sequence length. Therefore, at the m-th stage, we introduce an additional learnable positional embedding $\mathbf{E}_{b_m}$ to capture the positional information, which can be formulated as

$$\hat{\mathbf{X}}_{b_m} = \text{MaxPool1D}(\mathbf{X}_{b_m}) + \mathbf{E}_{b_m}, \qquad (5)$$

where $\mathbf{X}_{b_m}$ is the output of the Transformer block b_m. We then forward the resulting embeddings $\hat{\mathbf{X}}_{b_m}$ into the next Transformer block $b_m + 1$.

3.2.2 Prediction without the Class Token

Previous works [11, 36] make predictions by taking the class token as input in classification tasks as described in Eq. (4). However, such structure relies solely on the single class token with limited capacity while discarding the remaining sequence that is capable of storing more discriminative information. To this end, we propose to remove the class token in the first place and predict with the remaining output sequence on the last stage.

Specifically, given the output sequence without the class token on the last stage $\mathbf{X}_L$, we first apply average pooling, then directly apply an FC layer on the top of the pooled embeddings and make predictions. The process can be formulated as

$$\mathbf{y} = \text{FC}(\text{AvgPool}(\text{LN}(\mathbf{X}_L))). \qquad (6)$$

3.3. Complexity Analysis

In this section, we analyse the block-wise compression ratio with hierarchical pooling. Following ViT [11], we use FLOPs to measure the computational cost of a Transformer. Let n be the number of tokens in a sequence and d is the dimension of each token. The FLOPs of a Transformer block $\phi_{BLK}(n, d)$ can be computed by

$$\begin{aligned} \phi_{BLK}(n, d) &= \phi_{MSA}(n, d) + \phi_{MLP}(n, d), \\ &= 12nd^2 + 2n^2d, \end{aligned} \qquad (7)$$

where $\phi_{MSA}(n, d)$ and $\phi_{MLP}(n, d)$ are the FLOPs of the MSA and MLP, respectively. Details about Eq. (7) can be found in the supplementary material.

Without loss of generality, suppose that the sequence length n is reduced by half after performing hierarchical pooling. In this case, the block-wise compression ratio α can be computed by

$$\alpha = \frac{\phi_{BLK}(n, d)}{\phi_{BLK}(n/2, d)} = 2 + \frac{2}{12(d/n) + 1}. \qquad (8)$$

Clearly, Eq. (8) is monotonic, thus the block-wise compression ratio α is bounded by $(2, 4)$, *i.e.*, $\alpha \in (2, 4)$.
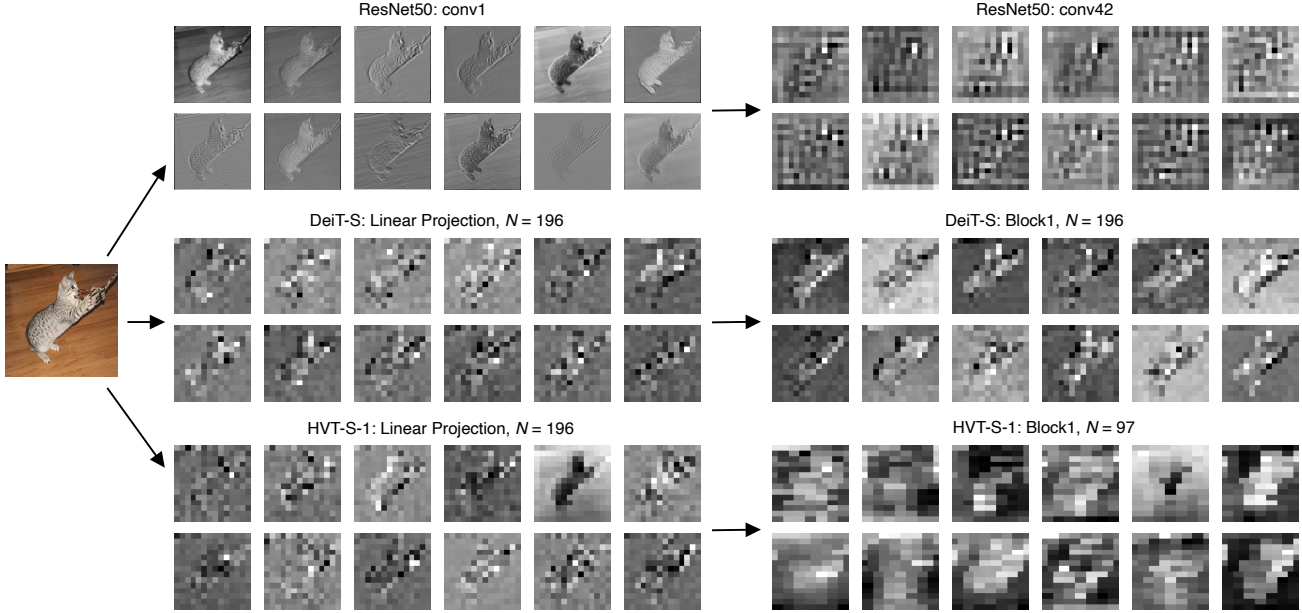
Figure 3: Feature visualization of ResNet50 [14], DeiT-S [36] and our HVT-S-1 trained on ImageNet. DeiT-S and our HVT-S-1 correspond to the small setting in DeiT, except that our model applies a pooling operation and performing predictions without the class token. The resolution of the feature maps from ResNet50 conv1 and conv4_2 are 112×112 and 14×14, respectively. For DeiT and HVT, the feature maps are reshaped from tokens. For our model, we interpolate the pooled sequence to its initial length then reshape it to a 2D map.

4. Discussions

4.1. Analysis of Hierarchical Pooling

In CNNs, feature maps are usually downsampled to smaller sizes in a hierarchical way [33, 14]. In this paper, we show that this principle can be applied to ViT models by comparing the visualized feature maps from ResNet conv4_2, DeiT-S [36] block1 and HVT-S-1 block1 in Figure 3. From the figure, in ResNet, the initial feature maps after the first convolutional layer contain rich edge information. After feeding the features to consecutive convolutional layers and a pooling layer, the output feature maps tend to preserve more high-level discriminative information. In DeiT-S, following the ViT structure, although the image resolution for the feature maps has been reduced to 14×14 by the initial linear projection layer, we can still observe clear edges and patterns. Then, the features get refined in the first block to obtain sharper edge information. In contrast to DeiT-S that refines features at the same resolution level, after the first block, the proposed HVT downsamples the hidden sequence through a pooling layer and reduces the sequence length by half. We then interpolate the sequence back to 196 and reshape it to 2D feature maps. We can find that the hidden representations contain more abstract information with high discriminative power, which is similar to ResNet.

4.2. Scalability of HVT

The computational complexity reduction equips HVT with strong scalability in terms of width/depth/patch size/resolution. Take DeiT-S for an example, the model consists of 12 blocks and 6 heads. Given a 224×224 image with a patch size of 16, the computational cost of DeiT-S is around 4.6G FLOPs. By applying four pooling operations, our method is able to achieve nearly $3.3 \times$ FLOPs reduction. Furthermore, to re-allocate the reduced FLOPs, we may construct wider or deeper HVT-S, with 11 heads or 48 blocks, then the overall FLOPs would be around 4.51G and 4.33G, respectively. Moreover, we may consider a longer sequence by setting a smaller patch size or using a larger resolution. For example, with a patch size of 8 and an image resolution of 192×192, the FLOPs for HVT-S is around 4.35G. Alternatively, enlarging the image resolution into 384×384 will lead to 4.48G FLOPs. In all of the above mentioned cases, the computational costs are still lower than that of DeiT-S while the model capacity is enhanced.

It is worth noting that finding a principled way to scale up HVT to obtain the optimal efficiency-vs-accuracy trade-off remains an open question. At the current stage, we take an early exploration by evenly partitioning blocks and following model settings in DeiT [36] for a fair comparison. In fact, the improved scalability of HVT makes it possible for using Neural Architecture Search (NAS) to automatically find optimal configurations, such as EfficientNet [35]. We

leave for more potential studies for future work.

5. Experiments

Compared methods. To investigate the effectiveness of HVT, we compare our method with DeiT [36] and a BERT-based pruning method PoWER-BERT [13]. DeiT is a representative Vision Transformer and PoWER progressively prunes unimportant tokens in pretrained BERT models for inference acceleration. Moreover, we consider two architectures in DeiT for comparisons: **HVT-Ti**: HVT with the tiny setting. **HVT-S**: HVT with the small setting. For convenience, we use "Architecture-M" to represent our model with M pooling stages, *e.g.*, HVT-S-1.

Datasets and Evaluation metrics. We evaluate our proposed HVT on two image classification benchmark datasets: CIFAR-100 [22] and ImageNet [31]. We measure the performance of different methods in terms of the Top-1 and Top-5 accuracy. Following DeiT [36], we measure the computational cost by FLOPs. Moreover, we also measure the model size by the number of parameters (Params).

Implementation details. For experiments on ImageNet, we train our models for 300 epochs with a total batch size of 1024. The initial learning rate is 0.0005. We use AdamW optimizer [25] with a momentum of 0.9 for optimization. We set the weight decay to 0.025. For fair comparisons, we keep the same data augmentation strategy as DeiT [36]. For the downsampling operation, we use max pooling by default. The kernel size k and stride s are set to 3 and 2, respectively, chosen by a simple grid search on CIFAR100. Besides, all learnable positional embeddings are initialized in the same way as DeiT. More detailed settings on the other hyper-parameters can be found in DeiT. For experiments on CIFAR-100, we train our models with a total batch size of 128. The initial learning rate is set to 0.000125. Other hyper-parameters are kept the same as those on ImageNet.

5.1. Main Results

We compare the proposed HVT with DeiT and PoWER, and report the results in Table 1. First, compared to DeiT, our HVT achieves nearly $2\times$ FLOPs reduction with a hierarchical pooling. However, the significant FLOPs reduction also leads to performance degradation in both the tiny and small settings. Additionally, the performance drop of HVT-S-1 is smaller than that of HVT-Ti-1. For example, for HVT-S-1, it only leads to 1.80% drop in the Top-1 accuracy. In contrast, it results in 2.56% drop in the Top-1 accuracy for HVT-Ti-1. It can be attributed to that, compared with HVT-Ti-1, HVT-S-1 is more redundant with more parameters. Therefore, applying hierarchical pooling to HVT-S-1
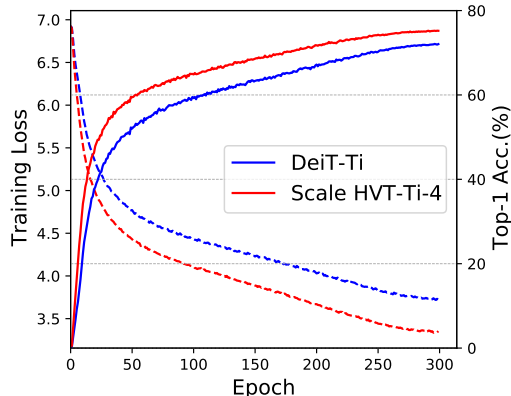

Figure 4: Performance comparisons of DeiT-Ti (1.25G FLOPs) and the proposed Scale HVT-Ti-4 (1.39G FLOPs). All the models are evaluated on ImageNet. Solid lines denote the Top-1 accuracy (y-axis on the right). Dash lines denote the training loss (y-axis on the left).

can significantly reduce redundancy while maintaining performance. Second, compared to PoWER, HVT-Ti-1 uses less FLOPs while achieving better performance. Besides, HVT-S-1 reduces more FLOPs than PoWER, while achieving slightly lower performance than PoWER. Also note that PoWER involves three training steps, while ours is a simpler one-stage training scheme.

Moreover, we also compare the scaled HVT with DeiT under similar FLOPs. Specifically, we enlarge the embedding dimensions and add extra heads in HVT-Ti. From Table 1 and Figure 4, by re-allocating the saved FLOPs to scale up the model, HVT can converge to a better solution and yield improved performance. For example, the Top-1 accuracy on ImageNet can be improved considerably by 3.03% in the tiny setting. More empirical studies on the effect of model scaling can be found in Section 5.2.

5.2. Ablation Study

Effect of the prediction without the class token. To investigate the effect of the prediction without the class token, we train DeiT-Ti with and without the class token and show the results in Table 2. From the results, the models without the class token outperform the ones with the class token. The performance gains mainly come from the extra discriminative information stored in the entire sequence without the class token. Note that the performance improvement on CIFAR-100 is much larger than that on ImageNet. It may be attributed that CIFAR-100 is a small dataset, which lacks varieties compared with ImageNet. Therefore, the model trained on CIFAR-100 benefits more from the increase of model's discriminative power.

Effect of different pooling stages. We train HVT-S with different pooling stages $M \in \{0, 1, 2, 3, 4\}$ and show the results in Table 4. Note that HVT-S-0 is equivalent to the

Table 1: Performance comparisons with DeiT and PoWER on ImageNet. "Embedding Dim" refers to the dimension of each token in the sequence. "#Heads" and "#Blocks" are the number of self-attention heads and blocks in Transformer, respectively. "FLOPs" is measured with a 224×224 image. "Ti" and "S" are short for the tiny and small settings, respectively. "Architecture-M" denotes the model with M pooling stages. "Scale" denotes that we scale up the embedding dimension and/or the number of self-attention heads. "DeiT-Ti/S + PoWER" refers to the model that applies the techniques in PoWER-BERT [13] to DeiT-Ti/S.

Model	Embedding Dim	#Heads	#Blocks	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
DeiT-Ti [36]	192	3	12	1.25	5.72	72.20	91.10
DeiT-Ti + PoWER [13]	192	3	12	0.80	5.72	69.40 (-2.80)	89.20 (-1.90)
HVT-Ti-1	192	3	12	0.64	5.74	69.64 (-2.56)	89.40 (-1.70)
Scale HVT-Ti-4	384	6	12	1.39	22.12	75.23 (+3.03)	92.30 (+1.20)
DeiT-S [36]	384	6	12	4.60	22.05	79.80	95.00
DeiT-S + PoWER [13]	384	6	12	2.70	22.05	78.30 (-1.50)	94.00 (-1.00)
HVT-S-1	384	6	12	2.40	22.09	78.00 (-1.80)	93.83 (-1.17)

Table 2: Effect of the prediction without the class token. "CLS" denotes the class token.

Model	FLOPs (G)	Params (M)	ImageNet		CIFAR-100	
			Top-1 Acc. (%)	Top-5 Acc. (%)	Top-1 Acc. (%)	Top-5 Acc. (%)
DeiT-Ti with CLS	1.25	5.72	72.20	91.10	64.49	89.27
DeiT-Ti without CLS	1.25	5.72	72.42 (+0.22)	91.55 (+0.45)	65.93 (+1.44)	90.33 (+1.06)

Table 3: Performance comparisons on HVT-S-4 with three downsampling operations: convolution, max pooling and average pooling. We report the Top-1 and Top-5 accuracy on CIFAR-100.

Model	Operation	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
HVT-S	Conv	1.47	23.54	69.75	92.12
HVT-S	Avg	1.39	21.77	70.38	91.39
HVT-S	Max	1.39	21.77	75.43	93.56

Table 4: Performance comparisons on HVT-S with different pooling stages M. We report the Top-1 and Top-5 accuracy on CIFAR-100.

M	FLOPs	Params	ImageNet		CIFAR100	
			Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
0	4.57	21.70	80.39	95.13	71.99	92.44
1	2.40	21.74	78.00	93.83	74.27	93.07
2	1.94	21.76	77.36	93.55	75.37	93.69
3	1.62	21.77	76.32	92.90	75.22	93.90
4	1.39	21.77	75.23	92.30	75.43	93.56

Table 5: Performance comparisons on HVT-S-4 with different number of Transformer blocks. We report the Top-1 and Top-5 accuracy on CIFAR-100.

#Blocks	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
12	1.39	21.77	75.43	93.56
16	1.72	28.87	75.32	93.30
20	2.05	35.97	75.35	93.35
24	2.37	43.07	75.04	93.39

Table 6: Performance comparisons on HVT-Ti-4 with different number of self-attention heads. We report the Top-1 and Top-5 accuracy on CIFAR-100.

#Heads	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
3	0.38	5.58	69.51	91.78
6	1.39	21.77	75.43	93.56
12	5.34	86.01	76.26	93.39
16	9.39	152.43	76.30	93.16

DeiT-S without the class token. With the increase of M, HVT-S achieves better performance with decreasing FLOPs on CIFAR-100, while on ImageNet we observe the accuracy degrades. One possible reason is that HVT-S is very redundant on CIFAR-100, such that pooling acts as a regularizer to avoid the overfitting problem and improves the generalization of HVT on CIFAR-100. On ImageNet, we assume HVT is less redundant and a better scaling strategy is required to improve the performance.

Effect of different downsampling operations. To investigate the effect of different downsampling operations, we train HVT-S-4 with three downsampling strategies: convolution, average pooling and max pooling. As Table 3 shows, downsampling with convolution performs the worst even it

introduces additional FLOPs and parameters. Besides, average pooling performs slightly better than convolution in terms of the Top-1 accuracy. Compared with the two settings, HVT-S-4 with max pooling performs much better as it significantly surpasses average pooling by 5.05% in the Top-1 accuracy and 2.17% in the Top-5 accuracy. The result is consistent with the common sense [2] that max pooling performs well in a large variety of settings. To this end, we use max pooling in all other experiments by default.

Effect of model scaling. One of the important advantages of the proposed hierarchical pooling is that we can re-allocate the saved computational cost for better model capacity by constructing a model with a wider, deeper, larger resolution or smaller patch size configuration. Similar to the

CNNs literature [14, 44, 46], we study the effect of model scaling in the following.

Based on HVT-S-4, we first construct deeper models by increasing the number of blocks in Transformers. Specifically, we train 4 models with different number of blocks $L \in \{12, 16, 20, 24\}$. As a result, each pooling stage for different models would have 3, 4, 5, and 6 blocks, respectively. We train 4 models on CIFAR-100 and report the results in Table 5. From the results, we observe no more gains by stacking more blocks in HVT.

Based on HVT-Ti-4, we then construct wider models by increasing the number of self-attention heads. To be specific, we train 4 models with different numbers of self-attention heads, i.e., 3, 6, 12, and 16, on CIFAR-100 and report the results in Table 6. From the results, our models achieve better performance with the increase of width. For example, the model with 16 self-attention heads outperforms those with 3 self-attention heads by 6.79% in the Top-1 accuracy and 1.38% in the Top-5 accuracy.

Based on HVT-S-4, we further construct models with larger input image resolutions. Specifically, we train 4 models with different input image resolutions, i.e., 160, 224, 320, and 384, on CIFAR-100 and report the results in Table 7. From the results, with the increase of image resolution, our models achieve better performance. For example, the model with the resolution of 384 outperforms those with the resolution of 160 by 2.47% in the Top-1 accuracy and 1.12% in the Top-5 accuracy. Nevertheless, increasing image resolutions also leads to high computational cost. To make a trade-off between computational cost and accuracy, we set the image resolution to 224 by default.

We finally train HVT-S-4 with different patch sizes $P \in \{8, 16, 32\}$ and show the results in Table 8. From the results, HVT-S-4 performs better with the decrease of patch size. For example, when the patch size decreases from 32 to 8, our HVT-S achieves 9.14% and 4.03% gain in terms of the Top-1 and Top-5 accuracy. Intuitively, a smaller patch size leads to fine-grained image patches and helps to learn high-resolution representations, which is able to improve the classification performance. However, with a smaller patch size, the patch sequence will be longer, which significantly increases the computational cost. To make a balance between the computational cost and accuracy, we set the patch size to 16 by default.

Exploration on 2D pooling. Compared to 1D pooling, 2D pooling brings more requirements. For example, it requires a smaller patch size to ensure a sufficient sequence length. Correspondingly, it is essential to reduce the heads at the early stages to save FLOPs and memory consumption from high-resolution feature maps. Besides, it also requires to vary the blocks at each stage to control the overall model complexity. In Table 9, we apply 2D pooling to HVT-S-2

Table 7: Performance comparisons on HVT-S-4 with different image resolutions. We report the Top-1 and Top-5 accuracy on CIFAR-100.

Resolution	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
160	0.69	21.70	73.84	92.90
224	1.39	21.77	75.43	93.56
320	3.00	21.92	75.54	94.18
384	4.48	22.06	76.31	94.02

Table 8: Performance comparisons on HVT-S-4 with different patch sizes P. We report the Top-1 and Top-5 accuracy on CIFAR-100.

P	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
8	6.18	21.99	77.29	94.22
16	1.39	21.77	75.43	93.56
32	0.37	22.55	68.15	90.19

Table 9: Effect of 2D pooling on HVT-S-2. We report the Top-1 and Top-5 accuracy on CIFAR-100. For HVT-S-2, we apply 2D max pooling and use a patch size of 8.

Model	FLOPs (G)	Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
DeiT-S	4.60	21.70	71.99	92.44
HVT-S-2 (2D)	4.62	21.80	77.58	94.40

and compare it with DeiT-S. The results show that HVT-S-2 with 2D pooling outperforms DeiT-S on CIFAR100 by a large margin with similar FLOPs. In this case, we assume that HVT can achieve promising performance with a dedicated scaling scheme for 2D pooling. We will leave this exploration for future work.

6. Conclusion and Future Work

In this paper, we have presented a Hierarchical Visual Transformer, termed HVT, for image classification. In particular, the proposed hierarchical pooling can significantly compress the sequential resolution to save computational cost in a simple yet effective form. More importantly, this strategy greatly improves the scalability of visual Transformers, making it possible to scale various dimensions - depth, width, resolution and patch size. By re-allocating the saved computational cost, we can scale up these dimensions for better model capacity with comparable or fewer FLOPs. Moreover, we have empirically shown that the visual tokens are more important than the single class token for class prediction. Note that the scope of this paper only targets designing our HVT as an encoder. Future works may include extending our HVT model to decoder and to solve other mainstream CV tasks, such as object detection and semantic/instance segmentation. In addition, it would be interesting to find a principled way to scale up HVT that can achieve better accuracy and efficiency.

7. Acknowledgements

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[2] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pages 111–118, 2010. 7

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2, 4

[4] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 1, 3

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021. 3

[6] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *Arxiv preprint 2102.10882*, 2021. 2

[7] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020. 2

[8] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *NeurIPS*, 2020. 3, 4

[9] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019. 1

[10] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 1, 3

[11] 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, 2, 3, 4

[12] Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. Compressing BERT: studying the effects of weight pruning on transfer learning. In *RepL4NLP@ACL*, pages 143–155, 2020. 1, 2

[13] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating BERT inference via progressive word-vector elimination. In *ICML*, 2020. 3, 4, 6, 7

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 4, 5, 8

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016. 4

[16] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019. 2

[17] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019. 2

[18] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. 2

[19] 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 *EMNLP*, pages 4163–4174, 2020. 1

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, pages 5156–5165. PMLR, 2020. 1, 3

[21] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. 3

[22] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 6

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017. 1

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 2

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[26] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *NeurIPS*, pages 14014–14024, 2019. 2

[27] Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, pages 68–80, 2019. 2

[28] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. In *ICLR*, 2021. 1

[29] Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. Fully quantized transformer for machine translation. In Trevor Cohn, Yulan He, and Yang Liu, editors, *EMNLP*, pages 1–14, 2020. 1

[30] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020. 1

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 6

[32] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*, 2019. 2

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 4, 5

[34] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *CVPR*, 2021. 2

[35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 1, 5

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 1, 2, 4, 5, 6, 7

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 3, 4

[38] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. HAT: hardware-aware transformers for efficient natural language processing. In *ACL*, pages 7675–7688, 2020. 3

[39] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021. 1

[40] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan L. Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020. 2

[41] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 1, 3

[42] 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 *ICCV*, 2021. 3

[43] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *CVPR*, 2021. 1

[44] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016. 8

[45] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021. 2

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 8

[47] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. Ternarybert: Distillation-aware ultra-low bit BERT. In *EMNLP*, pages 509–521, 2020. 2

[48] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. *ICLR*, 2021. 1, 2