SUPPLEMENTARY MATERIAL FOR: CHOPPING FORMERS IS WHAT YOU NEED IN VISION.

Anonymous authors
Paper under double-blind review

1 EINSTEIN NOTATION FOR NEURAL NETWORKS

We defined a *neural network layer* as a function f that takes as input a tensor X_{mc} composed of $m \in [1, M]$ spatial positions (or tokens) with $c \in [1, C]$ features (or channels) and produces as output a tensor Y_{nd} composed of $n \in [1, N]$ tokens with $d \in [1, D]$ channels:

$$Y_{nd} = f(X_{mc}).$$

We then defined the fully-connected layer (FC) as the most general instantiation of a *linear* neural network layer, parametrized by a 4-dimensional weight tensor W_{mncd} such that:

$$Y_{nd} = X_{\underline{mc}} W_{\underline{mn}\underline{c}d}.$$

And we defined the dynamic fully-connected layer (DFC) as a non-linear generalization of the FC layer where the weight tensor is computed dynamically as a function g of the input: $W_{mncd} = g(X_{mc})$, with the instance dimension indexed by $i \in [1, I]$:

$$Y_{ind} = X_{i\underline{mc}} W_{i\underline{mn}\underline{c}d} \qquad \text{with} \quad W_{imncd} = [g(X^1_{mc}), \ldots, g(X^I_{mc})].$$

We show below using the Einsum notation that a number of well-known neural network layers are special cases of the FC or DFC layers.

1.1 LINEAR LAYERS

The Convolutional layer (Conv) significantly reduces the complexity of the FC layer by making use of a *local receptive field* with *shared weights*. More specifically, the convolutional layer is obtained from the FC layer by applying the following three steps, where steps 1) and 2) together correspond to the process of "patch extraction".

1) explicitly broadcast the input tensor along the n dimension: $Y_{nd} = X_{\underline{mn}\underline{c}} W_{\underline{mn}\underline{c}d}$
2) shrink the receptive field to $k \in [1, K]$ local tokens: $Y_{nd} = X_{\underline{kn}\underline{c}} W_{\underline{kn}\underline{c}d}$
3) share the weights spatially: $Y_{nd} = X_{\underline{kn}\underline{c}} W_{\underline{kc}d}$

The Depthwise Convolutional layer (DW-Conv) reduces complexity further by dropping the channel mixing part, focusing on spatial mixing only: $Y_{nc} = X_{\underline{kn}c} W_{\underline{k}c}$.

The Pointwise Convolutional layer (Point-Conv) is orthogonal to the DW-Conv layer: it drops the spatial mixing part and focuses on the channel mixing part only: $Y_{md} = X_{m\underline{c}} W_{\underline{c}d}$.

The Average Pooling layer (Avg-Pool) can simply be seen as a special case of the DW-Conv layer where the weights are constant: $Y_{nc} = X_{\underline{k}nc} P_{\underline{k}c}$ with $p_{kc} = 1/K$.

1.2 DYNAMIC LAYERS

The Self-Attention layer (SA) used in transformer architectures (Vaswani et al., 2017) reduces the complexity of the DFC layer by turning off channel mixing while using multiple heads.

1) turn off the channel mixing part: $Y_{inc} = X_{i\underline{mc}} W_{i\underline{mn}}$
2) split the processing into $h \in [1, H]$ heads with $c \in [1, C/H]$: $Y_{inch} = X_{i\underline{mch}} W_{i\underline{mn}h}$

Ignoring the query, key, value embeddings for simplicity, the self-attention weight construction mechanism is a dot product of X with itself:

$$Y_{inch} = X_{i\underline{mch}} W_{i\underline{mn}h} \qquad \text{with} \quad W_{imnh} = softmax(X_{im\underline{ch}} X_{in\underline{ch}}).$$

The Dynamic Convolutional layer (Dyn-Conv) is a Conv layer where the kernels are computed dynamically (Jia et al., 2016; Mildenhall et al., 2018). It is often used 1) without the shared weights constraint, to allow the weights to adapt spatially to their inputs, and 2) as a depthwise variant, to keep the complexity manageable. In general, the weight construction mechanism g_{cnn} simply consists of a stack of a few standard convolutional layers:

$$Y_{inc} = X_{i\underline{k}nc}\, W_{i\underline{k}nc} \qquad \text{with} \quad W_{iknc} = g_{cnn}(X_{iknc}).$$

The Deformable Convolutional layer (Def-Conv) is a variant of the Dyn-Conv where not only the weights but also the locations where they are applied, are dynamically computed (Dai et al., 2017; Zhu et al., 2019). The patch extraction process can be understood as a multiplication with a dynamic mask M implementing an interpolation with offsets, followed by a dynamic convolution:

$$Y_{inc} = (X_{i\underline{m}nc}\, M_{i\underline{m}kc})_{ik\underline{k}nc}\, W_{i\underline{k}nc} \qquad \text{with} \quad M_{imkc}, W_{iknc} = g_{cnn}(X_{iknc}).$$

Squeeze and Excitation Convolutional layer (S&E) is a variant of the Dyn-Conv where a standard convolution is followed by a dynamic channel modulation M (Hu et al., 2018). The construction mechanism g_{cnn} usually starts with a global average pooling generating channel-wise statistics:

$$Y_{ind} = (X_{i\underline{k}n\underline{c}}\, W_{\underline{kc}d})_{ind}\, M_{id} \qquad \text{with} \quad M_{id} = g_{cnn}(X_{iknc}).$$

The Max Pooling layer (Max-Pool) is similar to the Avg-Pool layer, but the weights are not constant anymore and depend on the input:

$$Y_{inc} = X_{i\underline{k}nc} P_{i\underline{k}c} \qquad \text{with} \quad p_{ikc} = \begin{cases} 1 & \text{if } \underset{l\in[1,K]}{\arg\max}(x_{ilc}) = k\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

2 Implementation Details

Here we report implementation details for the experiments in the main paper. We use the same setup for all the evaluated methods and do not ablate training setup in our experiments. We leave the exploration of the ideal training setup to future work.

Puzzle Reconstruction
We used a two-layer encoder composed of two convolutions with kernel size 5x5 followed by ReLU non-linearities. We used a two-layer decoder consisting of two transposed convolutions. After the decoder, one final residual convolution is used to map the features directly into the predicted output image. We used the MNIST dataset and divided each input image into pieces of size 7x7. We train all variants end-to-end for 500 epochs with a batch size of 128 on one GPU, minimizing the mean square error and using Adam optimizer with a learning rate of $1e^{-3}$.

Classification
We follow the experimental setup of Yu et al. (2022) for training and testing. We used the Imagenet 1K datasets as input images of size 224x224. We train each model for 300 epochs using AdamW optimizer cosine schedule and learning rate of $1e^{-3}$. We used the standard data augmentations strategies of CutMix, MixUP, CutOut, and RandAugment. For all our experiments, we used 8 GPUs and an effective batch size of 1024. We refer to the original paper for more in-depth details.

Detection and Segmentation
For training and testing, we follow the experimental setup of Yu et al. (2022). We use the models trained on Imagenet 1K as backbones for the detection and segmentation task. We use the COCO dataset, specifically we train on the train2017 and test on the 5K validation images of val2017. We report performance for the best epoch. We use the Mask R-CNN models and trained with AdamW and a learning rate of $1e^{-4}$ on 8 GPUS and 12 epochs (1x training schedule). We refer to the original paper for more in-depth details.

3 ABLATIONS

3.1 SIZE

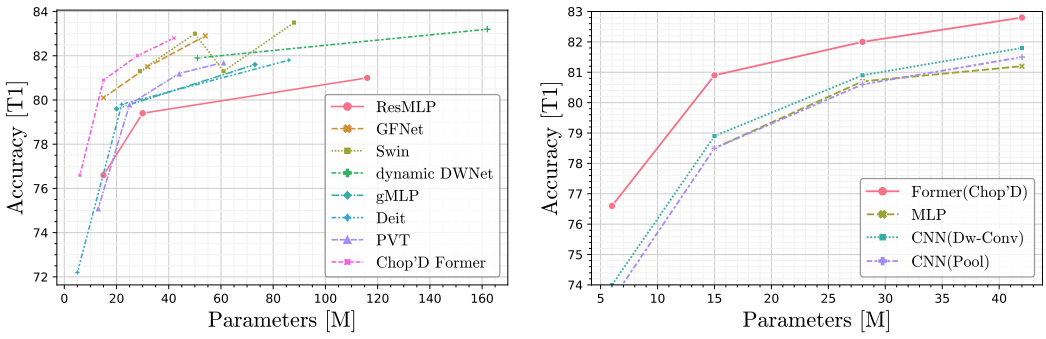
We scale the Chop'D-Former architecture and design networks with various sizes, namely Nano N, Tiny T, Small S, and Medium M. In all cases we use the same backbone as explained in the main paper. In Table 1 we summarize our design choices by specifying for each of the four stages the number of the R channels, the number of spatial positions, and the number of blocks used. In our implementation, we assume C=D and R=C/4 at all depth. Obviously, increasing parameters and Flops results in better accuracy, however, even for very small sizes, Chop'D Former performance remains competitive.

In Figure 1 we continue our analysis by providing an overview of the trade-off between performance and size from two different perspectives. Figure 1 (a) compares Chop'D Former against methods built as a stack of dynamic layers like ours, but trained using different setups and macro-design (Rao et al., 2021; Touvron et al., 2022; Liu et al., 2021b; Touvron et al., 2021; Han et al., 2021; Wang et al., 2021; Liu et al., 2021a). This figure shows that Chop'D Former design can be used as a solid backbone for CV tasks. As apparent from the plot, Chop'D Former scales well across various sizes, and its performance is even comparable to models having larger parameter counts. For example, Chop'D Former performance is comparable with a large dynamic DWNet (83.2 vs 82.8) which uses almost 4 times its amount of parameters (162 vs 42). When compared with the strong Swin-Transformer baseline, it reaches a performance of 82.0 points for 28 million parameters, stacking well against the 81.3 T1 of a Swin model of similar complexity (29 million).

Figure 1 (b) completes our analysis by comparing different classes of architecture under controlled training setup and network design. Using this figure, we provide evidence of how DFC layers represent, by design, a stronger alternative to the rest of neural networks layers.

Architecture	Spatial Positions	Channels (R)	Depth	Param M	FLOPS G	T1	T5
Chop'D Former - N	[56, 28, 14, 7]	[32, 63, 160, 256]	[2, 2, 6, 2]	6	1.0	76.6	93.4
Chop'D Former - T	[56, 28, 14, 7]	[64, 128, 320, 512]	[2, 2, 6, 2]	15	2.4	80.9	95.4
Chop'D Former - S	[56, 28, 14, 7]	[64, 128, 320, 512]	[4, 4, 12, 4]	28	4.5	82.0	95.6
Chop'D Former - M	[56, 28, 14, 7]	[64, 128, 320, 512]	[6, 6, 18, 6]	42	6.8	82.8	95.9

Table 1: **Ablation on Architecture sizes** Chop'D-Former is a hierarchical architecture with four stages. We ablate design choices and report performance together with size.



(a) Architectures deploying dynamic layers

(b) Architectures classes under fair comparison

Figure 1: **ImageNet-1K validation accuracy vs. Model Size.** (a) Comparison of different models using dynamic layers (i.e. DFC variants) independently from their architecture design. (b) Comparison of different classes of architectures under the same macro design choices and training setup. Chop'D Former stacks favorably against similar methods using dynamic layers, and maintains a stable gap when compared with other classes of architectures in a controlled training setup.

In the figure, the performances of Chop'D Former models of various sizes are plotted next to comparable CP decomposed CNN and MLP networks. As one can clearly see, Chop'D Former keeps a steady advantage against these two families of networks. For example, the DFC layers used in Chop'D Former allow for a dynamic and global response, outperforming by a large margin a fully connected network (MLP) across the different compared sizes ($+1.6, +1.3$ for models of sizes $42, 28$ M of parameters). Against a Convolutional Network, the performance gap is similar ($+1, +1.1$ on $42, 28$ M), thus confirming the clear advantage in using a dynamic, spatially adaptive, and global reasoning component when building neural network architectures.

3.2 ISOTROPIC ARCHITECTURE

In the main paper, we focus on computer vision networks with a hierarchical structure (that is with a multi-scale representation of the input obtained with subsequent downsampling stages). In this ablation, we examine if our DFC block design can generalize to isotropic architectures, where no downsampling layers are used across the model. In our experiments, features are kept at the same 14 $\times$ 14 resolutions throughout the entire network and, similarly, the dimensionality of the layers is kept the same at all depths. We construct two isotropic models using 192 and 384 feature dimensions. Depths are set at 12 to match the number of layers in the hierarchical version of our model. To provide evidence that DFC layers are beneficial independently from the hierarchical or isotropic variant of the network, we compare these two types of architectures side-by-side by progressively silencing various characteristics of the DFC layer. To ease comparison with the main paper, we replicate the same training setup and arrange results in Table 2 accordingly. Results for ImageNet-1K show how DFC layers are able to outperform FC layers (non-dynamic) and Convolutional layers (non-dynamic, local) by a large margin, replicating a trend observed in the hierarchical case. For example, replacing an FC layer with its non-linear dynamic extension (i.e. DFC) for a network of 20 M parameters brings more than $+2$ points boost in performance, even higher than the $+1.6$ registered for the hierachical models of 15 M parameters. Similarly it outperforms convolution by a large margin ($+2.5$, $+2.7$), providing further evidence that DFC design is competitive when used in non-hierarchical models.

Layer		Architecture	Complexity		Classification			
Type	Weights		P(M)	F(G)	T1	T5	v2	Real
DFC-CP	$\approx \mathbf{W_{imncd}}$	Former (Chop'D)	5	1.0	**71.9**	**90.9**	**59.7**	**79.9**
			20	3.8	**79.7**	**95.0**	**68.8**	**86.0**
FC -CP	$\approx \mathbf{W_{mncd}}$	MLP	5	1.0	68.8	89.2	56.6	76.9
			20	3.8	77.2	93.5	64.8	83.6
Conv-CP	$\approx \mathbf{W_{kcd}}$	CNN (Dw-Conv)	5	1.0	69.4	89.5	56.7	77.5
			20	3.8	77.0	93.3	65.4	83.7
Conv-CP	$\approx \mathbf{P_{kcd}}$	CNN (Pool)	5	1.0	63.5	85.6	50.6	71.3
			20	3.8	73.0	91.1	59.3	80.1

Table 2: **Isotropic Architecture.** Comparisons among CP decomposition for different Class of Architectures on Large scale classification on Imagenet. All the architecture share the same macro-design and training procedure, processing input of size 14x14 with no extra pooling stages. Similarly to what observed on a hierarchical architecture, The Chop'D Former approximates via CP decomposition DFC layers reaching best performance.

3.3 RANK R

We recall that our Chop'D Former uses CP-decomposed DFC layers, which act as approximations for the non-linear extension of FC layers. In our implementation, the DFC weight tensor W_{inmcd} has the same number of input and output channels $C = D$, and the rank of the CP decomposition (R) is assumed to be a quarter of the original dimensionality. However, a natural question is to assess the change in the performance of our network when a different fraction of the original channel is used as R.

In this ablation, we follow the same setup used for classification experiments in the main paper, and train the same Chop'D Former network allowing a higher or lower number of channels in the latent space. We used an Isotropic version of Chop'D Former since in this case, the number of channels

is the same for every layer of the network. Table 3 illustrates the performance of the same model with different R and the same $C = D$. It is evident that, as R decreases, the performance degrades. In fact, in these cases, the decomposition is not able to approximate properly the weight tensor, and thus converges to a suboptimal solution.

Architecture	C,D	R	T1	T5
Iso Chop'D Former	768	384	77.7	93.8
Iso Chop'D Former	768	192	71.9	90.9
Iso Chop'D Former	768	96	62.6	83.7
Iso Chop'D Former	768	48	47.5	73.5

Table 3: **Ablation on the CP decomposition rank** R for a Chop'D Former Isotropic architecture. In an isotropic architecture, all the layer have the same values for C,D and R. Decreasing the rank R results in higher approximation error and lower performance.

3.4 Spatial Reasoning Module

We provide two extra ablations on the Chop'D Former architecture. For all our experiments, we use Chop'D Former of size Tiny (Chop'D Former-T), which approximately corresponds to 2.4 GFlops, trained for the classification task on ImageNet. We left the training procedure unchanged and investigate the different number of groups for the dynamic pooling module and the different number of spatial positions for the gating module. Results are shown in Table 4 and 5, respectively. As seen from Table 4, increasing the number of learnable pooling modules steadily increases performance until it saturates at a value of 12. Therefore we use this value as the default setting in all our experiments. Then, referring to Table 5, we observe that leveraging all available information results in the best overall performance. Interestingly, Chop'D-Former can perform reasonably well even when ignoring 60% of all the available pixels in its spatial adaptive module (Gating positions N*40%).

Architecture	SAT groups	T1	T5
Chop'D Former-T	1	80.6	95.2
Chop'D Former-T	4	80.6	95.3
Chop'D Former-T	8	80.7	95.2
Chop'D Former-T	12	80.9	95.4
Chop'D Former-T	24	80.5	95.2

Table 4: **Ablations on number of groups G for Chop'D Former-Tiny**. A group composed of 12 pooling layer achieves best performance.

Architecture	Gating positions	T1	T5
Chop'D Former-T	N*100%	80.9	95.4
Chop'D Former-T	N*80%	80.6	95.2
Chop'D Former-T	N*40%	80.4	95.1

Table 5: **Ablations on number of spatial positions used to estimate gating module.** Even using 40% of all available tokens yields good performance.

4 Additional Experiments

In the main paper, we fix the choices for the architecture and training procedure to isolate the contribution of spatial reasoning in computer vision tasks. In that context, we have shown strong evidence that DFC layers are capable to outperform alternative spatial-reasoning layers thanks to the joint use of a dynamic and spatially adaptive response to the input paired with an adaptive and global receptive field.

In this section, we extend our findings with comparisons against well-established backbones for computer vision for ImageNet classification, and we showcase that, even without any specialized macro design choices nor adjustments of the training recipe, Chop'D-Former is capable of reaching competitive performance against methods highly tuned towards maximizing performance in this specific classification task.

4.1 IMAGE CLASSIFICATION

In Table 6 we report the performance of Chop'D Former of various sizes and complexities compared to other established computer-vision architectures on ImageNet. For each of the different methods, we highlight architectural characteristics and report efficiency metrics as provided by the authors, namely parameter and Flops count (both measures are independent of hardware and software choices and thus provide fair grounds for comparison). In terms of performance, we report results on the original ImageNet-1K validation (T1) as well as two additional test-set as measure of overfitting: the cleaned-up ReaL validation set (Beyer et al., 2020) and ImageNet-V2 (Recht et al., 2019). As visible from the table, Chop'D Former outperforms the majority of the considered architectures, thus demonstrating strong evidence that dynamic and spatially adaptive reasoning is a crucial component in architecture design, even in the case of a very quickly developing landscape of computer vision backbones. In particular, our Chop'D Former-S is capable to provide a $+0.6$ T1 in performance with a -0.5G Flops decrease in complexity against Poolformer-S36 (Yu et al., 2022), which, we wish to stress, shares the same training procedure and has a similar amount of parameters. Moreover, when compared against methods that use dynamic layers (i.e. DFC, and DCNN) of similar Flop count, our Chop'D Former-S is capable to increase performance by $+2.2$ T1 over Deit-S (Touvron et al., 2021), $+2.4$ T1 over gMLP-S (Liu et al., 2021a) and $+0.7$ T1 over Swin-S (Liu et al., 2021b). The only networks that exhibit performance either use a very large number of parameters, are extensively finetuned in terms of architectural choices and training strategies, or are based on hybrid models which explore joint use of different token-mixers modules in the same architecture. We wish to highlight that all of these are contributions orthogonal to our method. As such, we leave the investigation of stronger training procedures and enhanced spatial-reasoning strategies to future work as such investigations geared towards maximizing performance on this classification task are not in the scope of the present work. Here, we have proven that it is possible to achieve spatially adaptive dynamic spatial reasoning with complexity comparable to a traditional convolution.

Architecture					Classification				
Name	Class	Token-Mixer	Adaptivity (i, n)	Receptive Field	Params (M)	Flops (G)	T1	v2	Real
EfficientNetV2-S (Tan & Le, 2021)	Hybrid	Conv/MBConv/F-MBConv	i, n	3x3	22	8.8	83.9	-	-
EfficientNetV2-M (Tan & Le, 2021)	Hybrid	Conv/MBConv/F-MBConv	i, n	3x3	54	24.0	85.1	-	-
EfficientNetV2-L (Tan & Le, 2021)	Hybrid	Conv/MBConv/F-MBConv	i, n	3x3	120	53.0	85.7	-	-
CoAtNet-0 (Dai et al., 2021)	Hybrid	Conv/MBConv/Global-SA	i, n	3x3/Global	25	4.2	81.6	-	-
CoAtNet-1 (Dai et al., 2021)	Hybrid	Conv/MBConv/Global-SA	i, n	3x3/Global	42	8.4	83.3	-	-
CoAtNet-2 (Dai et al., 2021)	Hybrid	Conv/MBConv/Global-SA	i, n	3x3/Global	75	15.7	84.1	-	-
CoAtNet-3 (Dai et al., 2021)	Hybrid	Conv/MBConv/Global-SA	i, n	3x3/Global	168	34.7	84.5	-	-
LeViT-128S (Graham et al., 2021)	Hybrid	Conv/Global-SA	i,n	3x3/Global	8	0.3	76.6	64.3	83.1
LeViT-128 (Graham et al., 2021)	Hybrid	Conv/Global-SA	i,n	3x3/Global	9	0.4	78.6	66.6	84.7
LeViT-192 (Graham et al., 2021)	Hybrid	Conv/Global-SA	i,n	3x3/Global	11	0.6	80.0	68.0	85.7
LeViT-256 (Graham et al., 2021)	Hybrid	Conv/Global-SA	i,n	3x3/Global	19	1.1	81.6	70.0	86.8
LeViT-384 (Graham et al., 2021)	Hybrid	Conv/Global-SA	i,n	3x3/Global	39	2.3	82.6	71.3	87.6
MLP-Mixer-B/16 Tolstikhin et al. (2021)	MLP	Spatial Layer (SL)	n	Global	59	12.7	76.4	-	82.4
ResMLP-S12 Touvron et al. (2022)	MLP	Spatial Layer (SL)	n	Global	15	3.0	76.6	64.4	83.3
ResMLP-S24 Touvron et al. (2022)	MLP	Spatial Layer	n	Global	30	6.0	79.4	67.9	85.3
ResMLP-B24 Touvron et al. (2022)	MLP	Spatial Layer	n	Global	116	23	81.0	69.0	86.1
GFNet-H-Ti Rao et al. (2021)	MLP	FFT (SL)	n	Global	15	2.1	80.1	-	-
GFNet-H-S Rao et al. (2021)	MLP	FFT	n	Global	32	4.6	81.5	-	-
GFNet-H-B Rao et al. (2021)	MLP	FFT	n	Global	54	8.6	82.9	-	-
Poolformer-S12 (Yu et al., 2022)	CNN	Pooling	-	3x3	12	1.8	77.2	-	-
Poolformer-S24 (Yu et al., 2022)	CNN	Pooling	-	3x3	21	3.4	80.3	-	-
Poolformer-S36 (Yu et al., 2022)	CNN	Pooling	-	3x3	31	5.0	81.4	-	-
Poolformer-M36 (Yu et al., 2022)	CNN	Pooling	-	3x3	56	8.8	82.1	-	-
Poolformer-M48 (Yu et al., 2022)	CNN	Pooling	-	3x3	73	11.6	82.5	-	-
RSB-ResNet-18(A1) (Wightman et al., 2021)	CNN	Convolution	-	3x3	12	1.8	71.5	59.4	79.4
RSB-ResNet-34(A1) (Wightman et al., 2021)	CNN	Convolution	-	3x3	22	3.7	76.4	65.1	83.4
RSB-ResNet-50(A1) (Wightman et al., 2021)	CNN	Convolution	-	3x3	26	4.1	80.4	68.7	85.7
RSB-ResNet-101(A1) (Wightman et al., 2021)	CNN	Convolution	-	3x3	45	7.9	81.5	70.3	68.7
RSB-ResNet-152(A1) (Wightman et al., 2021)	CNN	Convolution	-	3x3	60	11.6	82.0	70.6	86.4
ConvNext-T (Liu et al., 2022)	CNN	Depthwise-Conv	-	7x7	29	4.5	82.1	-	-
ConvNext-S (Liu et al., 2022)	CNN	Depthwise-Conv	-	7x7	50	8.7	83.1	-	-
ConvNext-B (Liu et al., 2022)	CNN	Depthwise-Conv	-	7x7	89	15.4	83.8	-	-
Swin-Mixer-B/D24 (Liu et al., 2021b)	CNN	Local-SL	n	7x7	61	10.4	81.3	-	-
Swin-T (Liu et al., 2021b)	DCNN	Local-Self Attention (SA)	i,n	7x7	29	4.5	81.3	-	-
Swin-S (Liu et al., 2021b)	DCNN	Local-SA	i,n	7x7	50	8.7	83.0	-	-
Swin-B (Liu et al., 2021b)	DCNN	Local-SA	i,n	7x7	88	15.4	83.5	-	-
dynamic DWNet-T (Han et al., 2021)	DCNN	Dynamic DW Conv	i,n	7x7	51	3.8	81.9	-	87.3
dynamic DWNet-B (Han et al., 2021)	DCNN	Dynamic DW Conv	i,n	7x7	162	12.9	83.2	-	87.9
gMLP-S Liu et al. (2021a)	DFC	GMLP	i,n	Global	20	4.5	79.6	-	-
gMLP-B Liu et al. (2021a)	DFC	GMLP	i,n	Global	73	15.8	81.6	-	-
ViT-B/16 (Dosovitskiy et al., 2020)*	DFC	Global-SA	i,n	Global	86	17.6	79.7	-	85.0
ViT-L/16 (Dosovitskiy et al., 2020)*	DFC	Global-SA	i,n	Global	307	63.6	76.1	-	80.9
Deit-T (Touvron et al., 2021)	DFC	Global-SA	i,n	Global	5	1.3	72.2	60.4	80.1
Deit-S (Touvron et al., 2021)	DFC	Global-SA	i,n	Global	22	4.6	79.8	68.5	85.7
Deit-B (Touvron et al., 2021)	DFC	Global-SA	i,n	Global	86	17.5	81.8	71.5	86.7
PVT-T (Wang et al., 2021)	DFC	Global-SA	i,n	Global	13	1.9	75.1	-	-
PVT-S (Wang et al., 2021)	DFC	Global-SA	i,n	Global	25	3.8	79.8	-	-
PVT-M (Wang et al., 2021)	DFC	Global-SA	i,n	Global	44	6.7	81.2	-	-
PVT-L (Wang et al., 2021)	DFC	Global-SA	i,n	Global	61	9.8	81.7	-	-
Chop'D Former - N	DFC	GSAT	i,n	Global	6	1.0	76.6	64.6	83.8
Chop'D Former - T	DFC	GSAT	i,n	Global	15	2.4	80.9	69.6	86.6
Chop'D Former - S	DFC	GSAT	i,n	Global	28	4.5	82.0	70.6	86.7
Chop'D Former - M	DFC	GSAT	i,n	Global	42	6.8	82.8	72.5	87.2

Table 6: **Comparison between architectures on ImageNet classification** Trained on Imagenet 1K, input image size 224 x 224. * results taken from Tolstikhin et al. (2021)

4.2 Object detection and instance segmentation

We extend the experimental section by further testing the proposed Chop'D Former as backbone for downstream dense prediction tasks in computer vision which require pixel-level analysis of the features, namely object detection and semantic segmentation.

RetinaNet. In the main paper, we report experiments for Mask R-CNN, a widely used two-stage detector. In this ablation, we examine if our DFC block design can generalize to other types of detectors. As a first step, we provide additional experimental results with RetinaNet (Lin et al., 2017), a well-known single-stage detector. We follow Lin et al. (2017); He et al. (2017); Wang et al. (2021); Yu et al. (2022) and employ 12 epochs training schedule and use ImageNet pre-trained weights as starting point for the backbone optimization. We use the training setup of Mask R-CNN deploy-

Layer		Architecture	Complexity		Detection					
Type	Weights		P(M)	F(G)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DFC-CP	$\approx \mathbf{W_{imncd}}$	Former (Chop'D)	15	2.4	**38.8**	**58.3**	**41.2**	**21.9**	**42.1**	**51.6**
			28	4.5	**41.2**	**61.4**	**44.0**	**24.1**	**44.6**	**55.0**
FC -CP	$\approx \mathbf{W_{mncd}}$	MLP	15	2.4	-	-	-	-	-	-
			28	4.5	-	-	-	-	-	-
Conv-CP	$\approx \mathbf{W_{kcd}}$	CNN (Dw-Conv)	15	2.4	37.3	56.9	39.6	19.9	40.8	49.4
			28	4.5	40.5	60.5	43.4	23.3	44.2	53.8
Conv-CP	$\approx \mathbf{P_{kcd}}$	CNN (Pool)	15	2.4	36.7	56.8	39.0	19.8	39.8	49.0
			28	4.5	39.0	59.5	41.5	22.8	42.1	51.1
Linear-CP	$\approx \mathbf{W_{cd}}$	Linear	15	2.4	30.7	49.0	31.8	16.5	33.0	40.5
			28	4.5	31.9	50.4	33.7	16.9	33.9	42.9

Table 7: **Detection and Segmentation using RetinaNet**. Under the same complexity and training strategy, Formers outperform other classes of architecture by a large margin. They deploy DFC layers that make use of dynamic weights, generate a spatially adaptive response, and leverage a global receptive field.

Backbone	RetinaNet 1×							Mask R-CNN 1×						
	Params (M)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Params (M)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet-18 (He et al., 2016)	21.3	31.8	49.6	33.6	16.3	34.3	43.2	31.2	34.0	54.0	36.7	31.2	51.0	32.7
PoolFormer-S12 (Yu et al., 2022)	21.7	36.2	56.2	38.2	20.8	39.1	48.0	31.6	37.3	59.0	40.1	34.6	55.8	36.9
PVT-Tiny (Wang et al., 2021)	23.0	36.7	56.9	38.9	22.6	38.8	50.0	32.9	36.7	59.2	39.3	35.1	56.7	37.3
Chop'D Former-T	27.3	**38.8**	**58.4**	**41.5**	**22.3**	**42.0**	**51.5**	37.3	**40.1**	**61.4**	**43.8**	**37.1**	**58.6**	**39.6**
ResNet-50 (He et al., 2016)	37.7	36.3	55.3	38.6	19.3	40.0	48.8	44.2	38.0	58.6	41.4	34.4	55.1	36.7
PoolFormer-S24 (Yu et al., 2022)	31.1	38.9	59.7	41.3	23.3	42.1	51.8	41.0	40.1	62.2	43.4	37.0	59.1	39.6
PVT-Small (Wang et al., 2021)	34.2	40.4	61.3	43.0	25.0	42.9	55.7	44.1	40.4	62.9	43.8	37.8	60.1	40.3
Chop'D Former-S	40.9	**41.2**	**61.4**	**44.0**	**24.1**	**44.6**	**55.0**	50.8	**42.4**	**63.6**	**46.7**	**38.7**	**60.5**	**41.6**
ResNet-101 (He et al., 2016)	56.7	38.5	57.8	41.2	21.4	42.6	51.1	63.2	40.4	61.1	44.2	36.4	57.7	38.8
ResNeXt101-32x4d (Xie et al., 2017)	56.4	39.9	59.6	42.7	22.3	44.2	52.5	62.8	41.9	62.5	45.9	37.5	59.4	40.2
PoolFormer-S36 (Yu et al., 2022)	40.6	39.5	60.5	41.8	22.5	42.9	52.4	50.5	41.0	63.1	44.8	37.7	60.1	40.0
PVT-Medium (Wang et al., 2021)	53.9	41.9	63.1	44.3	25.0	44.9	57.6	63.9	42.0	64.4	45.6	39.0	61.6	42.1
Chop'D Former-M	55.5	**42.8**	**62.8**	**46.0**	**25.4**	**46.9**	**56.7**	65.3	**43.8**	**64.6**	**48.3**	**39.5**	**61.7**	**42.4**
ResNeXt101-64x4d (Xie et al., 2017)	95.5	41.0	60.9	44.0	23.9	45.2	54.0	101.9	42.8	63.8	47.3	38.4	60.6	41.3
PVT-Large (Wang et al., 2021)	71.1	42.6	63.7	45.4	25.8	46.0	58.4	81.0	42.9	65.0	46.6	39.5	61.9	42.5

Table 8: **Comparison between backbones in object detection and instance segmentation.** Methods use COCO dataset with Mask R-CNN or RetinaNet detectors. Chop'D Former shows a good trade-off between efficiency and performance, demonstrating its capacity to act as a versatile backbone for dense prediction.

ing an AdamW optimizer with an initial learning rate of $1e^{-4}$ and weight decay of $1e^{-4}$, fixing the short size of the image to 800 pixels during testing, and using a batch size of 16. In order to evaluate the ability of the DFC layers (deployed in Chop'D Formers) to work as a generalization of the FC layer, we compare different classes of architectures under the same training setup, macro design choices, and computational budget. For each of the evaluated methods, we repeat experiments with three separate random initialization and report the best performance. Results are reported in Table 7. Similarly to results found in instance segmentation experiments based on Mask R-CNN and in classification (Table 1 of the main manuscript), our Former architecture vastly outperforms its counterparts. Compared with a traditional CP decomposed CNN, Chop'D Former demonstrates a clear boost in performance with $+1.5$ AP and $+0.7$ AP for an architecture of 15 and 28 millions of parameters respectively. Note that the difference in performance across runs was no bigger than 0.2 AP, further showcasing the significance of our results. We demonstrate that, for the same complexity as a traditional convolution, DFC layers represent a complete alternative capable to reach better performance, ensuring a dynamic weight generation, a spatially adaptive response, and global reasoning by design.

Additional Comparisons. Next, we continue our analysis in Table 8 by comparing Chop'D Former of various sizes against alternative backbones for CV detectors. We extend the experimental results of Wang et al. (2021); Yu et al. (2022) and verify the effectiveness of Chop'D Former-T, Chop'D Former-S, and Chop'D Former-M as backbones for RetinaNet and Mask R-CNN. As apparent from the table, Chop'D Former models show good performance even when compared to larger models. Under a comparable number of parameters, Chop'D Former-T is 7.0 points better than ResNet18 (38.8 vs. 31.8), 2.6 points better than PoolFormer-S12 (36.2), and 2.1 points better than PVT-Tiny (36.7). Moreover, Chop'D Former-M is capable to outperform ResNeXt101-64x64 by 2.9 points (42.8 vs. 39.9), and achieving better performance than PVT-Large while using 42% fewer

parameters. These results indicate that our Chop'D Former can be a solid choice as a CNN backbone for object detection. Similar insights can be drawn from instance segmentation experiments based on Mask R-CNN. In this setup our Chop'D Former-T achieves 37.1 mask AP (APm), which is 2.5 points better than PoolFormer-S12 and even 2.0 points higher than PVT-Tiny. The best APm obtained by Chop'D Former-M is 43.8 which is 1.0 points higher than PVT-Large (43.8 vs. 42.9) and ResNetXt101-64x4d (43.8 vs. 42.8), and we are able to obtain this performance with 32% and 46% fewer parameters than these architectures, respectively.

4.3 SEMANTIC SEGMENTATION

Lastly, we evaluate Chop'D Former on the ADE20K semantic segmentation task with FPN (Kirillov et al., 2019), a simple segmentation head designed to process features extracted from its backbone. We train for $40K$ iterations with a bath size of 32, AdamW optimizer with a polynomial decay schedule, and an initial learning rate of $2e^{-4}$. We report validation mIoU as metric of performance. As for the task of object detection, we initialize the backbone of the architecture with weights pre-trained on ImageNet and initialize the remaining parameters using Xavier initialization. As visible from Table 9 Chop'D Former models can achieve competitive performance across different model capacities, further validating the effectiveness of DFC in various CV tasks. For example, Chop'D Former-M results are 2.9 points higher than ResNeXt101-64x4d (40.2 vs 43.3) while having 42% fewer parameters. Similarly, when compared with the competitive Poolformer-M48 its mIoU is still 0.6 points higher (43.3 vs. 42.7), further demonstrating the capacity of DFC layers to extract meaningful dense features from the input images.

Backbone	Semantic FPN	
	Params (M)	mIoU (%)
ResNet-18 (He et al., 2016)	15.5	32.9
PVT-Tiny (Wang et al., 2021)	17.0	35.7
PoolFormer-S12 (Yu et al., 2022)	15.7	37.2
Chop'D Former-T	21.4	39.0
ResNet-50 (He et al., 2016)	28.5	36.7
PVT-Small (Wang et al., 2021)	28.2	39.8
PoolFormer-S24 (Yu et al., 2022)	23.2	40.3
PoolFormer-S36 (Yu et al., 2022)	34.6	42.0
Chop'D Former-S	35.0	42.2
ResNet-101 (He et al., 2016)	47.5	38.8
ResNeXt-101-32x4d (Xie et al., 2017)	47.1	39.7
PVT-Medium (Wang et al., 2021)	48.0	41.6
PoolFormer-M36 (Yu et al., 2022)	59.8	42.4
PVT-Large (Wang et al., 2021)	65.1	42.1
ResNeXt-101-64x4d (Xie et al., 2017)	86.4	40.2
PoolFormer-M48 (Yu et al., 2022)	77.1	42.7
Chop'D Former-M	49.5	43.3

Table 9: **Performance of semantic segmentation on ADE 20k validation set**. Models use semantic FPN model (Kirillov et al., 2019), a simple segmentation head that only minimally processes the features. Chop'D Former acts as an effective backbone for semantic segmentation.

5 VISUAL RESULTS PUZZLE.

We provide additional visual comparisons for the image-to-image translation task of Puzzle reconstruction. Figure 2 reports extra predicted outputs for all the evaluated layers next to the GT and Input. As visible, the use of Dynamic fully connected layers yields best performance.

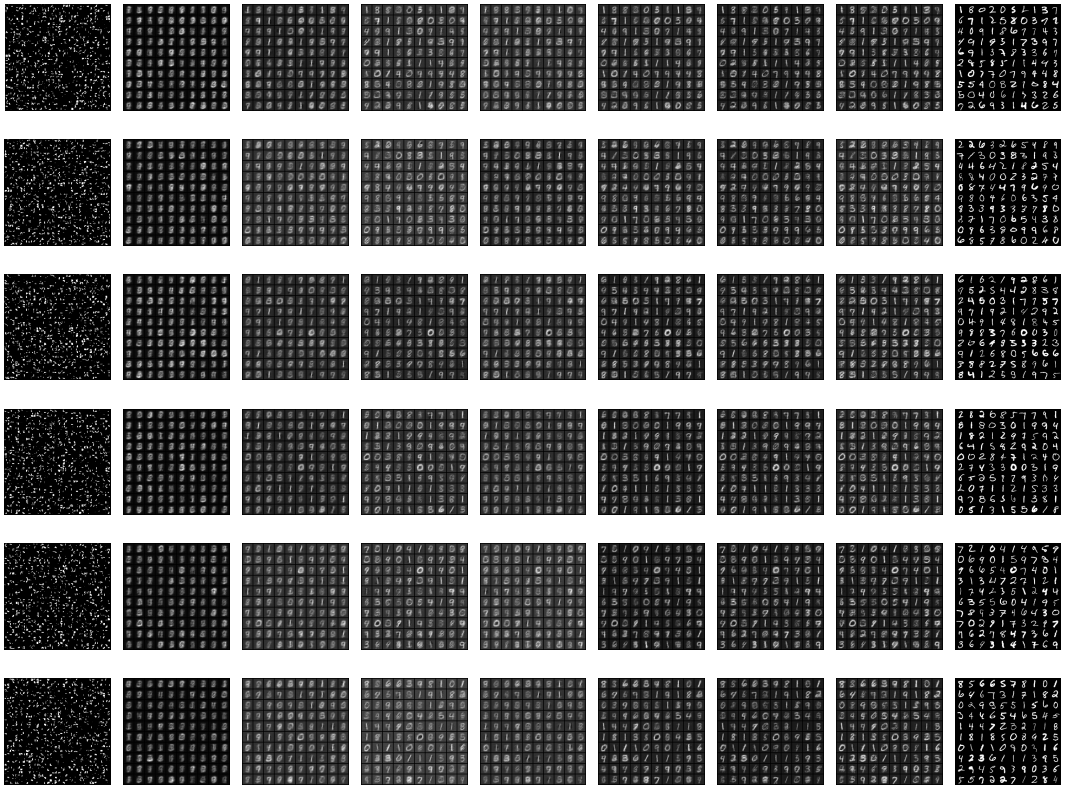


Figure 2: **Visual Comparisons for Shuffle MNIST.** Layers with different inductive biases are added between the encoder and decoder. The image-to-image translation task relies on sound global spatial reasoning and favors methods that can access the total number of tokens. Interestingly, Our CP-decomposed DFC layer approximates the results of a heavier DFC.

6 CP-DECOMPOSED FORMERS PSEUDO-CODE

We present pseudo-code for two Former variants presented in the main manuscript. Listing 1 describes implementation for a CP-Decomposed Former described in equation 6 of the main manuscript and Listing 2 describes the implementation for a Chop'D Former building block. Following notation of the main manuscript, in the pseudo-code N refers to the number of tokens, R the dimension of CP decomposition, C input channels, D output channels. Chop'D Former also takes in input the number G of dynamic pooling to learn, and achieves global token-mixing by using the `chop_gate` modulation function followed by the `dynamic_pool` function which is in practice implemented with sum-area tables (SAT).

```
# CP Decomposed Former Eq.6
def gate(x):
  mask = CNN(x)
  mask = mask.broadcast(inr)
  return x * mask + x

def former_cp_decomposition(x, C, D, R, N):
  I,N,C = x.shape                          # X_{inc}
  x_4   = pw_conv(x, in_ch=C, out_ch=R)    # U^{4}_{cr}
  x_2   = dw_conv(x_4, kernel=N)           # U^{2}_{mr}
  x_3   = x_2 * U3_parameters              # U^{3}_{nr}
  x_1   = gate(x_3)                        # U^{1}_{ir}
  x_5   = pw_conv(x_1, in_ch=R, out_ch=D) # U^{5}_{dr}
  return activation(x_5) + x
```

Listing 1: Pseudo-code for Former CP Decomposition.

```
# Chop'D Former Eq.10
def chop_gate(x):
  x_downscale = downscale(x)
  mask = pw_conv(x_downscale, in_ch=R, out_ch=R//4)
  mask = gelu(mask)
  mask = pw_conv(mask, in_ch=R//4, out_ch=R)
  mask = mask.broadcast(inr)
  return x * upscale(mask) + x

def chop_decomposition(x, C, D, R, N, G):
  I,N,C = x.shape                                  # X_{inc}
  x_4   = pw_conv(x, in_ch=C, out_ch=R)            # U^{4}_{cr}
  x_13  = chop_gate(x_4)                           # U^{13}_{inr}
  x_2   = dynamic_pool(x_13, kernel=N, groups=G)  # U^{2}_{mr}
  x_5   = pw_conv(x_2, in_ch=R, out_ch=D)          # U^{5}_{dr}
  return gelu(x_5) + x
```

Listing 2: Pseudo-code for Chop'D Former CP Decomposition.

REFERENCES

Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12259–12269, 2021.

Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. On the connection between local attention and dynamic depth-wise convolution. In *International Conference on Learning Representations*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016.

Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6399–6408, 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2502–2510, 2018.

Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *Advances in Neural Information Processing Systems*, 34:980–993, 2021.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pp. 10096–10106. PMLR, 2021.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

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

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10819–10829, 2022.

Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9308–9316, 2019.