WIDTH EXPANSION VIA PARALLEL BLOCK INTERACTION FOR VISION MODEL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Increasing the width of a vision backbone improves its representation capacity. However, directly widening increases the cost of dense channel processing. This paper argues that block-level lateral connectivity, a dimension largely abandoned since the advent of deep learning, deserves to be reopened. We propose *Parallel Block Interaction* (PBI), a general block-level width expansion framework that distributes the feature channels across multiple parallel blocks while enabling lightweight information exchange among them. Each parallel block retains the original backbone-specific block design while processing a subset of the channels. The shared public channels provide an internal information path across parallel blocks without changing the feature width passed to downstream modules. Public-to-public, private-to-public, and private-to-private channel interactions enable sufficient and controllable information exchange across blocks. On ConvNeXt, ViT, Swin Transformer, and VMamba backbones, PBI improves image classification and dense prediction performance under comparable parameter and computation budgets. Comparisons with direct channel expansion and Mixture-of-Experts, and full ablation studies show favorable accuracy–efficiency trade-offs and a clear contribution of each component. Class-level and spatial-region analysis further validates that PBI strengthens the representation diversity of the parallel blocks. PBI reveals the advantage of width expansion via block-level intra-layer connectivity, a potential model scaling dimension orthogonal to "going deep".

1 INTRODUCTION

Modern vision backbones can increase their capacity through depth and width expansion (Zagoruyko & Komodakis, 2016; Zhai et al., 2022). Depth expansion increases the number of sequentially stacked blocks, whereas width expansion increases the number of feature channels. By adding more channels, width expansion provides a higher dimensional representation space while allowing more computation to run in parallel without increasing the number of sequential blocks. However, directly widening each block increases the cost of dense channel processing. Distributing channels across narrower groups, as in grouped convolutions (Xie et al., 2017), can reduce this cost, but repeatedly processing these groups independently restricts information exchange and can weaken the representation (Zhang et al., 2018). Therefore, the benefit of width expansion depends on both how channels are processed and how information is exchanged among them. This raises a central question: *How can a vision backbone benefit from wider representations while controlling the cost of channel processing and maintaining information exchange across groups?*

Previous work has explored different ways to distribute computation and enable information exchange across branches or groups. Inception introduces multiple branches, while ResNeXt uses grouped convolutions to increase model capacity under controlled computation (Szegedy et al., 2016; Xie et al., 2017). ShuffleNet further uses channel shuffle to exchange information across groups by reassigning channels before subsequent grouped computation (Zhang et al., 2018). However, this mechanism couples feature learning with information exchange by reassigning channels across groups, potentially disrupting feature learning within each group. Dense channel mixing through pointwise convolutions provides another way to exchange information, but its cost grows rapidly with feature width. Lightweight mixing methods reduce this overhead through more selective or compact interaction (Yu et al., 2021; Liu et al., 2023). These grouping and channel interaction mechanisms show that structured computation can be combined with lightweight informa-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

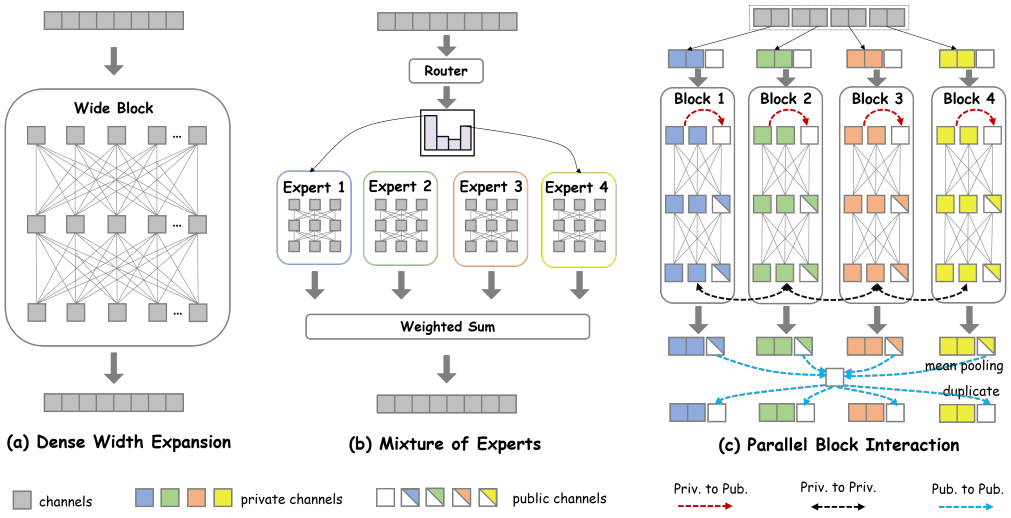


Figure 1: **Comparison of width expansion strategies.** A block denotes a complete building unit of the backbone, such as a ViT block or a ConvNeXt block. (a) Direct widening processes all channels in a single dense block. (b) Mixture-of-Experts routes inputs to experts, typically FFNs, and combines their outputs through a weighted sum. (c) PBI applies complete backbone blocks to different channel groups and enables information exchange through lightweight channel interactions.

tion exchange, but they are typically tied to specific operators and require modifying how features are processed within them. Vision Mixture-of-Experts (MoE) methods also distribute computation across multiple experts, but they are typically introduced to FFN or MLP layers (Xue et al., 2022; Riquelme et al., 2021). Moreover, increasing the number of experts mainly increases model parameter capacity without necessarily increasing the feature width retained for each input. These observations motivate a general block-level width expansion design that separates feature learning from cross block information exchange. By preserving the original block design, such a framework can apply across different vision architectures without changing the internal structure of their blocks.

To this end, we propose *Parallel Block Interaction* (PBI), a general block-level width expansion framework. Unlike the dense lateral connections in early neural networks (Hopfield, 1982; Hinton, 2007) that were discarded for trainability (Hinton et al., 2006), PBI establishes sparse, controllable, and trainable information pathways across parallel blocks. The idea is to place vision blocks side by side so that they can learn distinct features while sharing information. As illustrated in Figure 1, PBI distributes feature channels across multiple narrower parallel blocks instead of processing them within a single dense block. Each parallel block has independent parameters and retains the original block design of the backbone. The channels assigned to an individual block are referred to as *private channels* and remain associated with that block. PBI further introduces a small set of *public channels* shared across the parallel blocks. The private channels carry features specific to each block, while the public channels provide a compact shared path for information exchange. This organization preserves the total private width C while reducing the block-wise width processing cost. Since dense channel mixing typically has a quadratic cost with respect to width, when the public width is small relative to each block's private width, the quadratic channel processing cost within the blocks is approximately $1/G$ of that of a single dense block. More importantly, PBI retains the original block design, thus can be seamlessly integrated with convolutional networks, Vision Transformers, and visual state space models.

Specifically, we introduce three complementary forms of interaction among channels. The *public-to-public* channel interaction progressively integrates information from different parallel blocks as features pass through a stage, a sequence of backbone layers at the same spatial resolution and channel width. Each block jointly processes its private channels and the shared public channels, after which the output public channels are averaged across blocks and redistributed to the next layer. To strengthen the sample-specific information at the beginning of each stage, the *private-to-public* channel interaction applies a lightweight transformation to the input private channels and uses the resulting features to update the learnable public channels before the first layer. The *private-to-private* channel interaction further provides a direct information path between the private channels of dif-

ferent blocks. It extracts compact summaries of the private features, aggregates them across blocks, and redistributes the resulting information to the corresponding private channels. To reduce the computational overhead, it can be performed periodically. Together, these designs enable sufficient and controllable interaction through a small set of public channels or compact private summaries, avoiding dense matrix multiplication in channel mixing or prudential training of routing functions. Notably, PBI demonstrates the potential of reintroducing intra-layer connectivity at the block level, a design dimension orthogonal to the "going deep" direction. See Appendix A for further discussion.

We conduct experiments to evaluate the applicability, effectiveness, and behavior of PBI. Across ConvNeXt, ViT, Swin Transformer, and VMamba, PBI consistently improves ImageNet-1K classification accuracy under comparable parameter and computation budgets, with improvements also observed on downstream dense prediction tasks. Experiments on ImageNet-21K further show that these benefits extend beyond training from scratch on ImageNet-1K. Comparisons with direct channel expansion and MoE-based expansion demonstrate favorable accuracy–efficiency trade-offs in the evaluated settings. We further evaluate the three channel interactions, compare public-to-public interaction with channel shuffling and pointwise channel mixing, and examine the effects of public width and the number of parallel blocks. These studies validate the clear contribution of the proposed channel interactions. Representation analyses reveal lower feature similarity among parallel blocks, together with distinct class-level preferences and spatial responses. These observations suggest that information exchange among parallel blocks strengthens the feature representation diversity.

Our contributions are summarized as follows:

- We propose a general width expansion framework PBI, which reintroduces block-level lateral connectivity for vision model training. It distributes feature channels across multiple narrower parallel blocks while retaining the original backbone-specific block design.
- We introduce public channels as a lightweight information path across parallel blocks and develop public-to-public, private-to-public, and private-to-private channel interactions to exchange information across different combinations of shared and private channels.
- We evaluate PBI across different vision architectures and image/pixel-level tasks, demonstrating consistent improvements and favorable width-expansion efficiency.

2 METHODOLOGY

We aim to widen vision backbones through lightweight information exchange among parallel blocks. At each layer, feature channels are divided among parallel backbone blocks as described in Section 2.1 and illustrated in Figure 1. We denote the feature channels corresponding to one block as private channels, which remain associated with their corresponding block. We introduce an additional set of learnable public channels, which is attached to each set of private channels, and aggregated after layer-wise computation. Since each parallel block processes only a subset of the channels, information exchange across blocks is needed to integrate information from different channels. Specifically, public-to-public interaction aggregates and redistributes public channels to share information across parallel blocks, as formulated in Equations 2–4. Private-to-public interaction introduces information from the input private channels into the public channels at stage entry, as formulated in Equations 5. Private-to-private interaction exchanges compact summaries among the private channels of different blocks, as formulated in Equations 6. The information exchange mechanism allows the parallel blocks to share information with one another and be jointly optimized.

2.1 PARALLEL BLOCK CONSTRUCTION

A vision backbone transforms an image into intermediate representations. In CNN-based models, the representation is typically a feature map with the shape of $H \times W \times C$, where C denotes the number of feature channels. In Transformer-based models, the representation is usually a sequence of tokens with the shape of $N \times C$, where C denotes the embedding dimension of each token. Despite the different forms of these representations, commonly used vision backbones all contain a feature dimension that determines the representation width. We refer to this dimension as channels and use C to denote the number of these channels. PBI constructs parallel blocks by partitioning this channel dimension while preserving the original backbone-specific block design.

At layer l, we denote the intermediate representation as $\mathbf{x}^{(l)} \in \mathbb{R}^{N\times C}$, where N denotes the number of spatial positions or tokens. PBI evenly divides its C channels into G sets, assuming that C is divisible by G. Each subset $\mathbf{x}_g^{(l)}$, containing C/G channels, is assigned to an independent parallel block $\mathcal{B}_g^{(l)}$ and constitutes its private channels. Concatenating these channel subsets along the channel dimension recovers the complete representation,

$$\mathbf{x}^{(l)} = \left[\mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_G^{(l)}\right], \qquad \mathbf{x}_g^{(l)} \in \mathbb{R}^{N\times C/G}, \qquad g = 1, \ldots, G. \tag{1}$$

Each parallel block follows the original backbone-specific block design, while its channel width is adjusted according to its private channels and the public channels introduced later. After processing, the output private channels from all parallel blocks can be concatenated along the channel dimension to recover the representation with C channels if needed. In this way, the total number of private channels remains C, while their computation is distributed across G narrower parallel blocks.

2.2 Parallel Block Interaction

Public-to-Public Channel Interaction. The public channels are maintained and shared within each stage, where a stage refers to a sequence of consecutive layers in a vision backbone. Each stage has its own set of public channels, which are learnable and initialized with zeros. At the beginning of the stage, these public channels are copied and distributed to all parallel blocks. In each block, the public channels are concatenated with its private channels to form the block input, which is then jointly processed by the corresponding parallel block. Within each layer, the output public channels from all parallel blocks are averaged to form the shared public channels for the next layer. This aggregation and redistribution are repeated layer by layer until the end of the stage, progressively integrating information from different parallel blocks.

At layer l, we use $\mathbf{p}^{(l)} \in \mathbb{R}^{N\times P}$ to represent the P shared public channels. Each parallel block $\mathcal{B}_g^{(l)}$ receives the same public channels, concatenated with its own C/G private channels, and produces updated private and public channels as outputs. The output public channels are averaged across blocks, while the output private channels are passed to the corresponding parallel block at the next layer. The output private channels can be concatenated along the channel dimension to reconstruct the full intermediate representation, which has the same feature shape as the corresponding representation in the original backbone and can therefore be used in the same way by subsequent stages or task-specific modules. These operations can be written as

$$\left[\mathbf{x}_g^{(l+1)}, \widetilde{\mathbf{p}}_g^{(l+1)}\right] = \mathcal{B}_g^{(l)}\left(\left[\mathbf{x}_g^{(l)}, \mathbf{p}^{(l)}\right]\right), \qquad g = 1, \ldots, G, \tag{2}$$

$$\mathbf{p}^{(l+1)} = \frac{1}{G}\sum_{g=1}^{G} \widetilde{\mathbf{p}}_g^{(l+1)}, \tag{3}$$

$$\mathbf{x}^{(l+1)} = \left[\mathbf{x}_1^{(l+1)}, \ldots, \mathbf{x}_G^{(l+1)}\right]. \tag{4}$$

Here, $\widetilde{\mathbf{p}}_g^{(l+1)} \in \mathbb{R}^{N\times P}$ is the output public channels produced by block g, and $\mathbf{x}_g^{(l+1)} \in \mathbb{R}^{N\times C/G}$ is the output private channels. The output public channels remain internal to each stage and therefore do not increase the output channel count C.

Private-to-Public Channel Interaction. The public channels are progressively updated through block-level computation and public-to-public interaction after each layer. The layer-by-layer computation allows public channels to gradually incorporate information from the private channels. However, since each parallel block receives identical mean-averaged public channel representation as input, the private channel representation may encounter over-smoothness risk. We perform private-to-public interaction before the first layer of each stage as a countermeasure, ensuring that different blocks receive both public shared information and sample-specific information.

Specifically, for the first layer l of a stage, the input private channels from all parallel blocks are concatenated along the channel dimension to form $\mathbf{x}^{(l)} \in \mathbb{R}^{N\times C}$. A lightweight transformation $\mathcal{C}^{(l)}$ maps them from C channels to P channels, producing $\mathbf{c}^{(l)} \in \mathbb{R}^{N\times P}$. The shared public channels are then updated as

$$\mathbf{c}^{(l)} = \mathcal{C}^{(l)}\left(\mathbf{x}^{(l)}\right), \qquad \widehat{\mathbf{p}}^{(l)} = \mathbf{p}^{(l)} + \alpha^{(l)}\mathbf{c}^{(l)}, \tag{5}$$

where $\alpha^{(l)}$ controls the contribution of the collected information. The updated public channels $\widehat{\mathbf{p}}^{(l)}$ are then copied and distributed to all parallel blocks as the public input to the first layer. For subsequent layers, the aggregated output public channels from the preceding layer already carry information from the private channels, so this private-to-public update is not required.

Private-to-Private Channel Interaction. The interactions described above exchange information across parallel blocks through the shared public channels. Since the number of public channels is kept small for efficiency, they provide a compact shared representation rather than directly connecting the private channels of different parallel blocks. We therefore introduce a private-to-private interaction as a complementary information path, allowing the private channels of each parallel block to directly incorporate compact information from the other blocks without incurring the computational cost of densely mixing their full representations.

Specifically, the private-to-private interaction first extracts a compact summary from the private channels of each parallel block, aggregates these summaries across blocks, and then redistributes the aggregated information to the corresponding private channels. Let $\mathcal{S}_g^{(l)}$ denote the summary operator, $\mathcal{A}_g^{(l)}$ denote the aggregation operator, and $\mathcal{R}_g^{(l)}$ denote the redistribution operator. The whole interaction operation is formulated as

$$\mathbf{s}_g^{(l)} = \mathcal{S}_g^{(l)}\left(\mathbf{x}_g^{(l)}\right), \qquad \mathbf{a}_g^{(l)} = \mathcal{A}_g^{(l)}\left(\mathbf{s}_1^{(l)}, \ldots, \mathbf{s}_G^{(l)}\right), \qquad \widehat{\mathbf{x}}_g^{(l)} = \mathbf{x}_g^{(l)} + \beta_g^{(l)}\mathcal{R}_g^{(l)}\left(\mathbf{a}_g^{(l)}\right). \tag{6}$$

Here, $\beta_g^{(l)}$ controls the contribution of the aggregated information. In our implementation, $\mathcal{S}_g^{(l)}$ summarizes the private channels within local windows, $\mathcal{A}_g^{(l)}$ aggregates the resulting summaries across parallel blocks to produce a compact representation for block g, and $\mathcal{R}_g^{(l)}$ redistributes this representation to the corresponding spatial regions by broadcasting. This interaction does not need to be applied at every layer and can instead be performed periodically across several layers to reduce its computational overhead. In this way, each parallel block can incorporate information from the private channels of the other blocks without directly exchanging their private channels.

When both private-to-public interaction and private-to-private interaction are applied, $\widehat{\mathbf{p}}^{(l)}$ and $\widehat{\mathbf{x}}_g^{(l)}$ replace the corresponding public and private inputs to parallel block g, respectively. Together with the public-to-public interaction, these operations enable information exchange across different combinations of public and private channels. Detailed implementations are provided in Appendix C.

2.3 Computational Cost and Interaction Data Volume

Computational cost increases more slowly with width expansion by distributing the private channels among multiple narrower parallel blocks. Many common operations in vision backbones have computational costs that grow quadratically with channel width when both the input and output dimensions increase. For example, linear projections and MLPs in Transformers involve matrix multiplications whose cost contains a quadratic term with respect to the feature width, while the computational cost of standard convolutions in CNNs also grows quadratically with the number of channels when their input and output widths grow together. We therefore consider a transformation whose dominant computational cost at width D is proportional to D^2.

$$G\left(\frac{C}{G} + P\right)^2 = \frac{C^2}{G} + 2CP + GP^2. \tag{7}$$

Compared with directly processing all C channels in a single block, the quadratic channel transformations in the parallel blocks require approximately $1/G$ of the computation when $P \ll C/G$. Therefore, PBI slows the growth of computational cost during width expansion.

When the parallel blocks are placed on different devices, their private channels can remain local to the corresponding devices, while the interaction mechanisms exchange the required features across devices. For the public-to-public channel interaction, synchronizing P public channels over L layers gives an approximate total data volume of

$$V_{\text{pub}} = 2L(G-1)BNPs, \tag{8}$$

Table 1: **Classification performance on ImageNet-1K.** We report vision models and their PBI-expanded variants (+PBI). Smaller models are shown on the *left*, while larger variants are shown on the *right*. Subscripts denote top-1 accuracy gains over the corresponding original models. References for the listed models are provided in Appendix E.1.

Model	Type	#Params	FLOPs	Top-1
RegNetY-4G	CNN	21M	4.0G	80.0
ConvNeXt-T	CNN	29M	4.5G	82.1
+PBI (Ours)	CNN	32M	4.6G	$82.6_{+0.5}$
PVTv2-B2	Transformer	25M	4.0G	82.0
Focal-T	Transformer	29M	4.9G	82.2
FocalNet-T (LRF)	Transformer	28M	5.7G	82.2
MViTv2-T	Transformer	24M	4.7G	82.3
Swin-T	Transformer	29M	4.5G	81.3
+PBI (Ours)	Transformer	33M	5.2G	$82.5_{+1.2}$
DeiT-S	Transformer	22M	4.6G	79.8
+PBI (Ours)	Transformer	22M	4.6G	$80.0_{+0.2}$
PlainMamba-L1	Mamba	7M	3.0G	77.9
Vim-S	Mamba	26M	5.1G	80.3
LocalVim-S	Mamba	28M	4.8G	81.2
PlainMamba-L2	Mamba	25M	8.1G	81.6
EfficientVMamba-B	Mamba	33M	4.0G	81.8
LocalVMamba-T	Mamba	26M	5.7G	82.7
VMamba-T	Mamba	31M	4.9G	82.6
+PBI (Ours)	Mamba	35M	5.1G	$\mathbf{83.4}_{+0.8}$

Model	Type	#Params	FLOPs	Top-1
PVTv2-B3	Transformer	45M	7.9G	83.2
FocalNet-S (LRF)	Transformer	50M	8.7G	83.5
PVTv2-B4	Transformer	63M	10.1G	83.6
Focal-S	Transformer	51M	9.4G	83.6
MViTv2-S	Transformer	35M	7.0G	83.6
Swin-S	Transformer	50M	8.7G	83.0
+PBI (Ours)	Transformer	48M	8.5G	$83.4_{+0.4}$
LocalVMamba-S	Mamba	50M	11.4G	83.7
VMamba-S	Mamba	50M	8.7G	83.6
+PBI (Ours)	Mamba	52M	9.0G	$\mathbf{84.0}_{+0.4}$

Model	Type	#Params	FLOPs	Top-1
PVTv2-B5	Transformer	82M	11.8G	83.8
FocalNet-B (LRF)	Transformer	89M	15.4G	83.9
Focal-B	Transformer	90M	16.4G	84.0
Swin-B	Transformer	88M	15.4G	83.5
+PBI (Ours)	Transformer	87M	15.3G	$84.0_{+0.5}$
PlainMamba-L3	Mamba	50M	14.4G	82.3
VMamba-B	Mamba	89M	15.4G	83.9
+PBI (Ours)	Mamba	87M	15.1G	$\mathbf{84.2}_{+0.3}$

where B, N, and s denote the batch size, number of spatial positions or tokens, and bytes per element, respectively. This data volume depends on the public width P rather than the total private width C, and can therefore remain limited as the private width increases. More discussion and detailed derivations are provided in Appendix D.

3 Experiments

We evaluate PBI from four aspects. First, we examine its performance on ConvNeXt, ViT, Swin Transformer, and VMamba. Second, we compare PBI with direct channel expansion and MoE-based expansion. Third, we evaluate individual channel interactions and compare information interaction methods. We also examine the combined effect of the interactions and the effects of public width and the number of parallel blocks. Finally, we analyze feature similarity and class-level and spatial-region differentiation across parallel blocks to better understand the representations learned by PBI.

3.1 PBI for Width Expansion across Vision Backbones

Experimental settings. We evaluate PBI on four representative vision backbones, namely ConvNeXt, ViT, Swin, and VMamba for image classification and dense prediction. Appendix E provides the datasets, training protocols, and implementation details.

ImageNet-1K classification. Table 1 shows consistent accuracy gains across backbone families under comparable budgets. For Swin-S and VMamba-B, PBI improves accuracy by 0.4 and 0.3 percentage points, respectively, while reducing both parameters and FLOPs.

Downstream dense prediction. We further evaluate whether the expanded backbones transfer to dense prediction tasks. As reported in Table 2, PBI-expanded models consistently improve the evaluated backbones on COCO and ADE20K. These results show that the improvements are not limited to image-level recognition and extend to the evaluated dense prediction tasks across different downstream evaluation settings.

ImageNet-21K training. We further evaluate PBI on Swin Transformers trained on ImageNet-21K and fine-tuned on ImageNet-1K. Given the high cost of ImageNet-21K training, we consider Swin-T/B/L as representative variants with increasing capacity. As shown in Table 3, PBI consistently improves the fine-tuned ImageNet-1K accuracy over the corresponding Swin baselines across all three variants. These results suggest that the benefits of PBI extend beyond training from scratch on ImageNet-1K to models trained on a larger dataset and subsequently fine-tuned on ImageNet-1K.

Table 2: **Performance on downstream tasks.** We report the performance of the original models and their PBI-expanded variants (+PBI) on COCO object detection and instance segmentation (*left*) and ADE20K semantic segmentation (*right*). AP^b and AP^m denote box AP and mask AP, respectively.

Model	Mask R-CNN 1x on COCO								UPerNet on ADE20K		
	Params	**FLOPs**	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	**Params**	**FLOPs**	**mIoU**
ConvNeXt-T Liu et al. (2022)	48M	262G	44.2	66.6	48.3	40.1	63.3	42.8	60M	939G	46.0
+PBI (Ours)	48M	262G	**44.4**	**66.8**	**48.6**	**40.3**	**63.6**	**43.0**	65M	952G	**46.2**
VMamba-T Liu et al. (2024)	42M	262G	46.5	68.5	50.7	42.1	65.5	45.3	55M	939G	47.3
+PBI (Ours)	52M	282G	**47.2**	**69.4**	**51.7**	**42.7**	**66.3**	**46.2**	65M	960G	**48.5**
DeiT-S Touvron et al. (2021)	44M	450G	41.6	64.6	44.7	37.9	60.9	40.2	54M	249G	44.6
+PBI (Ours)	43M	478G	**42.0**	**65.2**	**45.1**	**38.2**	**61.4**	**40.6**	53M	252G	**45.0**
Swin-T Liu et al. (2021)	48M	267G	42.7	65.2	46.8	39.3	62.2	42.2	60M	945G	44.4
+PBI (Ours)	52M	261G	**43.3**	**65.9**	**47.6**	**39.9**	**62.5**	**42.9**	65M	964G	**44.7**
Swin-S Liu et al. (2021)	69M	354G	44.8	66.6	48.9	40.9	63.2	44.2	81M	1039G	47.6
+PBI (Ours)	67M	324G	**45.8**	**67.9**	**50.5**	**41.6**	**64.9**	**44.7**	80M	1040G	**48.1**
Swin-B Liu et al. (2021)	107M	496G	46.9	-	-	42.3	-	-	121M	1188G	48.1
+PBI (Ours)	107M	457G	**47.1**	68.7	52.0	**42.5**	65.8	46.0	123M	1195G	**48.3**

Table 3: **ImageNet-21K pre-training.** ImageNet-1K top-1 accuracy (%) after fine-tuning Swin models and their PBI variants. Δ denotes the gain over the corresponding baseline in percentage points.

Model	Params	FLOPs	Top-1	Δ
Swin-T	29M	4.5G	80.9	–
+ PBI	33M	5.2G	**82.6**	**+1.7**
Swin-B	88M	15.4G	85.2	–
+ PBI	89M	15.6G	**85.5**	**+0.3**
Swin-L	197M	34.5G	85.9	–
+ PBI	199M	34.6G	**86.3**	**+0.4**

Table 4: **Width expansion on ImageNet-1K.** A 3-layer ViT uses direct widening (Dense), MoE with two or four experts, or PBI. Chan. denotes representation width (C for PBI).

Model	Chan.	Params	FLOPs	Top-1
ViT	768	23M	4.5G	70.1
Dense	1536	88M	17.3G	73.0
MoE (2)	768	65M	10.0G	71.1
PBI	1536	49M	9.3G	**74.9**
Dense	3072	350M	67.8G	NaN
MoE (4)	768	94M	12.8G	70.6
PBI	3072	63M	11.8G	**76.5**

3.2 Comparisons with Other Width Expansion Methods

Experimental settings. We compare PBI, direct channel expansion, and MoE on ImageNet-1K using a ViT with three layers. Appendix E specifies the expansion configurations.

Expansion effectiveness. Table 4 reports accuracy, parameter count, and FLOPs for the evaluated configurations. At the same representation width, direct channel expansion uses substantially more parameters and FLOPs than PBI because it processes all channels in a single dense block. Increasing the MoE expert count from two to four does not improve accuracy in this comparison.

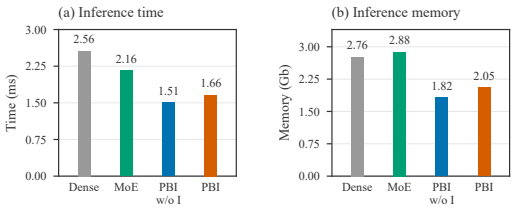


Figure 2: **Inference cost comparisons.** Execution time and memory consumption for Dense, MoE, PBI w/o I, and PBI.

At a width of 1536, PBI improves top-1 accuracy from 73.0% to 74.9% over direct channel expansion with 44.3% fewer parameters and 46.2% fewer FLOPs. At a width of 3072, PBI reaches 76.5%, whereas the directly widened model does not yield a valid accuracy result under the evaluated training setting. These results show that PBI benefits from increasing the total private width while keeping the main computation within narrower parallel blocks.

Table 5: **Channel interaction ablation.** Swin-T with PBI on ImageNet-1K. Pu and Pr denote public and private channels. Top-1 accuracy is reported in %. Replacing public channels with private channels can increase model cost, as detailed in Appendix E.2.

Pu→Pu	Pr→Pu	Pr→Pr	Params	FLOPs	Top-1
×	×	×	33.87M	5.203G	77.51
×	×	✓	34.69M	5.220G	79.62
✓	×	×	31.77M	4.985G	81.60
✓	✓	×	32.45M	5.212G	81.64
✓	×	✓	32.34M	4.997G	82.24
✓	✓	✓	33.03M	5.223G	**82.47**

Table 6: **Comparison of interaction methods.** Swin-T on ImageNet-1K with four branches and matched per-branch block widths. Top-1 accuracy is reported in %. Shuffle, 1×1 Mixing, and Public denote channel shuffling, pointwise channel mixing, and public-to-public interaction, respectively. See Appendix E.3 for corresponding implementation details.

Method	Params	FLOPs	Top-1
None	33.87M	5.203G	77.51
Shuffle	33.87M	5.203G	78.37
1×1 Mixing	42.51M	6.591G	72.86
Public	31.77M	4.985G	**81.60**

Empirical efficiency. Figure 2 compares inference time and memory consumption. In the evaluated configurations, PBI requires less time and memory than both direct channel expansion (Dense) and MoE. Compared with PBI without public channels and all three interactions (PBI w/o I), full PBI incurs additional time and memory costs. Training results are reported in Figure A6 in Appendix E.4. PBI has a similar training time to Dense and MoE. Together with the accuracy comparisons, these measurements show the balance between the performance benefits of interactions across blocks and their execution overhead.

3.3 Ablation Studies

Effects of individual channel interactions. Table 5 shows that private-to-private and public-to-public interaction alone improve accuracy from 77.51% to 79.62% and 81.60%, respectively. Combining these two reaches 82.24%, and adding private-to-public interaction further improves accuracy by 0.23 percentage points to 82.47%. With public-to-public interaction enabled, the gain from private-to-public interaction increases from 0.04 to 0.23 percentage points when private-to-private interaction is enabled. This suggests that its contribution depends on the other interaction paths available to the blocks. Public-to-public interaction thus provides the largest individual gain, with the other two interactions offering complementary benefits.

Configurations with public channels can have fewer parameters and lower reported FLOPs than those without them, which may seem counterintuitive. This is because removing public channels requires more private channels to preserve each Swin block's input width, widening the embedding, merging, and output modules and increasing their costs. Appendix E.2 provides a detailed cost breakdown and clarifies the FLOPs counting scope.

Comparison of information interaction methods. With four branches and matched block widths, public-to-public interaction achieves 81.60% accuracy, compared with 77.51% without additional interaction within a stage and 78.37% with channel shuffling (Table 6). Pointwise channel mixing reaches only 72.86% despite using more parameters and FLOPs. Shared public channels thus provide more effective information exchange in this comparison. Appendix E.3 specifies the implementations and training settings.

Effects of channel interactions and public width. Compared with removing public channels and all three interactions, PBI improves accuracy by 5.3 and 12.8 percentage points at total private widths of 384 and 3072, respectively (Figure 3(a)). Gains also hold across Swin, ConvNeXt, and VMamba (Figure 3(b)), supporting the combined contribution of the interactions. The configuration without interactions retains the parallel blocks, so the comparison examines the benefit of allowing these blocks to exchange information. The larger accuracy gap at the wider configuration suggests that such exchange becomes more valuable as the total private width increases.

In a ViT with three layers, increasing public width P improves accuracy for both four and eight blocks over the evaluated range (Figure 4(b)). Eight blocks perform worse at $P = 8, 16$ but better at $P = 24, 32$, suggesting that public width and block count should be chosen jointly.

Effects of the number of parallel blocks. At fixed total private width C, increasing the block count G reduces each block's private width to C/G and slows FLOPs growth with total width (Fig-

ure A7(a)). Under comparable parameter and computation budgets, with total private width adjusted for each configuration, accuracy first improves and then declines as G increases (Figure A7(b)). More blocks therefore do not always improve accuracy.

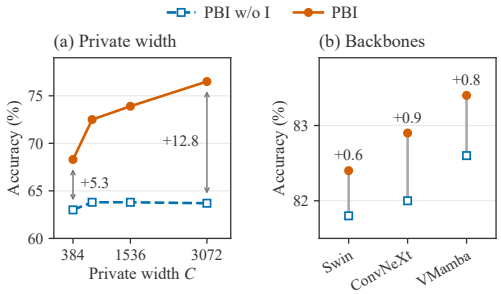


Figure 3: **Accuracy gains from channel interactions.** ImageNet-1K top-1 accuracy of PBI and PBI w/o I with (a) varying private width in a 3-layer ViT and (b) different hierarchical backbones (Swin, ConvNeXt, and VMamba).

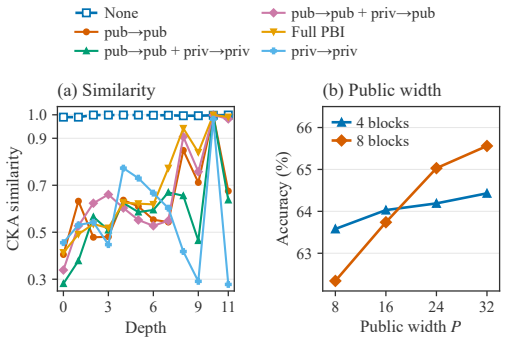


Figure 4: **Feature similarity and public width.** (a) CKA for different interactions in Swin with 12 layers. (b) Accuracy versus public width in ViT with three layers.

3.4 Analysis

Feature Similarity. CKA (Kornblith et al., 2019) in a Swin Transformer with 12 layers shows that PBI generally has lower similarity between blocks than the configuration without public channels and the proposed interactions, although the difference varies with depth (Figure 4(a)). Here, pub and priv denote public and private channels. This suggests that the blocks retain differentiated representations while exchanging information.

Class-Level Differentiation. For each block, we average the magnitudes of the classifier weights connecting its output channels to each class. This gives a class preference profile that allows us to compare how different blocks contribute to the classifier. Using classifier weight magnitude as a proxy, Figure A10 shows more distinct class preferences among PBI blocks than among channel groups produced by direct expansion.

Spatial-Region Differentiation. Grad-CAM (Selvaraju et al., 2017) maps show more varied spatial responses across PBI blocks than across blocks without the proposed interactions, highlighting different object parts or contextual regions in the visualized examples (Figure A11).

4 Conclusion and Future Work

We presented a general block-level width expansion framework PBI for vision model training by distributing feature channels across narrower parallel blocks while retaining the original block design. The shared public channels and three complementary channel interactions enable information exchange across blocks. Experiments on ConvNeXt, ViT, Swin-T, and VMamba demonstrate favorable accuracy–efficiency trade-offs on image classification and dense prediction tasks. Comparisons with direct widening and MoE-based expansion, together with component ablations, support the effectiveness of PBI. Qualitative analyses further suggest that information exchange among parallel blocks strengthens the representation diversity. PBI demonstrates the potential of reintroducing block-level intra-layer connectivity, a design dimension orthogonal to the "going deep" direction.

A limitation of our work is that the evaluation focuses on image classification and dense prediction, while the broader applicability of PBI to other tasks and modalities remains to be further explored. Future work includes investigation of automatic selection of parallel block configurations and more flexible interaction mechanisms. Notably, it will be interesting to investigate the possibility of establishing the collaboration mechanism across heterogeneous neural blocks, i.e., aggregating CNN, ViT, Mamba and even Spiking modules into one layer, reminiscing the philosophy of Minsky's Society-of-Mind (Minsky & Lee, 1988) in training and deploying vision foundation models.

AI USE STATEMENT

In this work, we used generative AI tools to assist with method implementation, translation, and interpretation of experimental results. We also used these tools to find and summarize related work, draft and polish the manuscript, and assist with figures and code. The authors reviewed the AI-assisted content, verified references, and tested the code. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies a general framework for expanding vision backbones using established benchmark datasets. It does not involve human participant studies or the collection of new personal data. We do not identify additional ethical concerns specific to the proposed method.

REPRODUCIBILITY STATEMENT

The proposed framework and experimental settings are described in Sections 2 and 3. Additional implementation details and experimental procedures are provided in Appendices C and E to support reproducibility.

REFERENCES

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Ibrahim M Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting vit in shape: Scaling laws for compute-optimal model design. *Advances in Neural Information Processing Systems*, 36:16406–16425, 2023.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts. *ArXiv preprint*, abs/2407.06204, 2024. URL https://arxiv.org/abs/2407.06204.

Vaggos Chatziafratis, Sai Ganesh Nagarajan, and Ioannis Panageas. Better depth-width trade-offs for neural networks through the lens of dynamical systems. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1469–1478. PMLR, 2020. URL http://proceedings.mlr.press/v119/chatziafratis20a.html.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *ArXiv preprint*, abs/1906.07155, 2019. URL https://arxiv.org/abs/1906.07155.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

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. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace. *arXiv preprint arXiv:2103.01197*, 2021.

Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *ArXiv preprint*, abs/2407.08083, 2024. URL https://arxiv.org/abs/2407.08083.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2980–2988. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.322. URL https://doi.org/10.1109/ICCV.2017.322.

Geoffrey E Hinton. Boltzmann machine. *Scholarpedia*, 2(5):1668, 2007.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2752–2761, 2018.

Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *ArXiv preprint*, abs/2403.09338, 2024. URL https://arxiv.org/abs/2403.09338.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 4794–4804. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00476. URL https://doi.org/10.1109/CVPR52688.2022.00476.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 14420–14430. Ieee, 2023.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model, 2024. URL https://arxiv.org/abs/2401.10166.

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 *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 9992–10002. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00986. URL https://doi.org/10.1109/ICCV48922.2021.00986.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 11966–11976. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01167. URL https://doi.org/10.1109/CVPR52688.2022.01167.

Marvin Minsky and Juliana Lee. *The society of mind*, volume 1. Simon and Schuster New York, 1988.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=KJNcAkY8tY4.

Xiaohuan Pei, Tao Huang, and Chang Xu. Efficientvmamba: Atrous selective scan for light weight visual mamba. *ArXiv preprint*, abs/2403.09977, 2024. URL https://arxiv.org/abs/2403.09977.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 10425–10433. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01044. URL https://doi.org/10.1109/CVPR42600.2020.01044.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 8583–8595, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/48237d9f2dea8c74c2a72126cf63d933-Abstract.html.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 618–626. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.74. URL https://doi.org/10.1109/ICCV.2017.74.

Deepak Sridhar, Yunsheng Li, and Nuno Vasconcelos. Scheme: Scalable channel mixer for vision transformers. *arXiv preprint arXiv:2312.00412*, 2023.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.308. URL https://doi.org/10.1109/CVPR.2016.308.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021. URL http://proceedings.mlr.press/v139/touvron21a.html.

Jianfeng Wang, Amine M'Charrak, Luk Koska, Xiangtao Wang, Daniel Petriceanu, Ruizhi Wang, Michael Bumbar, Luca Pinchetti, and Thomas Lukasiewicz. Vision hopfield memory networks for image recognition. *arXiv preprint arXiv:2603.25157*, 2026.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational visual media*, 8(3):415–424, 2022.

Zhaofeng Wu, Oliver Sieberling, Shawn Tan, Rameswar Panda, Yury Polyanskiy, and Yoon Kim. Variable-width transformers. *arXiv preprint arXiv:2606.18246*, 2026.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 418–434, 2018.

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.

Fuzhao Xue, Ziji Shi, Futao Wei, Yuxuan Lou, Yong Liu, and Yang You. Go wider instead of deeper. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 8779–8787. AAAI Press, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/20858.

Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. *ArXiv preprint*, abs/2403.17695, 2024. URL https://arxiv.org/abs/2403.17695.

Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *ArXiv preprint*, abs/2107.00641, 2021. URL https://arxiv.org/abs/2107.00641.

Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/1b08f585b0171b74d1401a5195e986f1-Abstract-Conference.html.

Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10435–10445. IEEE, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 6848–6856. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00716. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_ShuffleNet_An_Extremely_CVPR_2018_paper.html.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641, 2017.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *ArXiv preprint*, abs/2401.09417, 2024. URL https://arxiv.org/abs/2401.09417.

A Discussion on Restoring Lateral Connections at the Block Level

PBI restores lateral information flow among parallel blocks by giving them an explicit route for sharing intermediate features. Without interaction within a stage, separate channel groups can be combined at stage boundaries or at the prediction head, but cannot use one another's intermediate results during the intervening layers. The broader question of how to organize neural interactions appears in classical Hopfield networks (Hopfield, 1982) and Boltzmann machines (Ackley et al., 1985). The restricted Boltzmann machines used by Hinton et al. (2006) simplify connectivity by excluding connections within each layer. PBI revisits lateral exchange with complete backbone blocks that can process their own features and information received from other blocks.

The public pathway changes what each block can use to compute its features. After layer l, the public outputs are averaged across blocks and supplied to every block at layer $l + 1$. A feature contributed by one block can therefore influence the subsequent private outputs of other blocks. For example, evidence about an object part could become available to a branch processing complementary context. This broadens the information available to each branch during feature learning and allows interaction before the final representation is assembled. In the Swin ablation with matched input widths per block, enabling public-to-public interaction raises accuracy from 77.51% to 81.60% (Table 5), supporting the value of this information path.

The other two interactions address when information becomes available and how it reaches private features. At stage entry, private-to-public interaction supplies the public state with information from the current input, so blocks can use shared context from the first layer. Private-to-private interaction aggregates summaries from different blocks and returns a separate update to each block, providing an additional route beyond the narrow public state. Each block retains its own private channels and parameters while incorporating these updates. Public width, summary construction, and interaction frequency control the amount and timing of exchange. This combines the feature learning capacity of complete backbone blocks with explicit information exchange paths in a feedforward graph.

The shared public state also provides a concrete basis for coordination. Averaging and redistribution give every branch the same public input at the next layer, so all branches can condition their computation on a common summary of the preceding blocks. In this limited sense, the public state establishes consensus about the information made available to the branches. Their private features remain distinct and undergo different transformations. This is consistent with research on shared workspaces (Goyal et al., 2021) and communicating modules (Goyal et al., 2019). Our representation analyses likewise show lower similarity among blocks and differentiated class preferences and spatial responses (Section 3.4). PBI therefore organizes width as interacting feature streams that share context while retaining distinct representations, making lateral information exchange a concrete part of how a wider network constructs its features.

B Related Work

Vision Backbones. Modern vision backbones include convolutional, Transformer, and state-space architectures. Classical CNN and ViT backbones established convolutional and self-attention-based feature processing, while recent models such as ConvNeXt and larger Vision Transformers continue to improve their efficiency and representation quality (He et al., 2016; Dosovitskiy et al., 2021; Liu et al., 2021; 2022; Zhai et al., 2022). Visual state-space models provide another family of backbones with efficient alternatives to attention-based token mixing (Liu et al., 2024; Zhu et al., 2024; Yang et al., 2024; Hatamizadeh & Kautz, 2024). Vision Hopfield Memory Networks use local and global associative memory modules with iterative refinement to retrieve stored visual features (Wang et al., 2026). These memory modules help refine features without increasing their channel dimensions. Although these backbone families use different spatial or token mixers, they maintain intermediate features with an explicit channel dimension. PBI uses this common representation format to reorganize channels while retaining the original backbone-specific block design.

Width Expansion Methods. Width expansion is a standard way to increase the representation capacity of neural networks, and prior work has studied its interaction with depth and computational budgets (Zagoruyko & Komodakis, 2016; Chatziafratis et al., 2020; Zhai et al., 2022; Alabdulmohsin et al., 2023). Variable-Width Transformers further studies assigning different hidden

widths to different layers of language Transformers under a fixed parameter budget (Wu et al., 2026). Directly widening a block, however, increases the cost of dense channel transformations because all added channels are processed together. Earlier grouped, multi-branch, depthwise, and learned-connectivity designs reduce channel interaction through structured operators, while recent channel-mixing work continues to explore efficient alternatives (Xie et al., 2017; Chollet, 2017; Zhang et al., 2018; Huang et al., 2018; Sridhar et al., 2023). These methods typically redesign a particular operator or architecture-specific connectivity pattern, whereas PBI organizes channels into interacting parallel blocks while retaining the original backbone-specific block design. Mixture-of-Experts models provide another capacity-expansion route by activating only a subset of experts for each input, thereby increasing conditional rather than fully activated capacity (Riquelme et al., 2021; Fedus et al., 2022; Cai et al., 2024). PBI instead expands feature width. All parallel blocks process each input, while internal public channels provide a compact path for cross-block information exchange.

C Additional Method Details

This appendix specifies the interaction operators using Swin Transformer as an implementation example. We retain the notation of the main text and write $d = C/G$ for the private width of one parallel block. The window-based operators below are specific to this Swin instantiation, and the general PBI formulation does not require the same interaction operator in every backbone or stage.

C.1 Private-to-Public Channel Interaction in Swin

Figure A5(a) illustrates the transformation $\mathcal{C}^{(l)}$ in Equation 5. At the first layer of a Swin stage, a channel projection maps the concatenated private channels to width d, after which a lightweight encoder processes the projected features and produces an output with the public width P. With the batch dimension omitted, the intermediate features and output are

$$\mathbf{u}^{(l)} = \mathrm{Proj}^{(l)}_{C\to d}(\mathbf{x}^{(l)}), \qquad \mathbf{u}^{(l)} \in \mathbb{R}^{N\times d}, \tag{A9}$$

$$\mathbf{c}^{(l)} = \mathrm{Proj}^{(l)}_{d\to P}\left(\mathcal{E}^{(l)}(\mathbf{u}^{(l)})\right), \qquad \mathbf{c}^{(l)} \in \mathbb{R}^{N\times P}. \tag{A10}$$

In this Swin instantiation, $\mathcal{E}^{(l)}$ is a compact window-attention block in early stages and an MLP encoder in later stages. These choices illustrate that the internal structure of $\mathcal{C}^{(l)}$ can vary with the stage while preserving its $N \times C$ input and $N \times P$ output. The private-to-public output $\mathbf{c}^{(l)}$ is added to the stage's public channels using Equation 5. This operation is applied once at stage entry, and the subsequent layers use the public channels obtained by averaging the preceding layer's output public channels.

C.2 Private-to-Private Channel Interaction in Swin

Figure A5(b) instantiates $\mathcal{S}^{(l)}_g$, $\mathcal{A}^{(l)}_g$, and $\mathcal{R}^{(l)}_g$ in Equation 6. For features on an $H \times W$ grid, we use non-overlapping $w \times w$ windows and mean-pool the tokens in each window. Spatial dimensions are padded when necessary. The summaries from different parallel blocks are concatenated along the token dimension, giving GM key/value tokens. A multi-head attention operator with GM learnable queries aggregates information across these summaries and produces GM output tokens, which are divided into G groups for the corresponding parallel blocks,

$$M = \lceil H/w \rceil \lceil W/w \rceil, \qquad \mathbf{s}^{(l)}_g = \mathrm{WindowMean}_w(\mathbf{x}^{(l)}_g) \in \mathbb{R}^{M\times d}, \qquad g = 1,\dots,G, \tag{A11}$$

$$\mathbf{S}^{(l)} = \mathrm{Concat}_{\mathrm{token}}(\mathbf{s}^{(l)}_1,\dots,\mathbf{s}^{(l)}_G) \in \mathbb{R}^{GM\times d}, \qquad \mathbf{Q}^{(l)} \in \mathbb{R}^{GM\times d}, \tag{A12}$$

$$\mathbf{A}^{(l)} = \mathrm{Attn}(\mathbf{Q}^{(l)}, \mathbf{S}^{(l)}, \mathbf{S}^{(l)}) = \mathrm{Concat}_{\mathrm{token}}(\mathbf{a}^{(l)}_1,\dots,\mathbf{a}^{(l)}_G) \in \mathbb{R}^{GM\times d}, \qquad \mathbf{a}^{(l)}_g \in \mathbb{R}^{M\times d}. \tag{A13}$$

Each group $\mathbf{a}^{(l)}_g$ provides the aggregated window features for parallel block g, while each query can attend to the summaries from all parallel blocks. The redistribution operator broadcasts each output window feature to the spatial tokens within its corresponding window, crops any padding, and applies the residual update in Equation 6. During training, token-level stochastic dropping is

applied to mitigate overfitting. When the input resolution changes, the learnable queries are reshaped according to the parallel-block and two-dimensional window dimensions, and the window grids are interpolated to the target resolution.

The operation can be inserted periodically, with K denoting the number of applications in a stage of L layers. Window pooling reduces the spatial dimension from $N = HW$ to M, while preserving the d-dimensional private representation in each window summary. Therefore, each application exchanges GM window summaries of width d, and the amount of data exchanged depends on both the window count M and the private width d, as detailed in Appendix D.3.

C.3 Stage Boundaries and Input Resolution

At stage boundaries, PBI follows the original Swin transition. Patch merging transforms the concatenated output private channels to form the private input to the next stage. Each new stage uses its own learnable public channels, which receive a sample-dependent update from the current input private channels through the private-to-public interaction. Public-channel aggregation and redistribution are performed within each stage.

A separate issue arises when the model is applied to inputs with a different spatial resolution, such as during fine-tuning on downstream tasks. In this case, the spatial size of the feature maps and the number of windows within each stage may differ from those used during training. The learnable public feature grid is therefore spatially interpolated to the current stage resolution before the private-to-public interaction. Similarly, the learnable queries used in the private-to-private interaction are reshaped into their two-dimensional window grids and interpolated to match the target number of windows. These operations allow the interaction modules to adapt to different input resolutions without changing the underlying Swin stage transitions.

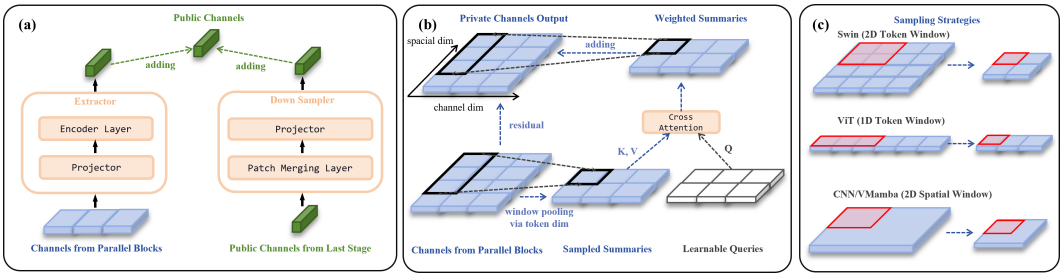


Figure A5: **Private-to-Public and Private-to-Private Channel Interaction.** (a) Private-to-Public Channel Interaction projects and encodes the input private channels to provide a sample-dependent update to the public channels at the beginning of a stage. The downsampling path from the preceding stage is optional and is omitted in our implementation. (b) Private-to-Private Channel Interaction performs window-based sampling on the private representations, uses learnable queries to aggregate the sampled summaries across parallel blocks, and broadcasts the aggregated information back to the corresponding regions with a residual connection. (c) Backbone-specific sampling strategies. Swin applies sampling on 2D token windows, ViT applies sampling on 1D token segments, and CNN/VMamba apply sampling on 2D feature maps. Thick outlines indicate the sampled windows or regions.

C.4 Effects of Public Channels on Backpropagation

Public channels enable information exchange among parallel blocks during forward propagation and provide an additional gradient path during backpropagation. We examine this effect using Equations 2 and 3. Let $\theta_i^{(l)}$ denote the parameters specific to block $\mathcal{B}_i^{(l)}$. Treating feature tensors as vectors when taking derivatives, we define the local Jacobians

$$J_{x,i}^{(l)} = \frac{\partial \mathbf{x}_i^{(l+1)}}{\partial \theta_i^{(l)}}, \qquad J_{p,i}^{(l)} = \frac{\partial \widetilde{\mathbf{p}}_i^{(l+1)}}{\partial \theta_i^{(l)}}. \tag{A14}$$

Since the output public channels are averaged across G blocks, the parameter gradient is

$$\nabla_{\theta_i^{(l)}} \mathcal{L} = \left(J_{x,i}^{(l)}\right)^\top \nabla_{\mathbf{x}_i^{(l+1)}} \mathcal{L} + \frac{1}{G} \left(J_{p,i}^{(l)}\right)^\top \nabla_{\mathbf{p}^{(l+1)}} \mathcal{L}. \tag{A15}$$

The first term propagates gradients through the output private channels. The second propagates gradients through the shared public channels, whose gradient includes contributions from subsequent blocks that use them. This second path is absent when public channels are removed. If a stage's final output public channels are discarded, their gradient contribution is zero.

To examine how one block influences the optimization of another, we consider the mixed derivative of the loss with respect to their parameters. For two distinct parallel blocks $\mathcal{B}_i^{(l)}$ and $\mathcal{B}_j^{(l)}$ at the same layer, where $i \neq j$, the block inputs do not depend on either block's layer-specific parameters. Their local Jacobians therefore do not depend on each other's parameters. Wherever the second derivatives exist, the gradient coupling is

$$\frac{\partial\left(\nabla_{\theta_i^{(l)}}\mathcal{L}\right)}{\partial\theta_j^{(l)}} = \left(J_{x,i}^{(l)}\right)^\top \frac{\partial\left(\nabla_{\mathbf{x}_i^{(l+1)}}\mathcal{L}\right)}{\partial\theta_j^{(l)}} + \frac{1}{G}\left(J_{p,i}^{(l)}\right)^\top \frac{\partial\left(\nabla_{\mathbf{P}^{(l+1)}}\mathcal{L}\right)}{\partial\theta_j^{(l)}}. \tag{A16}$$

Changing $\theta_j^{(l)}$ can change its output public channels and hence the shared public input to subsequent blocks. This can affect both gradient terms in Equation A16, allowing a block's update to depend on information contributed by other blocks within the stage. Even without public channels or the proposed channel interactions, parallel blocks can remain coupled through the shared prediction objective. Public channels provide an additional gradient path across blocks.

D Computational Cost and Interaction Data Volume

D.1 Experiment Setup

We provide a detailed analysis of the computational cost and interaction data volume introduced in Section 2.3. We first consider a constant-resolution stage with L layers. The global batch size is B, each sample contains N spatial positions or tokens, and the total private channel number C is partitioned across G blocks, each with width $d = C/G$. Each block additionally receives P public channels. For the Swin implementation, the private-to-private interaction uses M windows per block and is applied K times within each stage. Computational cost refers to the total forward arithmetic across all blocks.

To analyze the data volume of channel interactions, we assume that the $G > 1$ parallel blocks are persistently placed on G GPUs, with one block per GPU. The private features of each block remain on its corresponding GPU across consecutive layers, so only the information required by the interaction mechanisms is transferred between devices. All blocks process the same global batch. We count the global inter-GPU payload of one forward pass by summing the bytes sent between devices without counting the corresponding received copies again. Here, s denotes the number of bytes per element. We exclude protocol overhead, additional physical-link traffic, data transfers during backpropagation, and data-parallel gradient synchronization.

When analyzing width expansion, we vary the total private width C while keeping G, B, N, L, M, K, s, and the backbone block design fixed. The public width P is also kept fixed unless stated otherwise. For hierarchical backbones, the computational cost and transferred data volume are calculated for each stage using its corresponding dimensions and then summed over all stages. Operations that occur only at the input, between stages, or at the final output are counted separately from the layer-wise costs within each stage.

D.2 Computational Cost

Equation 7 isolates the quadratic channel transformations. To interpret their reduction, let F_{dense} and F_{parallel} denote these costs at the same total private width C, with the same spatial resolution and operator coefficients. The relative cost and the overhead of adding public channels to the parallel blocks are

$$\frac{F_{\text{parallel}}}{F_{\text{dense}}} = \frac{1}{G}\left(1 + \frac{P}{d}\right)^2, \qquad d = C/G, \tag{A17}$$

$$\frac{G(d+P)^2 - Gd^2}{Gd^2} = 2\frac{P}{d} + \left(\frac{P}{d}\right)^2. \tag{A18}$$

Thus, an approximately G-fold reduction of this component requires $P \ll d$. Terms that are linear in width, including depthwise convolutions and attention mixing at a fixed window size, do not receive the same reduction. For example, Swin attention in windows containing n_w tokens costs $\mathcal{O}(BNn_w(d+P))$ per block for attention-score and value-aggregation products, compared with $\mathcal{O}(BN(d+P)^2)$ for channel projections. Padding changes the token count but not this width dependence.

The interaction costs depend on their implementation and frequency. The Swin private-to-public operator uses two projections and an encoder of cost $F_{\mathcal{E}}$ once per stage. Each private-to-private application pools and redistributes the private features, projects GM queries and GM key/value tokens, and computes attention between the two sequences. Evaluating each projection once gives

$$F_{\text{private-to-public}} = \mathcal{O}(BN(Cd + dP)) + F_{\mathcal{E}}, \tag{A19}$$

$$F_{\text{private-to-private}} = \mathcal{O}\big(BNC + BGMd^2 + BG^2M^2d\big). \tag{A20}$$

The G^2M^2 term follows from the GM queries attending to all GM summaries in Equation A13. If summaries are gathered on one device, all outputs can be computed together at the above cost. If each device instead receives all summaries and evaluates only its own M queries, the attention products are partitioned across devices, but projecting the same key/value tokens locally on every device increases the aggregate projection cost to $\mathcal{O}(BG^2Md^2)$.

Let $F_{\mathcal{B}}^{(l)}$ denote the full cost of one width-$(d+P)$ backbone block at layer l, including its spatial operations. With each interaction projection evaluated once, the stage cost and full-model cost are

$$F_{\text{stage}} = \sum_{l=1}^{L} GF_{\mathcal{B}}^{(l)} + F_{\text{private-to-public}} + KF_{\text{private-to-private}} + \mathcal{O}(LGBNP), \tag{A21}$$

$$F_{\text{PBI}} = \sum_{t=1}^{T} F_{\text{stage},t} + F_{\text{boundary}}. \tag{A22}$$

Here T is the number of backbone stages, F_{boundary} includes the stem, private-feature transitions, and task-specific modules, and the last stage-level term accounts for public averaging. A schedule that repeats key/value projections must include that additional computation. These expressions separate the quadratic backbone savings from the cost of the interactions.

D.3 Data Volume of Channel Interactions

Public-to-Public Channel Interaction. Each device produces BNP public-channel elements. Equation 3 requires their mean, so the individual outputs need not be retained on every device. Reducing them to one device and broadcasting the result each send $(G-1)BNPs$ bytes. A ring AllReduce has the same aggregate payload. Counting L synchronizations, as in Equation 8, gives

$$V_{\text{pub}} = 2L(G-1)BNPs. \tag{A23}$$

An implementation that discards the final public output can omit its synchronization. Gathering all distinct outputs and averaging locally would instead cost $LG(G-1)BNPs$, without being necessary for the mean. At fixed P, public synchronization is independent of the private width d.

Private-to-Public Channel Interaction. The $N \times P$ output shape of $\mathcal{C}^{(l)}$ does not by itself determine the amount of data transferred to compute it. For the Swin implementation in Equations A9–A10, the first channel projection is linear. Partition its weight matrix along the input channels, writing $\mathbf{W}_g \in \mathbb{R}^{d\times d}$. Its output can be computed without gathering the full private representation,

$$\mathbf{u}^{(l)} = \sum_{g=1}^{G} \mathbf{x}_g^{(l)}\mathbf{W}_g^{(l)} + \mathbf{b}^{(l)}, \tag{A24}$$

$$V_{\text{private-to-public}} = (G-1)BN(d+P)s. \tag{A25}$$

Each device computes its local contribution. Reducing these width-d contributions to one device costs $(G-1)BNds$. That device adds the bias once, evaluates the encoder and final projection, and broadcasts the width-P update at cost $(G-1)BNPs$. This schedule evaluates the specified

transformation once per stage while preserving the position of its nonlinear encoder. Although the public update is narrow, the intermediate reduction still grows with d.

Private-to-Private Channel Interaction. Each block supplies a $B \times M \times d$ window summary and receives an aggregated representation of the same shape. With AllGather, each device obtains all GM summaries and computes the output for its own M queries. Alternatively, one device can gather the summaries, evaluate all GM queries in Equation A13, and scatter the resulting G output groups to their destination blocks. Over K applications,

$$V_{\text{priv,AG}} = KG(G-1)BMds, \tag{A26}$$

$$V_{\text{priv,central}} = 2K(G-1)BMds. \tag{A27}$$

For the centralized schedule, gathering the $G-1$ remote summaries and scattering the $G-1$ remote output groups each cost $(G-1)BMds$ per application. The outputs are block-specific and are not copies of a shared representation. Once its output arrives, each device performs the spatial broadcasting and residual update locally. Centralization reduces payload for $G > 2$ but places all aggregation computation on one device. For both schedules, the transferred data volume contains a term proportional to d, since window pooling compresses the spatial dimension rather than the channel dimension.

Total interaction data volume. Combining public reduction and broadcast, the stage-entry reduction and broadcast, and centralized private-to-private gather and scatter gives

$$V_{\text{int}} = (G-1)Bs\left[2LNP + N(d+P) + 2KMd\right], \tag{A28}$$

$$V_{\text{PBI}} = \sum_{t=1}^{T} V_{\text{int},t} + V_{\text{boundary}}, \tag{A29}$$

where each stage uses its own dimensions and interaction frequency. Learnable public channels and query parameters are assumed to reside on the devices that use them. With AllGather private-to-private interaction, $2KMd$ is replaced by $KGMd$ in the first line. The boundary term depends on input placement, patch merging, and task-specific modules. For example, gathering a full private representation on one device costs $(G-1)BNds$ before any output redistribution. The schedules above describe possible ways to execute the specified operators and do not represent measured distributed implementations.

D.4 Comparison under Width Expansion Setting

Data transfer baselines.

For layer parallelism, consider a constant-width model with hidden width D, split into G contiguous groups of layers. A $B \times N \times D$ activation crosses each device boundary once. Microbatching changes when and how the activations are transferred but does not change the total payload. For expert parallelism, let D_e be the routed hidden width, L_e the number of expert layers, and k the routing top-k. The BN tokens are partitioned among source devices, giving kBN assignments per expert layer. With uniformly distributed destinations independent of the source, the remote fraction is $1 - 1/G$. Dispatch and return then give

$$V_{\text{layer}} = (G-1)BNDs, \tag{A30}$$

$$V_{\text{expert}} = 2L_e k\left(1 - \frac{1}{G}\right)BND_e s. \tag{A31}$$

These estimates count activation payloads. The expert expression assumes one transfer per assignment, without dropping, padding, or coalescing assignments on the same destination device. For other routing distributions, the actual remote fraction replaces $1 - 1/G$. Routing metadata and additional collectives are separate costs. Reducing an expert's internal FFN width does not reduce its routed width D_e.

For hierarchical layer-parallel models, the first expression is replaced by $Bs\sum_b N_b D_b$, summing over device boundaries b. The expert expression similarly sums the token count and hidden width at each expert layer. These extensions preserve the dependence on model width when spatial resolutions and parallel assignments are fixed.

Parameter-matched widths.

Let Θ denote parameter count and $R = \Theta/\Theta_0$ the parameter-count ratio relative to a reference count Θ_0. For a fixed-depth dense model, dominant quadratic channel parameters give $\Theta_{\text{dense}} \approx cLD^2$, with c determined by the block design. Matching this component to PBI's $cLG(d+P)^2$ gives

$$D(R) \approx D_0\sqrt{R}, \qquad D_e(R) \approx D_{e,0}\sqrt{R}, \tag{A32}$$

$$d + P \approx D/\sqrt{G}, \qquad C = Gd \approx \sqrt{G}D - GP. \tag{A33}$$

Here D_0 and $D_{e,0}$ are the dense and MoE widths at the reference parameter count. The MoE relation holds with a fixed expert count, routing configuration, and relative expert dimensions. Its coefficient can differ from that of the dense model. The approximation for d omits interaction parameters, including the stage-entry projections and encoder. These can change the matching coefficient and must be included for exact parameter matching. In particular, the same-C computational comparison above and the parameter-matched comparison here use different constraints. The quadratic backbone cost is comparable at matched dominant parameter count. When $P \ll d$ and interaction parameters are small, PBI accommodates a total private width $C \approx \sqrt{G}D$.

Data volume under width expansion. Table A7 summarizes how the transferred data volume grows with width at fixed P. When quadratic parameter terms dominate, d grows asymptotically as $\sqrt{R}$. Public synchronization stays constant, but the other two Swin interactions transmit width-d features. The combined interaction payload therefore contains both constant and width-dependent terms.

Table A7: Forward global payload under width expansion with fixed public width P. The growth column gives the asymptotic dependence on relative parameter count R when quadratic parameter terms dominate. PBI rows describe individual interaction components.

Component or parallelization	Payload in bytes	Growth
Layer parallelism	$(G-1)BNDs$	$R^{1/2}$
Expert parallelism	$2L_ek(1-1/G)BND_es$	$R^{1/2}$
Public-to-public, fixed P	$2L(G-1)BNPs$	Constant
Swin private-to-public	$(G-1)BN(d+P)s$	$R^{1/2}$
Swin private-to-private, AllGather	$KG(G-1)BMds$	$R^{1/2}$
Swin private-to-private, centralized	$2K(G-1)BMds$	$R^{1/2}$

For a constant-resolution model with one stage-entry update, define $h = V_{\text{int}}/[(G-1)BNs]$. The centralized schedules in Equation A28 give

$$h = (2L+1)P + \left(1 + \frac{2KM}{N}\right)d, \tag{A34}$$

$$\frac{V_{\text{int}}}{V_{\text{layer}}} = \frac{h}{D}, \qquad \frac{V_{\text{int}}}{V_{\text{expert}}} = \frac{Gh}{2L_ekD_e}. \tag{A35}$$

The interaction payload is smaller than the respective baseline when $D > h$ or $D_e > Gh/(2L_ek)$. With AllGather private-to-private interaction, $h = (2L+1)P + (1+KGM/N)d$. These expressions show how public width, interaction frequency, spatial compression, and scheduling affect the comparison. Boundary traffic must be included for a whole-model comparison. Since h grows with d, the Swin ratios do not generally vanish with width. Under the approximation $d \approx D/\sqrt{G}$ at fixed P, the centralized interaction-to-layer-parallel ratio approaches $(1+2KM/N)/\sqrt{G}$.

Fixed interaction payload would require all transmitted dimensions to remain fixed. For example, private summaries could be projected to a fixed width q and exchanged by AllGather at cost $KG(G-1)BMqs$, with redistribution to d performed locally. Together with a fixed-width private-to-public transformation and fixed P, such a design would give interaction-to-baseline ratios proportional to $R^{-1/2}$. This requires changing the interaction operators. Spatial pooling alone in the present Swin implementation does not yield this dependence on model width.

E EXPERIMENT DETAILS

We evaluate PBI on ConvNeXt (Liu et al., 2022) for CNNs, ViT (Dosovitskiy et al., 2021) and Swin (Liu et al., 2021) for Transformers, and VMamba (Liu et al., 2024) for state space models. The evaluation covers ImageNet-1K classification, downstream dense prediction, and pretraining on ImageNet-21K followed by fine-tuning on ImageNet-1K.

For ImageNet-1K classification, we train PBI-expanded models following the official recipes of their corresponding baseline backbones. Unless otherwise specified, the reported classification results use top-1 accuracy on the ImageNet-1K validation set.

For ImageNet-21K pre-training, we follow the official Swin Transformer pre-training and fine-tuning protocol. Models are first pre-trained on ImageNet-21K and then fine-tuned on ImageNet-1K. We report the fine-tuned top-1 accuracy on the ImageNet-1K validation set.

For downstream evaluation, we use Mask R-CNN (He et al., 2017) in MMDetection (Chen et al., 2019) for object detection and instance segmentation on COCO (Lin et al., 2014), and UPerNet (Xiao et al., 2018) in MMSegmentation for semantic segmentation on ADE20K (Zhou et al., 2017). For dense prediction tasks with varying input resolutions, we adapt the interaction modules as described in Appendix C.3. In the Swin implementation, the learnable public feature grid is spatially interpolated to the current stage resolution, and the query grids used in Private-to-Private Channel Interaction are interpolated to match the number of windows. Window-based operations use spatial padding when needed.

Width expansion comparisons. Following prior width expansion studies (Nguyen et al., 2021; Xue et al., 2022), we compare PBI with direct channel expansion and MoE on ImageNet-1K using a ViT with three layers. Direct channel expansion and PBI increase the representation width, whereas the MoE configurations increase the number of experts while retaining a channel width of 768. This comparison examines the tradeoffs between accuracy and efficiency for these different ways of increasing model capacity.

E.1 REFERENCES FOR MODELS IN TABLE 1

The CNN models are RegNetY-4G from RegNet (Radosavovic et al., 2020) and ConvNeXt-T (Liu et al., 2022).

The Transformer models include PVTv2-B2/B3/B4/B5 (Wang et al., 2022), Focal-T/S/B (Yang et al., 2021), FocalNet-T/S/B (LRF) (Yang et al., 2022), MViTv2-T/S (Li et al., 2022), Swin-T/S/B (Liu et al., 2021), and DeiT (Touvron et al., 2021).

The state-space models include PlainMamba-L1/L2/L3 (Yang et al., 2024), Vim-S from Vision Mamba (Zhu et al., 2024), LocalVim-S and LocalVMamba-T/S from LocalMamba (Huang et al., 2024), EfficientVMamba-B (Pei et al., 2024), and VMamba-T/S/B (Liu et al., 2024).

E.2 PARAMETER AND FLOPS BREAKDOWN FOR THE INTERACTION ABLATION

To explain the parameter and FLOPs differences in Table 5, we compare the configuration with only public-to-public interaction against the configuration with all interactions disabled. Both configurations omit private-to-public and private-to-private interactions. The former uses a total private width of $C = 160$ at the first stage together with public channels. The latter replaces the public channels with private channels and uses a total private width of $C = 192$ at the first stage. These widths are summed across all parallel blocks, rather than referring to a single block. The input width of each Swin block is matched between the two configurations at each stage. This comparison therefore fixes the per-block input width, whereas the computational comparison in Appendix D.2 fixes the total private width C.

In the following parameter and FLOPs breakdowns, Final LN + classifier denotes the final LayerNorm and classification head.

Table A8 shows that the Swin blocks have identical parameter counts because their input widths are matched. Public channels remain internal to each stage, while the output private channels are concatenated for patch merging and the final normalization and classification head. Replacing public

Table A8: Module-wise parameter counts for two configurations in Table 5. Both omit private-to-public and private-to-private interactions. Difference is the private-only count minus the count with public channels.

Component	With public channels	Private-only	Difference
Patch embedding	8,160	9,792	$+1,632$
Swin blocks across all branches	26,117,728	26,117,728	0
Three patch-merging layers	4,309,760	6,203,904	$+1,894,144$
Learnable public features	47,040	0	$-47,040$
Final LN + classifier	1,283,560	1,540,072	$+256,512$
Total	31,766,248	33,871,496	$+2,105,248$

channels with private channels therefore increases the dimensions processed by these modules, as well as the patch-embedding output width. The additional parameters in these modules exceed the 47,040 parameters removed with the learnable public features, resulting in a net increase of 2,105,248 parameters. Patch merging accounts for the largest increase.

Table A9: Module-wise FLOPs for the same two configurations as in Table A8, measured for a single 224×224 image in GFLOPs. Difference is the private-only count minus the count with public channels.

Component	With public channels	Private-only	Difference
Patch embedding	0.026593	0.031912	$+0.005319$
Swin blocks across all branches	4.470682	4.470682	0
Three patch-merging layers	0.486080	0.698902	$+0.212822$
Final LN + classifier	0.001594	0.001912	$+0.000319$
Total	4.984948	5.203407	$+0.218459$

All values are rounded independently, so sums and differences of the displayed values may differ slightly from the reported totals and differences.

Table A9 shows that, for a single 224×224 image, replacing public channels with private channels increases the counted FLOPs by approximately 0.218G, or 4.38% relative to the configuration with public channels. Patch merging accounts for approximately 97.4% of this increase. The attention and MLP computations within the parallel Swin blocks remain identical because their input widths are matched. The lower reported FLOPs with public channels therefore reflect narrower private features outside the Swin blocks. The FLOPs counter does not include operations such as public-channel broadcasting and averaging. Their omission from the reported FLOPs does not imply that they incur no execution overhead.

E.3 Implementation Details for the Interaction Comparison

Table 6 compares four ways of organizing information exchange between parallel Swin blocks. All configurations use four branches, per-branch block widths of $[48, 96, 192, 384]$, stage depths of $[2, 2, 6, 2]$, and per-branch attention head counts of $[4, 8, 16, 32]$. Private-to-public and private-to-private interactions are disabled in every configuration. The first three configurations use only private channels, with a total first-stage private width of $C = 192$. The public-channel configuration uses $C = 160$ and stage-wise public widths of $[8, 16, 32, 64]$. Thus, the first-stage input width of each branch is $192/4 = 48$ in the private-only configurations and $160/4 + 8 = 48$ in the public-channel configuration. Per-branch block widths are also matched in the remaining stages.

None. Within each stage, the branches execute their Swin blocks independently without additional feature exchange between blocks. Standard patch merging still mixes the concatenated private channels at stage boundaries, and the final classification head uses the concatenated private representation. None therefore refers only to the absence of additional within-stage interaction modules.

Channel Shuffle. After each complete Swin block, the private outputs from all four branches are concatenated along the channel dimension and undergo a fixed channel permutation. The permuted channels are then split among the branches, so each branch in the next layer receives channels from all four preceding branches. This operation introduces no learnable parameters.

Pointwise Channel Mixing. After each complete Swin block, the private outputs from all branches are concatenated and transformed by a $C \times C$ linear mapping at each spatial position, where C is the total private width of the current stage. The transformed channels are then split among the branches. Each layer has its own weights, which are shared across spatial positions. No additional LayerNorm, activation, or residual connection is used. This operation mixes channels within each token without mixing different spatial positions.

Public-to-Public Interaction. Private and public channels are jointly processed within each branch's Swin block. The public outputs are averaged across branches after each block and shared as the public input to the next layer within the stage. This configuration uses neither private-to-public nor private-to-private interaction and does not include channel shuffling.

Training and cost measurement. All configurations are trained for 100 epochs with a global batch size of 1024. The learning rate, data augmentation, and DropPath settings are identical across configurations. Inputs have resolution 224×224, and FLOPs are counted for a single image with the same counter. The comparison matches per-branch block widths, while the public-channel configuration has narrower private features in patch embedding, patch merging, and the final normalization and classification head. Appendix E.2 explains the resulting whole-model cost differences. Reported FLOPs do not capture the memory-rearrangement overhead of channel shuffling, and the counter omits public-channel broadcasting and averaging. These operations can still incur execution overhead.

E.4 Additional Experimental Visualizations

The following visualizations provide additional results on training cost and the number of parallel blocks. The channel interaction and public width comparisons are shown in Figures 3 and 4 in the main text. In the comparisons involving PBI w/o I, this configuration removes public channels and all three channel interactions while retaining the partition of private channels across parallel blocks.

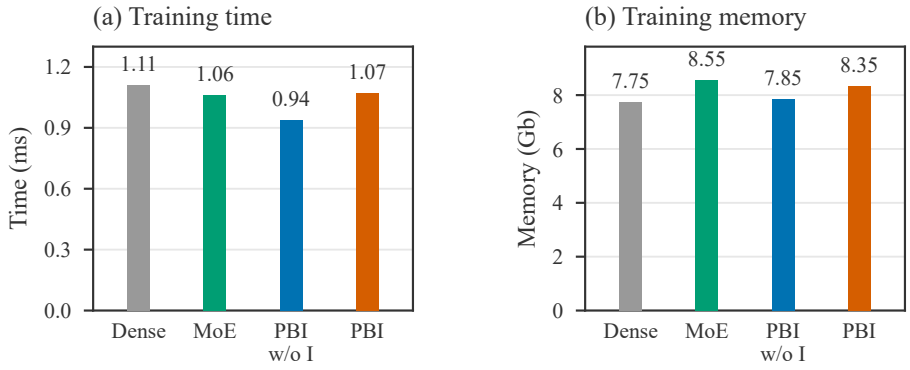


Figure A6: **Training cost comparisons.** Execution time and memory consumption for direct channel expansion within a single dense block (Dense), expansion using MoE, PBI without public channels and all three channel interactions (PBI w/o I), and PBI. Inference results are shown in Figure 2.

Single-layer analysis of parallel-block configurations. We conduct two experiments using a ViT backbone trained on ImageNet-1K for 100 epochs. In both experiments, we vary the parallel-block configuration of the fifth layer while keeping the configurations of the other layers unchanged. Only public-to-public interaction is enabled; private-to-public and private-to-private interactions are disabled. The two experiments distinguish partitioning a fixed-width representation from expanding the representation by adding fixed-width branches.

Fixed total width. Figure A8 keeps the total width of the fifth layer fixed and varies the number of parallel blocks. Increasing the block count therefore reduces the width allocated to each block. The multi-block configurations have slightly lower whole-model parameter counts and FLOPs than

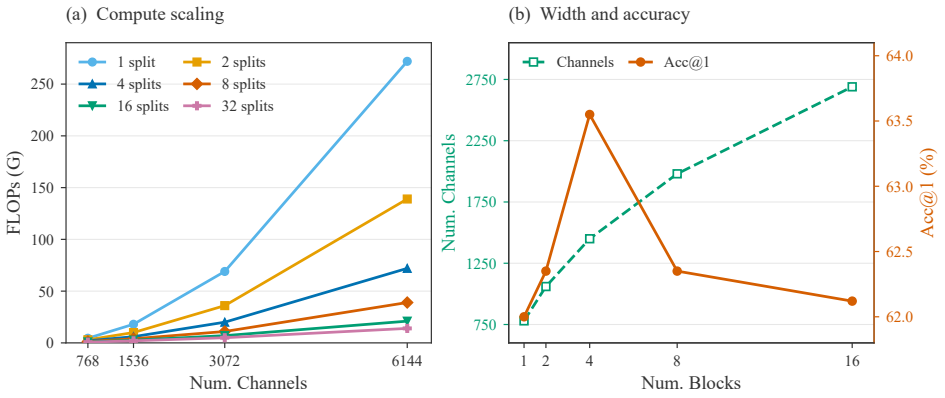


Figure A7: **Effects of the number of parallel blocks.** (a) FLOPs as a function of total private width C for different numbers of parallel blocks G. (b) Classification accuracy as a function of G under comparable parameter and computation budgets, with C adjusted for each configuration. All experiments are conducted using a 3-layer ViT on ImageNet-1K.

the single-block reference, but none surpasses its accuracy. Among the evaluated multi-block configurations, accuracy is highest with three blocks and decreases as the block count increases further. These results show that finer partitioning alone does not improve accuracy at a fixed total width in this setting.

Fixed per-branch width. Figure A9 instead fixes the width of each parallel block and increases the block count, thereby expanding the total width of the fifth layer. Accuracy initially improves as more blocks are added, while parameter count and FLOPs increase approximately linearly. At larger block counts, accuracy fluctuates rather than improving monotonically, with the highest observed accuracy of 70.84% attained at 30 blocks. This experiment shows that adding fixed-width branches can improve accuracy through width expansion, although further increases in model cost do not consistently yield additional gains.

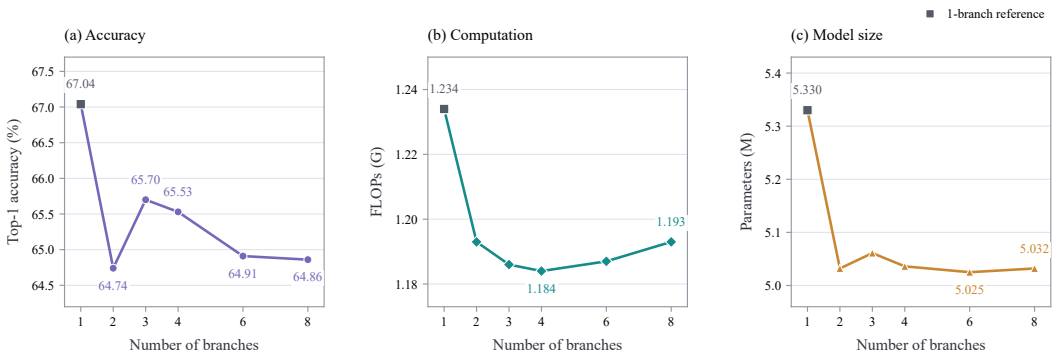


Figure A8: **Branch partitioning at fixed total width.** Top-1 accuracy, FLOPs, and parameter count when varying the number of parallel blocks in the fifth layer of a ViT while keeping its total width fixed. Models are trained on ImageNet-1K for 100 epochs using only public-to-public interaction.

F Detailed Analysis of Block Differentiation

Class-level preference measurement. To analyze whether parallel blocks develop different semantic preferences, we use the classifier head as a bridge between block representations and output classes. For each block, we compute the average magnitude of the classifier weights connecting its output channels to the logit of each class. A larger magnitude indicates a stronger association between the corresponding block and that class prediction.

Spatial-region response visualization. We further examine spatial differentiation using Grad-CAM (Selvaraju et al., 2017) maps computed from the output private channels of individual par-

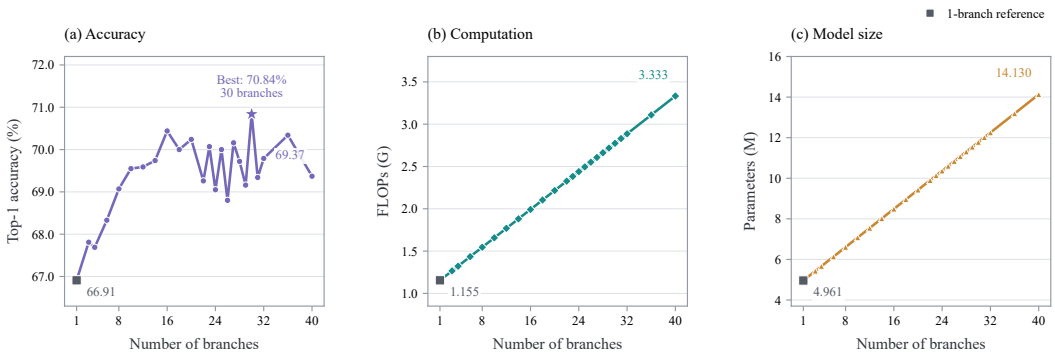


Figure A9: **Width expansion at fixed per-branch width.** Top-1 accuracy, FLOPs, and parameter count when increasing the number of parallel blocks in the fifth layer of a ViT while keeping each block's width fixed. Models are trained on ImageNet-1K for 100 epochs using only public-to-public interaction.

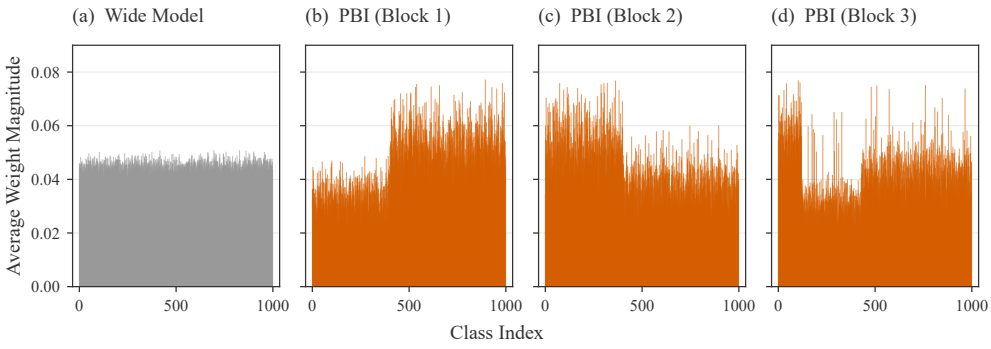


Figure A10: **Differentiation of parallel blocks in class prediction.** Experiments are conducted using a 3-layer ViT on ImageNet-1K. We report classifier-weight magnitudes for channel groups from direct channel expansion (Wide Model) and parallel blocks from PBI.

allel blocks. These maps highlight regions associated with the target class prediction, allowing us to compare spatial response patterns across blocks.

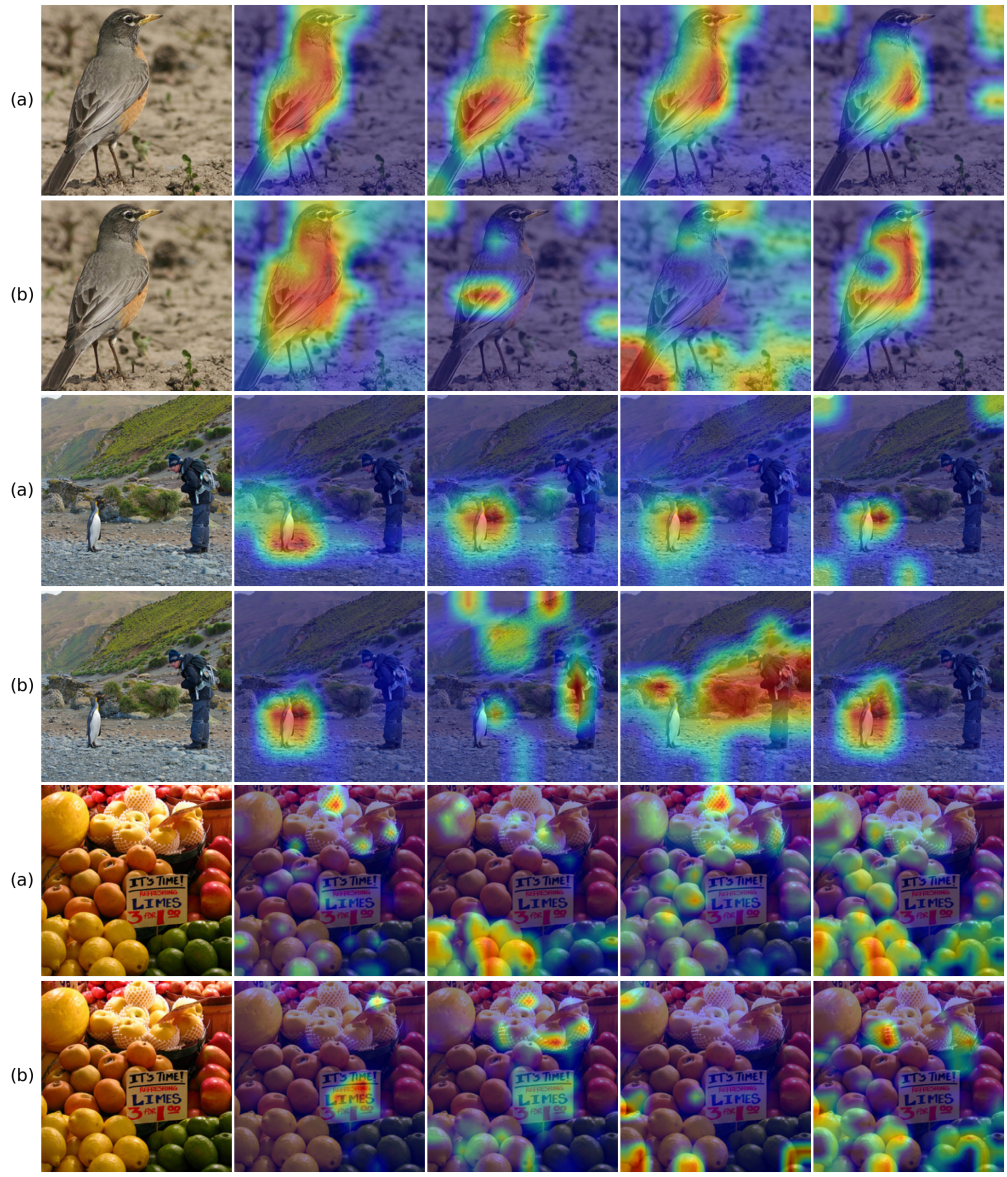


Figure A11: **Differentiation of parallel blocks on image regions.** We visualize Grad-CAM maps computed from the output private channels of individual parallel blocks in a Swin-based model. (a) Parallel blocks without the proposed interactions. (b) Parallel blocks with PBI interactions. With PBI, different parallel blocks show more differentiated spatial response patterns over the image, indicating more diverse representations.