

Figure 5: The feature maps of six samples with 3 rows and 4 columns. Each row represents a specific local modeling approach: identity (Iden.), convolution (Conv.) and SPC. The columns are the maps in different phases of Caterpillar (CPr.)-T.

A Appendix

A.1 Supplementary Visualization

As referred to in Sec. 4.4, we illustrated the SPC, Conv and Iden. based Caterpillar in Fig. 5.

A.2 Further Comparison with SOTA methods

There are some representative MLP models and recent SOTA convolutional models have published. We add a lot methods in Table A1. Due to the space limit, we only list the models with around 20-30M parameters.

Table A1: Results (%) on ImageNet-1K datasets.

Networks	Params	FLOPs	Top-1
MLP-Mixer-B/16[42]	30M	-	76.4
CycleMLP-T[2]	28M	4.4G	81.3
ConvNeXt-T[34]	29M	4.5G	82.1
SLak-T[32]	30M	5.0G	82.5
VAN-B2[13]	27M	5.0G	82.8
HorNet-T[38]	23M	3.9G	83.0
MogaNet-S[28]	25M	5.0G	83.4
InternImage-T[48]	30M	5.0G	83.5
Caterpillar-T	29M	6.0G	82.4

We compare the models in three aspects:

Performance. The model performance are mainly determined by the architecture design and training strategies. For the recent

SOTA methods like MogaNet, they mostly adopt richer strategies, such as larger batch size (i.e., 4096) and advanced augmentation (e.g., color jitter), which are effective but tricky. In our manuscript, we have compared Caterpillar with models with similar strategies, among which Caterpillar reached competitive or even superior performance. Therefore, despite the higher results of recent SOTA methods, Caterpillar is still competitive in classification capability.

Technique. Modern deep vision architectures can be decomposed into 5 levels, i.e., architecture, stage, block, module and layers, among which the spatial-mixing modules (in module level) mainly leads to the difference among various methods. MLP-Mixer and CycleMLP are early MLP models which focus on global and local information, respectively. Another models are advanced conv-based models, which all make full use of depth-wise convolution (DW-Conv), with both local and global information aggregated in parallel regimes. Caterpillar separately captures the local and global information through SPC and sparse-MLP modules.

Trend. Recent convolutional and MLP models have shown similar trends, i.e., capturing spatial information more elaborately and sparsely, from MLP-mixer, through ResMLP and to sMLPNet in MLP family, and from Conv-Mixer, through ConvNeXt, and to Moganet in Conv models. Therefore, combining lightweight techniques like depth-wise with SPC module could be promising works.

A.3 Further Explanation for Shift Methods

Shift spawns a broad class of methods, where the shifting steps, directions, dimensions as well as the restoring/learning ways mainly determines the various forms and functions for those methods. We add a detailed comparison of the SPC with some shift-base methods of different operations and applications.

Comparison with AS-MLP. The main difference between AS-MLP[29] and Caterpillar is shifting dimensions and directions. AS-MLP shifts each channel of the image along two directions, while Caterpillar shifts the entire image into four neighboring maps and encode them in individual sub-spaces, achieving parallel computing capability.

Comparison with Grouped Spatial-temporal Shift (GSTS) and X-volution. Both of the shifting operations in GSTS[26] and X-volution[4] are adopted to aggregate long-term information, with the GSTS shifted in both temporal and spatial dimensions for 3D data and X-volution shifted in more flexible directions (e.g., left-right) for wider receptive field. Different from them, the SPC is proposed with a 4-scoped receptive field to capture local information elaborately.

A.4 Efficiency Concerns

Further comparison between Caterpillar, sMLPNet and Strip-MLP. Both Strip-MLP[1] and Caterpillar are built upon sMLPNet[40] their original papers for ImageNet-1k training. Note that we don't employ 'EMA' for small-scale image recognition studies, since it decreases the performance of all models by a large margin. For the proposed Caterpillar, we list its training procedure in Table A5, which is applied for both ImageNet-1K and small-scale benchmarks.

More evaluation about efficiency. We add the comparison of SPC and conv-based ResNet in Table A2. As we can see, SPC module brings lower computational complexity than convolution, since SPC-based model achieves lower parameters and FLOPs. SPC module also provides higher Throughput and less training times. However, such trend is not with directly proportional relationship, e.g., 1/4 parameters is not bringing 4 times boosting of Throughput or 1/4 training time. This could be attribute to the lower computation vs. memory access rate in SPC module, i.e., memory access may takes more execution time than computation can not fully utilize the GPU capacity—the similar problem in depthwise separable convolution[36, 59]. We hope further optimizations on hardware computation can further improve its computational efficiency.

Table A2: Comparison between SPC- and Conv-based ResNet.

Networks	Params	FLOPs	Throughput	Training Time
Res-18	12M	1.8G	2174	20 hours
Res-18 (SPC)	3M	0.6G	4885	14 hours

A.5 Detailed Architecture Specifications

As mentioned in Section 3.3, we build our tiny, small, and base models called Caterpillar-T, -S, -B, which adopt identical backbone architectures to sMLPNet-T, -S, -B, respectively. The only difference

between the Caterpillar and sMLPNet is the local-mixing ways (i.e., SPC vs DWConv). To enable Caterpillar more friendly to limited computational resources, we also introduce the mini and tiny_x models of Caterpillar, namely Caterpillar-Mi and -Tx, which are variants of about $0.2 \times$, $0.5 \times$ the parameters and FLOPs of the -T model. Table A3 displays their detailed architectures.

A.6 Implementation of Models on Small Images

In Section 4.1, we have conducted fifteen vision models on small-scale image recognition tasks. Among them, [14, 43–45] are built with isotropic structure, [17, 50] are with 2Stage structure, [1, 2, 12, 16, 33, 40, 41, 60] and Caterpillar are with pyramid structure. For fair comparison (i.e., enabling the parameters and FLOPs of these models to be similar), we set the *patch size* to 3, 1, 1, 1 in their *patch embedding layer* for pyramid models when applied on the MIN, C10, C100 and Fashion datasets, 6, 2, 2, 2 for 2Stage models, and 12, 4, 4, 4 for Isotropic models, respectively. The feature maps in their main computational bodies on the four datasets are listed in Table A4.

A.7 Training Strategies

In Table A6, we present the training strategies for all models adopted in Section 4.1. These strategies are the same as those in their original papers for ImageNet-1k training. Note that we don't employ 'EMA' for small-scale image recognition studies, since it decreases the performance of all models by a large margin. For the proposed Caterpillar, we list its training procedure in Table A5, which is applied for both ImageNet-1K and small-scale benchmarks.

A.8 Sparse-MLP Module

The sparse-MLP (sMLP) module is proposed in [40] and also adopted in the Caterpillar block for aggregating global information. To have a comprehensive understanding of the proposed Caterpillar, we also depict the sMLP module in Figure 6. As we can see, the sMLP module consists of three branches: two of them are used to mix information along horizontal and vertical directions, respectively, which is implemented by two $H(W) \times H(W)$ linear projections, and the other path is an identity mapping. The output of the three branches are concatenated and then mixed by a $3C \times C$ linear projection to obtain the final output. Through the sMLP calculation, each pillar can gather information from other pillars in the same row and column. Stacking more sMLP blocks allows for the mixing of the gathered features across different rows and columns, with all pillars incorporating the global information of the entire image.

Table A3: Detailed settings of Caterpillar series.

Stages	Caterpillar-Mi	Caterpillar-Tx	Caterpillar-T	Caterpillar-S	Caterpillar-B
S1	patch_size: 4 [56×56, 40]×2	patch_size: 4 [56×56, 60]×2	patch_size: 4 [56×56, 80]×2	patch_size: 4 [56×56, 96]×2	patch_size: 4 [56×56, 112]×2
S2	downsp. rate: 2 [28×28, 80]×6	downsp. rate: 2 [28×28, 120]×8	downsp. rate: 2 [28×28, 160]×8	downsp. rate: 2 [28×28, 192]×10	downsp. rate: 2 [28×28, 224]×10
S3	downsp. rate: 2 [14×14, 160]×10	downsp. rate: 2 [14×14, 240]×14	downsp. rate: 2 [14×14, 320]×14	downsp. rate: 2 [14×14, 384]×24	downsp. rate: 2 [14×14, 448]×24
S4	downsp. rate: 2 [7×7, 320]×2	downsp. rate: 2 [7×7, 480]×2	downsp. rate: 2 [7×7, 640]×2	downsp. rate: 2 [7×7, 768]×2	downsp. rate: 2 [7×7, 896]×2

Table A4: Feature maps in models with different architectures on four small-scale benchmarks. C denotes the channel number of the used models in their first stage.

Architecture	Stages	MIN	C10	C100	Fashion
Pyramid	S1	[28×28, C]	[32×32, C]	[32×32, C]	[28×28, C]
	S2	[14×14, 2C]	[16×16, 2C]	[16×16, 2C]	[14×14, 2C]
	S3	[7×7, 4C]	[8×8, 4C]	[8×8, 4C]	[7×7, 4C]
	S4	[7×7, 8C]	[4×4, 8C]	[4×4, 8C]	[7×7, 8C]
2Stage	S1	[14×14, C]	[16×16, C]	[16×16, C]	[14×14, C]
	S2	[7×7, 2C]	[8×8, 2C]	[8×8, 2C]	[7×7, 2C]
Isotropic	S1	[7×7, C]	[8×8, C]	[8×8, C]	[7×7, C]

Table A5: Training strategies for Caterpillar models

Configs	Caterpillar Mi, Tx, T, S, B
Training epochs	300
Batch size	1024
Optimizer	AdamW
LR	1e-3
LR decay	cosine
Min LR	1e-5
Weight_decay	0.05
Warmup epochs	5
Warmup LR	1e-6
Rand Augment	9/0.5
Mixup	0.8
Cutmix	1.0
Stoch. Depth	0, 0, 0.05, 0.2, 0.3
Repeated Aug	✓
Erasing prob.	0.25
Label smoothing	0.1
EMA	0.99996

Table A6: Training strategies for various vision models

Configs	ResNet 18, 34, 50 [51]	ConvMixer 768/32 [45]	DeiT T, S [44]	Swin T [33]	CCT 7/3×1 [14]	NesT T [60]	ResMLP S12, S24 [43]
Training epochs	300	300	300	300	300	300	400
Batch size	2048	640	1024	1024	1024	512	1024
Optimizer	LAMB	AdamW	AdamW	AdamW	AdamW	AdamW	LAMB
LR	5e-3	1e-2	1e-3	1e-3	5e-4	5e-4	5e-3
LR decay	cosine	onecycle	cosine	cosine	cosine	cosine	cosine
Min LR	1e-6	1e-6	1e-5	5e-6	1e-5	0	1e-5
Weight_decay	0.02	0.00002	0.05	0.05	0.05	0.05	0.2
Warmup epochs	5	0	5	20	10	20	5
Warmup LR	1e-4	–	1e-6	5e-7	1e-6	1e-6	1e-6
Rand Augment	7/0.5	9/0.5	9/0.5	9/0.5	9/0.5	9/0.5	9/0.5
Mixup	0.1	0.5	0.8	0.8	0.8	0.8	0.8
Cutmix	1.0	0.5	1.0	1.0	1.0	1.0	1.0
Stoch. Depth	0.05	0	0.1	0.2	0	0.2	0.1
Repeated Aug	✓	✗	✓	✗	✗	✗	✓
Erasing prob.	0	0.25	0.25	0.25	0.25	0.25	0.25
Label smoothing	0	0.1	0.1	0.1	0.1	0.1	0.1
EMA	–	–	–	–	–	–	–
Configs	CycleMLP B1, B2 [2]	HireMLP Ti, S [12]	Wave-MLP T, S [41]	ViP S7 [17]	DynaMixer S [50]	sMLPNet T [40]	Strip-MLP T*, T [1]
Training epochs	300	300	300	300	300	300	300
Batch size	1024	2048, 1024	1024	2048	2048	1024	1024
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
LR	1e-3	1e-3	1e-3	2e-3	2e-3	1e-3	1e-3
LR decay	cosine	cosine	cosine	cosine	cosine	cosine	cosine
Min LR	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	5e-6
Weight_decay	0.05	0.05	0.05	0.05	0.05	0.05	0.05
Warmup epochs	5	20	5	20	20	20	30
Warmup LR	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6	5e-7
Rand Augment	9/0.5	9/0.5	9/0.5	9/0.5	9/0.5	9/0.5	9/0.5
Mixup	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Cutmix	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Stoch. Depth	0.1	0	0.1	0.1	0.1	0	0.2
Repeated Aug	✓	✓	✓	✗	✗	✓	✗
Erasing prob.	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Label smoothing	0.1	0.1	0.1	0.1	0.1	0.1	0.1
EMA	0.99996	–	0.99996	–	0.99996	0.99996	–

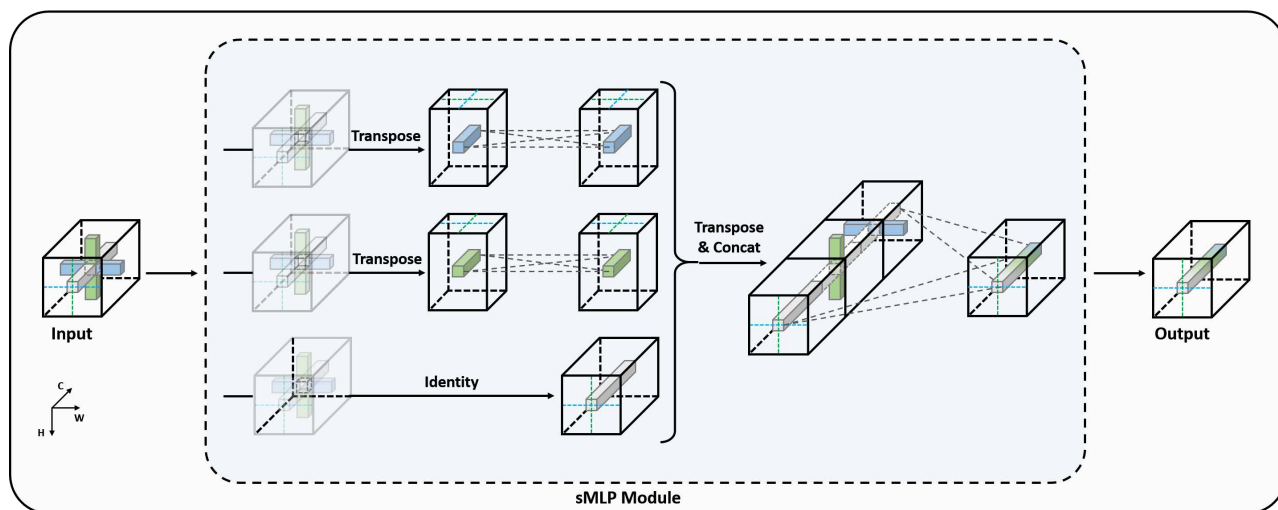


Figure 6: The sparse-MLP module proposed in sMLPNet [40]