Bridging the Gap Between Vision Transformers and Convolutional Neural Networks on Small Datasets–Supplementary Materials

Anonymous Author(s)
Affiliation
Address
email

1 Code of Dynamic Hybrid Vision Transformer

1.1 Code of the Dynamic Aggregation Feed Forward (DAFF)

```
class DAFF(nn.Module):
    def __init__(self, in_dim, hid_dim, out_dim, kernel_size=3):
        self.conv1 = nn.Conv2d(in_dim, hid_dim, kernel_size=1,
                               stride=1, padding=0)
        self.conv2 = nn.Conv2d(
            hid_dim, hid_dim, kernel_size=3, stride=1,
            padding=(kernel_size - 1) // 2, groups=hid_dim)
        self.conv3 = nn.Conv2d(hid_dim, out_dim, kernel_size=1,
                               stride=1, padding=0)
        self.act = nn.GELU()
        self.squeeze = nn.AdaptiveAvgPool2d((1, 1))
        self.compress = nn.Linear(in_dim, in_dim//4)
        self.excitation = nn.Linear(in_dim//4, in_dim)
        self.bn1 = nn.BatchNorm2d(hid_dim)
        self.bn2 = nn.BatchNorm2d(hid_dim)
        self.bn3 = nn.BatchNorm2d(out_dim)

    def forward(self, x):
        B, N, C = x.size()
        cls_token, tokens = torch.split(x, [1, N - 1], dim=1)
        x = tokens.reshape(B, int(math.sqrt(N - 1)),
                int(math.sqrt(N - 1)), C).permute(0, 3, 1, 2)

        x = self.act(self.bn1(self.conv1(x)))
        x = x + self.act(self.bn2(self.conv2(x)))
        x = self.bn3(self.conv3(x))

        weight = self.squeeze(x).flatten(1).reshape(B, 1, C)
        weight = self.excitation(self.act(self.compress(weight)))
        cls_token = cls_token * weight
        tokens = x.flatten(2).permute(0, 2, 1)
        out = torch.cat((cls_token, tokens), dim=1)
        return out
```

[1]

[1] Code is modified from https://github.com/coeusguo/ceit

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

1.2 Code of the Sequential Overlapping Patch Embedding (SOPE)

```python
def conv3x3(in_dim, out_dim):
    return torch.nn.Sequential(
        nn.Conv2d(in_dim,out_dim,kernel_size=3,stride=2,padding=1),
        nn.BatchNorm2d(out_dim)
    )

class Affine(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.alpha = nn.Parameter(torch.ones([1, dim, 1, 1]))
        self.beta = nn.Parameter(torch.zeros([1, dim, 1, 1]))
    def forward(self, x):
        x = x * self.alpha + self.beta
        return x

class SOPE(nn.Module):
    def __init__(self, patch_size, embed_dim):
        super().__init__()
        self.pre_affine = Affine(3)
        self.post_affine = Affine(embed_dim)
        if patch_size[0] == 16:
            self.proj = torch.nn.Sequential(
                conv3x3(3, embed_dim//8, 2),
                nn.GELU(),
                conv3x3(embed_dim//8, embed_dim//4, 2),
                nn.GELU(),
                conv3x3(embed_dim//4, embed_dim//2, 2),
                nn.GELU(),
                conv3x3(embed_dim//2, embed_dim, 2),
                )
        elif patch_size[0] == 4:
            self.proj = torch.nn.Sequential(
                conv3x3(3, embed_dim//2, 2),
                nn.GELU(),
                conv3x3(embed_dim//2, embed_dim, 2),
                )
        elif patch_size[0] == 2:
            self.proj = torch.nn.Sequential(
                conv3x3(3, embed_dim, 2),
                nn.GELU(),
                )
    def forward(self, x):
        B, C, H, W = x.shape
        x = self.pre_affine(x)
        x = self.proj(x)
        x = self.post_affine(x)
        Hp, Wp = x.shape[2], x.shape[3]
        x = x.flatten(2).transpose(1, 2)
        return x
```

[2]

[2]Code is modified from https://github.com/facebookresearch/xcit

1.3 Code of the Head-Interacted Multi-Head Self-Attention (HI-MHSA)

```
class Attention(nn.Module):
    def __init__(self, dim, num_heads=8):
        super().__init__()
        self.num_heads = num_heads
        head_dim = dim // num_heads
        self.scale = head_dim ** -0.5
        self.qkv = nn.Linear(dim, dim * 3, bias=True)
        self.proj = nn.Linear(dim, dim)
        self.act = nn.GELU()
        self.ht_proj = nn.Linear(head_dim, dim,bias=True)
        self.ht_norm = nn.LayerNorm(head_dim)
        self.pos_embed = nn.Parameter(
                    torch.zeros(1, self.num_heads, dim))

    def forward(self, x):
        B, N, C = x.shape

        # head token
        head_pos = self.pos_embed.expand(x.shape[0], -1, -1)
        ht = x.reshape(B, -1, self.num_heads, C//self.num_heads).
                                            permute(0, 2, 1, 3)
        ht = ht.mean(dim=2)
        ht = self.ht_proj(ht)
                    .reshape(B, -1, self.num_heads, C//self.num_heads)
        ht = self.act(self.ht_norm(ht)).flatten(2)
        ht = ht + head_pos
        x = torch.cat([x, ht], dim=1)

        # common MHSA
        qkv = self.qkv(x).reshape(B, N+self.num_heads, 3,
                    self.num_heads, C//self.num_heads)
                    .permute(2, 0, 3, 1, 4)
        q, k, v = qkv[0], qkv[1], qkv[2]
        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = attn.softmax(dim=-1)
        attn = self.attn_drop(attn)
        x = (attn @ v).transpose(1, 2).reshape(B, N+self.num_heads, C)
        x = self.proj(x)

        # split, average and add
        cls, patch, ht = torch.split(x, [1,N-1,self.num_heads], dim=1)
        cls = cls + torch.mean(ht, dim=1, keepdim=True)
        x = torch.cat([cls, patch], dim=1)

        return x
```

2 CIFAR-100 Dataset

2.1 Fine-tuning on CIFAR-100

We analyse fine-tuning results in this section. All the models are pre-trained on ImageNet-1K only and then fine-tuned on CIFAR-100 datasets. Results are shown in Table 1. We cite the reported results from corresponding papers. When fine-tuning our DHVT, we use AdamW optimizer with cosine learning rate scheduler and 2 warm-up epochs. We use a batch size of 256, initial learning rate of 0.0005, weight decay of 1e-8, and fine-tuning epochs of 100. We fine-tune our model on image size of 224×224 and we use patch size of 16, head numbers of 3 and 6 for DHVT-T and DHVT-S respectively, the same as the model pre-trained model on ImageNet-1K.

Table 1: Pretrained on ImageNet-1K and then fine-tuned on the CIFAR-100 (top-1 accuracy, 100 fine-tuning epochs). FT Epochs denotes fine-tuning epochs, resolution denotes the image resolution on fine-tuning.

Method	GFLOPs	ImageNet-1K	FT Epochs	Img Size	CIFAR-100 Acc (%)
ResNet-50 [1]	3.8	-	100	224	85.44
ViT-B/16 [2]	18.7	77.9	10000	384	87.13
ViT-L/16 [2]	65.8	76.5	10000	384	86.35
T2T-ViT-14 [1]	5.2	81.5	100	224	87.33
Swin-T [1]	4.5	81.3	100	224	88.22
DeiT-B [3]	17.3	81.8	7200	224	90.8
DeiT-B ↑384 [3]	52.8	83.1	7200	384	90.8
CeiT-T [4]	1.2	76.4	100	224	88.4
CeiT-T ↑384 [4]	3.6	78.8	100	384	88.0
CeiT-S [4]	4.5	82.0	100	224	90.8
CeiT-S ↑384 [4]	12.9	83.3	100	384	90.8
DHVT-T (Ours)	1.4	76.5	100	224	86.73
DHVT-S (Ours)	5.1	82.3	100	224	88.87

From Table 1, we can see that our model has competitive transferable performance to Swin Transformer[5], T2T-ViT [6]. We fail in competing with DeiT [3] maybe because we only fine-tuning our model for 100 epochs. The longer epochs experiments are left for the future. And we also fail to compete with CeiT [4] under comparable computational complexity. We consider that maybe our model introduced too much inductive biases so that the fine-tuning performance is constrained also. However, the target of our method is mainly on train-from-scratch on small datasets. Thus the fine-tuning results are not so important in our consideration. And we can also see that we achieve 85.68 accuracy when training from scratch on CIFAR-100 only, as we reported in the main part of this paper. Such result even outperforms ResNet-50 pretrained on ImageNet-1k, which only reaches 85.44 accuracy when fine-tuning. DHVT is able to beat the pre-trained and fine-tuned ResNet-50 while without any pre-training, suggesting the significant performance of our model.

2.2 Computational Complexity on CIFAR-100

We empircally measure the computational complexity of our proposed DHVT on CIFAR-100 dataset. Results are shown in Table 2. We show the number of parameters, GFLOPs, throughtputs and corresponding input image size and accuracy. We use DHVT-X/Y to denote the variants of models. Here X is the type of model, w.r.t. the number of parameters, and Y is the patch size. For example, DHVT-T/4 means our tiny model with patch size of 4. The throughput are measure as follows: The batch size is set to 256 except for DHVT-S/2, and DHVT-S/2 uses batch size of 128. The devices is one NVIDIA GeForce RTX 3090 GPU, and the two CPUs are Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz.

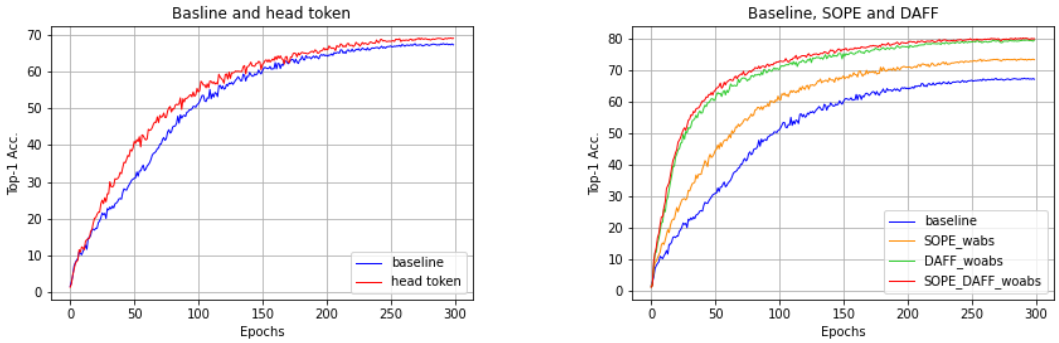
We can see that our DHVT-T/4 model achieves 80.93 accuracy with only 0.4 GFLOPs computation and is able to process more than 6000 images per second. Such light weight but accurate model has the potential to be used in mobile style. As the patch size drops from 4 to 2, the number of patch tokens grows quadratically, and the corresponding GFLOPs and throught puts also changes

Table 2: The computational complexity of our model on CIFAR-100 dataset. We use DHVT-X/Y to denote the variants of models. Here X is the type of model, w.r.t. the number of parameters, and Y is the patch size.

Method	#Params	GFLOPs	Throughput (img/sec)	Resolution	Acc (%)
DHVT-T/4	6.0M	0.4	6684	32	80.93
DHVT-S/4	23.4M	1.5	3631	32	82.91
DHVT-T/2	5.8M	1.7	1844	32	83.54
DHVT-S/2	22.8M	6.3	812	32	85.68

quadratically. Though our DHVT-S/2 reaches significant high accuracy, it is at the huge cost of computation. We hope to alleviate the burden and maintain accuracy in the future investigation.

2.3 Training Efficiency on CIFAR-100



(a) (b)

Figure 1: (a) Training Efficiency of applying head token only. (b) Training efficiency of applying SOPE and DAFF and with or without absolute positional embedding.

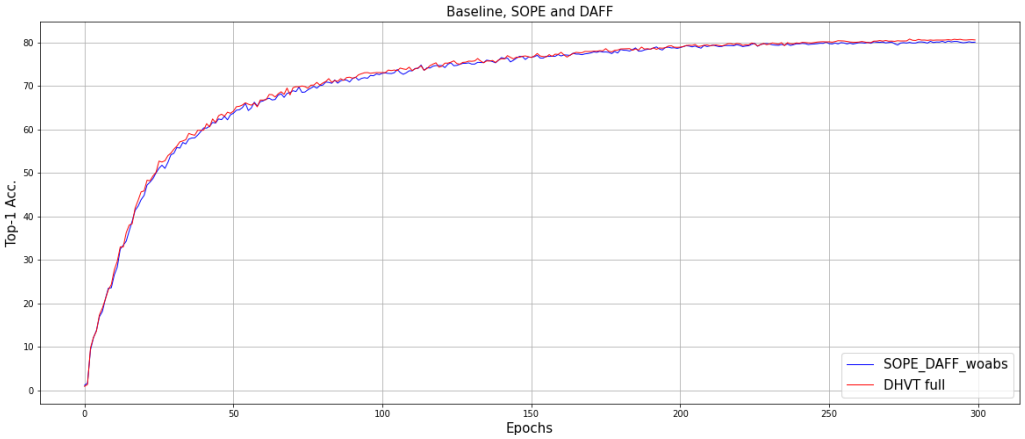


Figure 2: Training efficiency of head token when both SOPE and DAFF are applied and absolute positional embedding is removed.

In this section, we show the training efficiency of our method when applying each module. All the experimental settings are the same as in the ablation study section. The baseline module is DeiT-Tiny with 4 heads. Here we denote "without absolute positional embedding" as "woabs" and "wabs" denotes the opposite. From Fig. 1 (a), we can see that introducing head token into baseline can facilitate the overall training process and reaches higher accuracy, proving the effectiveness of our

novel head token design. And from Fig. 1 (b), we can see that both SOPE and DAFF is able to improve the whole training, and their combination has a positive influence on the model.

In addition, in Fig. 2, we use "DHVT full" to represent the full version of our model, including SOPE, DAFF and head tokens, while removing absolute positional embedding. In such circumstance, head token is still able to give rise to the performance during most of the training epochs, and the final result is also a little higher than the one without head token.

2.4 Visualization on CIFAR-100

To further understand the feature interaction style in our proposed model, we provide more visualization results in this section. First, we visualize the attention map of all the tokens, including class token, patch tokens and head tokens on the 2nd, 5th, 8th and 11th encoder layers.We provide three example input images in total. Second, we visualize the attention of head tokens to patch tokens in their corresponding head on the 2nd, 5th, 8th and 11th encoder layers. The model we visualize here is DHVT-T training from scratch on CIFAR-100 dataset, which contains 4 attention head in each layer thus the corresponding number of head token is 4. Patch size is set to 4 here.

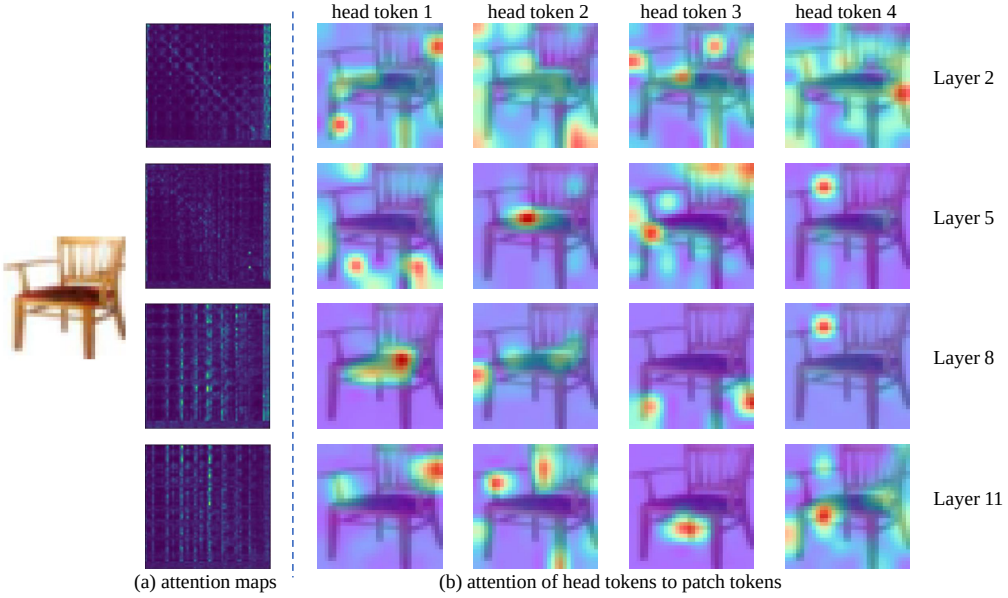


Figure 3: (a) Averaged attention maps. (b) Attention of head tokens to patch tokens in the corresponding heads.

From the above results in Figure 3,4,5, we can summarize two attributes that head tokens brought to the model. First, in the lower layers, such as the 2nd layer, model tends to attend neighbouring features and interacts with head tokens. As going deeper, such as in the 5th layer, attention is scattered around all patch tokens and head tokens do not receive much attention here. In higher layers, like in the 8th layer, attention focus on some of the patch tokens and now head tokens receive more attention than in the 5th layer. Finally, in the layers near output layer, such as the 11th layer, patch tokens do not focus too much on head tokens, and all the tokens converge their attention to the prominent patch tokens.

Second, each head token represents different representation as we visualized above. When head tokens participate in attention calculation, they help interaction of different representation, fusing poor representation encoded in different heads into a strong integral representation.

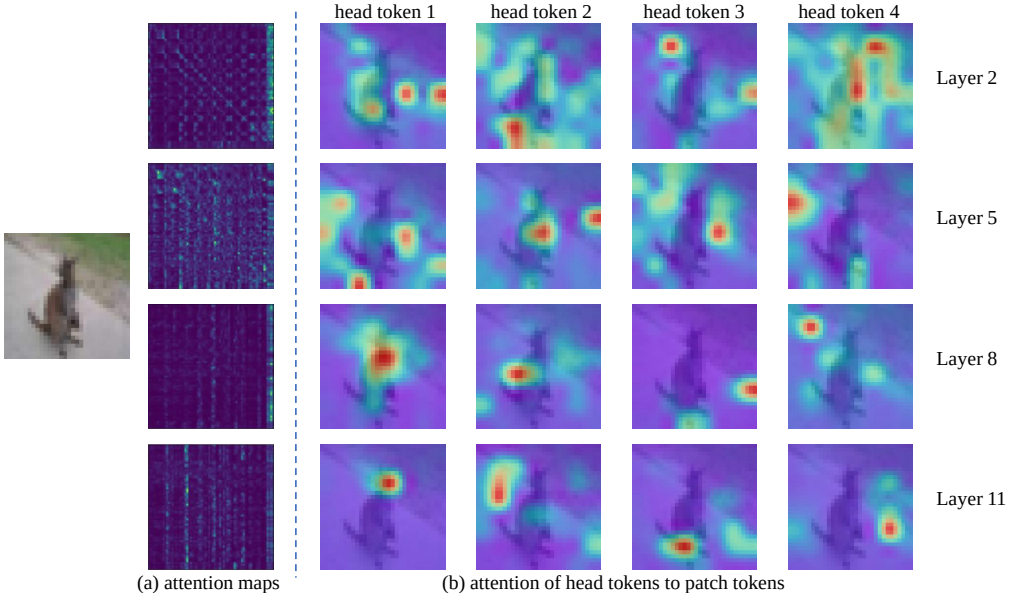


Figure 4: (a) Averaged attention maps. (b) Attention of head tokens to patch tokens in the corresponding heads.

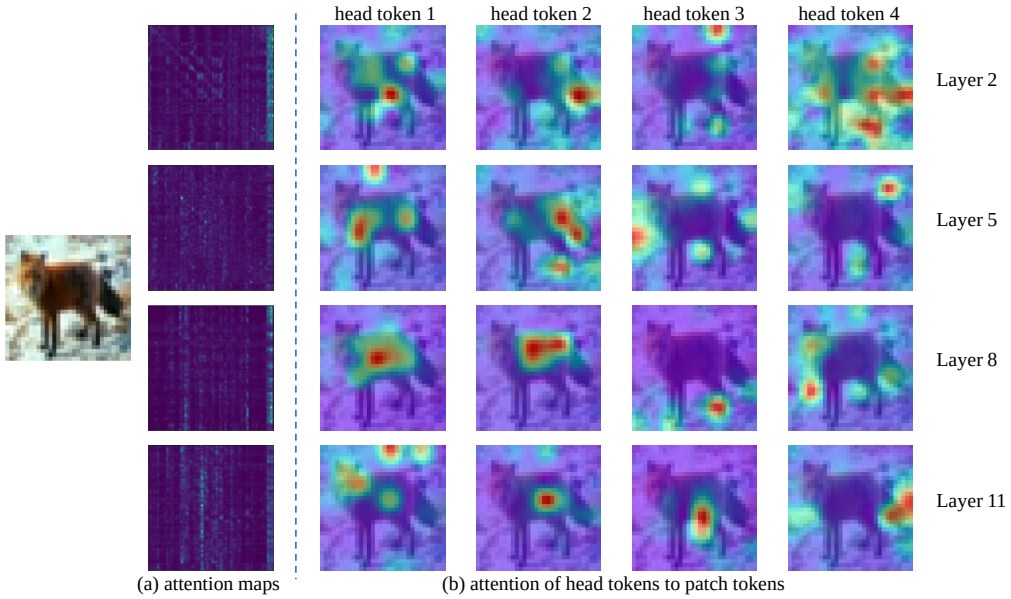


Figure 5: (a) Averaged attention maps. (b) Attention of head tokens to patch tokens in the corresponding heads.

3 DomainNet Dataset

3.1 Example of DomainNet

In this part, we visualize some example images in the DomainNet datasets as in Fig. 6. These datasets has a domain shift from traditional natural image dataset like ImageNet-1K and CIFAR. Also because of the scarce training data, models are hard to train from scratch on such datasets. However, our proposed DHVT is able to address the issue with satisfactory results in both train-from-scratch and pretrain-finetune scenario. Under comparable amount of computational complexity, our models exhibit non-trivial performance gain compared with baseline models on all of the three datasets.

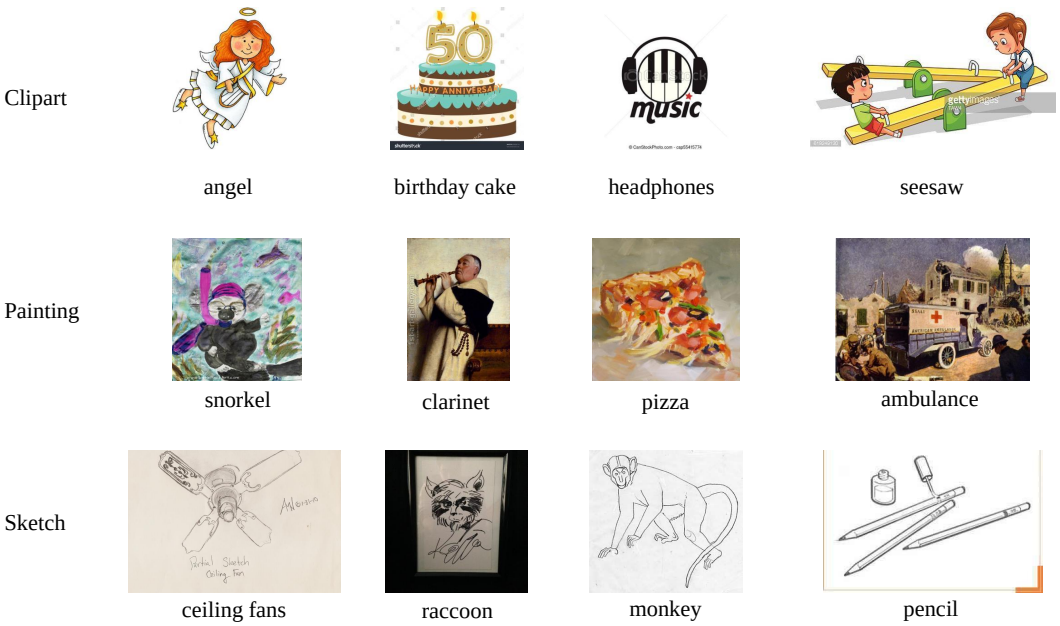


Figure 6: Visualization of example in DomainNet datasets Clipart, Painting and Sketch

3.2 Fine-tuning on DomainNet

We analyse the fine-tuning results on DomainNet datasets in this section. All the models are pre-trained on ImageNet-1K only and then fine-tuned on Clipart, Painting, Sketch. Results are shown in Table 3. We cite the reported results from corresponding papers. Note that the fine-tuning epochs in baseline models are 100, the same as we use. When fine-tuning our DHVT, we use AdamW optimizer with cosine learning rate scheduler and 2 warm-up epochs. We use a batch size of 256, initial learning rate of 0.0005, weight decay of 1e-8, and fine-tuning epochs of 100. We fine-tune our model on image size of 224×224 and we use patch size of 16, head numbers of 3 and 6 for DHVT-T and DHVT-S respectively, the same as the model pre-trained model on ImageNet-1K.

Table 3: Pretrained on ImageNet-1K and then fine-tuned on the DomainNet (top-1 accuracy (%), 100 fine-tuning epochs). The ImageNet-1K column shows the accuracy of pretrained model on ImageNet-1K.

Method	GFLOPs	ImageNet-1K	Clipart	Painting	Sketch
ResNet-50 [1]	3.8	-	75.22	66.58	67.77
T2T-ViT-14 [1]	5.2	81.5	74.59	72.29	72.18
Swin-T [1]	4.5	81.3	73.51	72.99	72.37
DHVT-T (Ours)	1.4	76.5	77.88	72.05	70.79
DHVT-S (Ours)	5.1	82.3	80.06	74.18	73.32

From Table 3, we can see that our models show better performance than baseline methods ResNet-50 [7], Swin Transformer[5], T2T-ViT [6]. Especially on Clipart, our DHVT-S reaches more than 80 accuracy, showing a significant performance gap compared with baseline methods. Our tiny model achieves comparable and even better accuracy than T2T-ViT and Swin Transformer with much lower computational complexity on Clipart and Painting. From the main part of this paper, the performance of training from scratch of DHVT-S is 68.72, which outperforms the fine-tuning result of ResNet-50, exhibiting the train-from-scratch capacity of our method.

4 ImageNet-1K Dataset

4.1 Comparision on ImageNet-1K

Table 4: Performance comparison of different method on ImageNet-1K. All models are trained from random initialization.

Method	#Params	Image Size	GFLOPs	Top-1 Acc (%)
RegNetY-800MF [8]	6.3M	224	0.8	76.3
RegNetY-4.0GF [8]	20.6M	224	4.0	79.4
ConvNeXt-T [9]	29M	224	4.5	82.1
T2T-ViT-7 [6]	4.3M	224	1.2	71.7
DeiT-T [3]	5.7M	224	1.3	72.2
PiT-Ti [10]	4.9M	224	0.7	72.9
ConViT-Ti [11]	5.7M	224	1.4	73.1
CrossViT-Ti [12]	6.9M	224	1.6	73.4
TNT-T [13]	6.2M	224	1.4	73.6
LocalViT-T [14]	5.9M	224	1.3	74.8
ViTAE-T [15]	4.8M	224	1.5	75.3
CeiT-T [4]	6.4M	224	1.2	76.4
DHVT-T (Ours)	6.2M	224	1.4	**76.5**
DeiT-S [3]	22.1M	224	4.6	79.8
PVT-S [16]	24.5M	224	3.8	79.8
PiT-S [10]	23.5M	224	2.9	80.9
CrossViT-S [12]	26.7M	224	5.6	81.0
PVT-Medium [16]	44.2M	224	6.7	81.2
Conformer-Ti [17]	23.5M	224	5.2	81.3
Swin-T [5]	29.0M	224	4.5	81.3
ConViT-S [11]	27.8M	224	5.4	81.3
TNT-S [13]	23.8M	224	5.2	81.3
T2T-ViT-14 [6]	21.5M	224	5.2	81.5
NesT-T [18]	17.0M	224	5.8	81.5
CvT-13 [19]	20.0M	224	4.5	81.6
Twins-SVT-S [20]	24.0M	224	2.8	81.7
CaiT-XS24 [21]	26.6M	224	5.4	81.8
CoaT-Lite Small [22]	20.0M	224	4.0	81.9
CeiT-S[4]	24.2M	224	4.5	82.0
ViL-S[23]	24.6M	224	4.9	82.0
PVTv2-B2[24]	25.4M	224	4.0	82.0
ViTAE-S[15]	23.6M	224	5.6	82.0
LG-T[25]	32.6M	224	4.8	82.1
Focal-T[26]	29.1M	224	4.9	82.2
DHVT-S (Ours)	24.1M	224	5.1	**82.3**

We conduct experiments on ImageNet-1K dataset to test the performance of our proposed Dynamic Hybrid Vision Transformer (DHVT) on common medium dataset. From the above Table 4, we can see that with less parameter and comparable computational complexity, our DHVT achieve state-of-the art results compared to recent CNNs and ViTs. Both of our model in trained from scratch on ImageNet-1K datasets, with image size of 224×224, patch size of 16, optimizer of AdamW and

base learning rate of 0.0005 following cosine learning rate decay, weight decay of 0.05, warm-up epoch of 10, batch size of 512. All the data-augmentations and regularization methods follow Deit [3], including random cropping, random flipping, label-smoothing [27], Mixup [28], CutMix [29] and random erasing [30].

Our DHVT-T reaches 76.5 accuracy with only 6.2M parameters, while DHVT-S achieves 82.3 accuracy with only 24.1M parameters. Our model not only outperforms the best non-hierarchical vision transformer CeiT [4], but also shows competitive performance to most of the hierarchical vision transformers like Swin Transformer [5] and hybrid architecture like ViTAE-S [15]. We also show better performance than recent strong CNNs RegNet [8]and ConvNeXt [9]. We achieve such results with much less parameters than existing methods, while our computational complexity is also higher than them. This is a kind of mixed blessing. On one hand, our method can be seen as using less parameter to conduct comprehensive and sufficient computation. On the other hand, such amount of computation is a huge burden for both training and testing. We hope to reduce the computational burden in the future research, while maintaining the same performance.

4.2 Visualization on ImageNet-1K

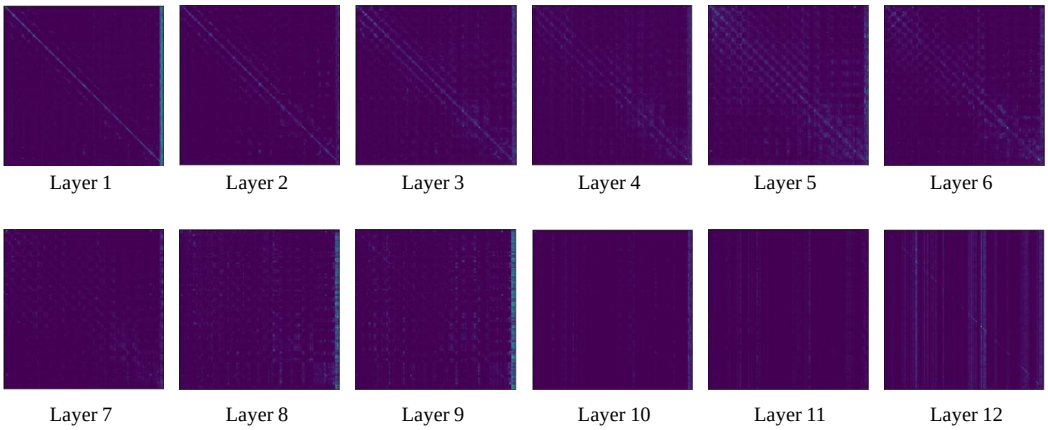


Figure 7: Averaged attention maps from DHVT-S training from scratch on ImageNet-1K.

We further visualize the attention maps of our proposed model training from scratch on ImageNet-1K only. Here the model is DHVT-S with 6 attention head, patch size of 16. Note that head tokens are concatenated behind patch tokens, so the right hand side of attention maps represents the attention from all the tokens to head tokens. Under the introduction of head tokens, we are able to understand the feature extraction and representation process in our model.

In the input encoder layer, i.e. the 1st layer, all the tokens focus on themselves and head tokens. And in the early stage, such as in the 2nd to 6th layers, all the tokens focus more on themselves and do not attend too much on head tokens. Further in middle stage, such as in 7th to 9th layers, head tokens draw more attention from other tokens. And in late stage, such as in 10th, 11th and 12th layers, attention are more on prominent patch tokens.

From such attention style, we can conclude the feature extraction and representation process as: Early stage focus on local and neighbouring features, extracting low-level fine-grained feature. Then feature representations interact and fuse with each other to generate strong enough representation. The representation in each token is enhanced by such interaction. And in the late stage, the model focus on the most prominent patch tokens to extract information for final classification.

In the future research, it maybe possible to only apply head token design in the middle stage of vision transformers to save computation cost. We hope this visualization will inspire more wonderful architecture in the future.

5 Architecture Variants

Table 5: Architecture variants of DHVT

Method	Dataset	Patch	DAFF		#heads	depth	Dim	Params	GFLOPs
			MLP	S&E					
DHVT-T	CIFAR	4	4	4	4	12	192	6.0M	0.4
DHVT-T	CIFAR	2	4	4	4	12	192	5.8M	1.7
DHVT-S	CIFAR	4	4	4	8	12	384	23.4M	1.5
DHVT-S	CIFAR	2	4	4	8	12	384	22.8M	6.3
DHVT-T	Domain	16	4	4	4	12	192	6.1M	1.4
DHVT-S	Domain	16	4	4	6	12	384	23.8M	5.1
DHVT-T	ImageNet	16	4	4	3	12	192	6.2M	1.4
DHVT-S	ImageNet	16	4	4	6	12	384	24.1M	5.1

We present the variants and architecture parameter of our proposed model in this section. Note that all the models remove the absolute positional embedding. For CIFAR-100 dataset, the image size is 32×32, and for DomainNet and ImageNet it is 224×224. In Table 5, "MLP" represents MLP projection ratio and "S&E" is the reduction ratio in the squeeze-excitation operation. Code will be released if accepted.

References

[1] Liu, Y., E. Sangineto, W. Bi, et al. Efficient training of visual transformers with small datasets. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] Dosovitskiy, A., L. Beyer, A. Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] Touvron, H., M. Cord, M. Douze, et al. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[4] Yuan, K., S. Guo, Z. Liu, et al. Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 579–588. 2021.

[5] Liu, Z., Y. Lin, Y. Cao, et al. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022. 2021.

[6] Yuan, L., Y. Chen, T. Wang, et al. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567. 2021.

[7] He, K., X. Zhang, S. Ren, et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 2016.

[8] Radosavovic, I., R. P. Kosaraju, R. Girshick, et al. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436. 2020.

[9] Liu, Z., H. Mao, C.-Y. Wu, et al. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[10] Heo, B., S. Yun, D. Han, et al. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11936–11945. 2021.

[11] d'Ascoli, S., H. Touvron, M. L. Leavitt, et al. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[12] Chen, C.-F. R., Q. Fan, R. Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 357–366. 2021.

[13] Han, K., A. Xiao, E. Wu, et al. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] Li, Y., K. Zhang, J. Cao, et al. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[15] Xu, Y., Q. Zhang, J. Zhang, et al. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Wang, W., E. Xie, X. Li, et al. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578. 2021.

[17] Peng, Z., W. Huang, S. Gu, et al. Conformer: Local features coupling global representations for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 367–376. 2021.

[18] Zhang, Z., H. Zhang, L. Zhao, et al. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *AAAI Conference on Artificial Intelligence (AAAI), 2022*. 2022.

[19] Wu, H., B. Xiao, N. Codella, et al. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31. 2021.

[20] Chu, X., Z. Tian, Y. Wang, et al. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[21] Touvron, H., M. Cord, A. Sablayrolles, et al. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42. 2021.

[22] Xu, W., Y. Xu, T. Chang, et al. Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9981–9990. 2021.

[23] Zhang, P., X. Dai, J. Yang, et al. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3008. 2021.

[24] Wang, W., E. Xie, X. Li, et al. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, pages 1–10, 2022.

[25] Li, J., Y. Yan, S. Liao, et al. Local-to-global self-attention in vision transformers. *arXiv preprint arXiv:2107.04735*, 2021.

[26] Yang, J., C. Li, P. Zhang, et al. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021.

[27] Szegedy, C., V. Vanhoucke, S. Ioffe, et al. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826. 2016.

[28] Zhang, H., M. Cisse, Y. N. Dauphin, et al. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*. 2018.

[29] Yun, S., D. Han, S. J. Oh, et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032. 2019.

[30] Zhong, Z., L. Zheng, G. Kang, et al. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pages 13001–13008. 2020.