

Transformers and large-kernel CNNs

Vlad Shakhuro



23 October 2025

Outline

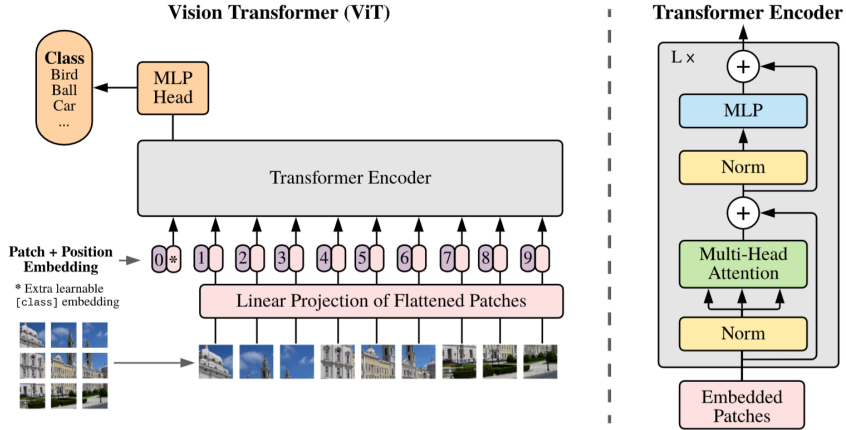
1. Vision Transformer

2. Swin Transformer

3. ConvNeXt

4. MobileNetV4

Vision Transformer



Dosovitskiy et al. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. ICLR 2021

Multi-Head Self-Attention

$$[q, k, v] = zU_{qkv},$$

$$z \in \mathbb{R}^{N \times D}, \quad U_{qkv} \in \mathbb{R}^{D \times 3D_h}$$

$$A = \text{softmax}(qk^T / \sqrt{D_h}),$$

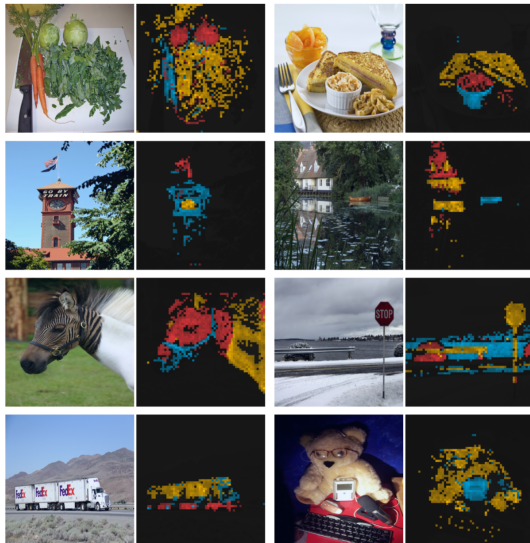
$$A \in \mathbb{R}^{N \times N}$$

$$\text{SA}(z) = Av$$

$$\text{MSA}(z) = [\text{SA}_1(z); \text{SA}_2(z); \dots; \text{SA}_k(z)]U_{msa},$$

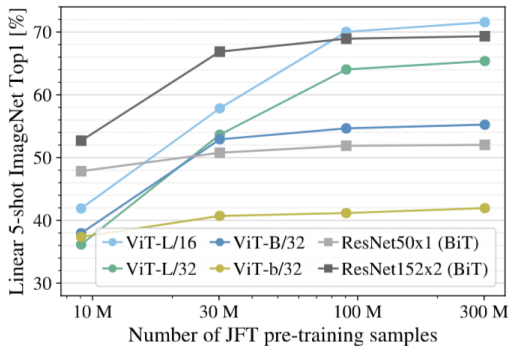
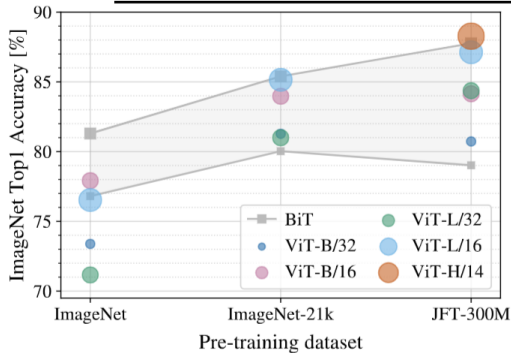
$$U_{msa} \in \mathbb{R}^{k \cdot D_h \times D}$$

MSA visualization, 8×8 patches



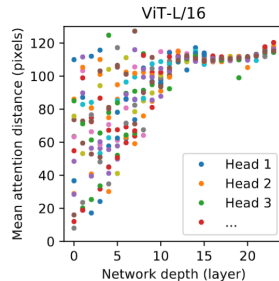
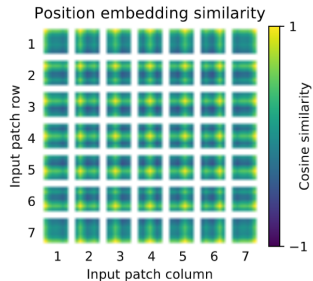
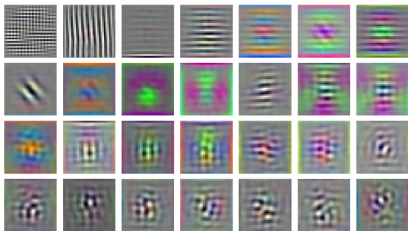
Vision Transformer

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M



Vision Transformer

RGB embedding filters
(first 28 principal components)



Data-Efficient Transformer

Methods	ViT-B	DeiT-B
Epochs	300	300
Batch size	4096	1024
Optimizer	AdamW	AdamW
learning rate	0.003	$0.0005 \times \frac{\text{batchsize}}{512}$
Learning rate decay	cosine	cosine
Weight decay	0.3	0.05
Warmup epochs	3.4	5
Label smoothing ϵ	$\times$	0.1
Dropout	0.1	$\times$
Stoch. Depth	$\times$	0.1
Repeated Aug	$\times$	$\checkmark$
Gradient Clip.	$\checkmark$	$\times$
Rand Augment	$\times$	9/0.5
Mixup prob.	$\times$	0.8
Cutmix prob.	$\times$	1.0
Erasing prob.	$\times$	0.25

Network	nb of param.	image size	im/s	ImNet top-1	Real top-1	V2 top-1
ResNet-18	12M	224	4458.4	69.8	77.3	57.1
ResNet-50	25M	224	1226.1	76.2	82.5	63.3
ResNet-101	45M	224	753.6	77.4	83.7	65.7
ResNet-152	60M	224	526.4	78.3	84.1	67.0
RegNetY-4GF*	21M	224	1156.7	80.0	86.4	69.4
RegNetY-8GF*	39M	224	591.6	81.7	87.4	70.8
RegNetY-16GF*	84M	224	334.7	82.9	88.1	72.4
EfficientNet-B0	5M	224	2694.3	77.1	83.5	64.3
EfficientNet-B1	8M	240	1662.5	79.1	84.9	66.9
EfficientNet-B2	9M	260	1255.7	80.1	85.9	68.8
EfficientNet-B3	12M	300	732.1	81.6	86.8	70.6
EfficientNet-B4	19M	380	349.4	82.9	88.0	72.3
EfficientNet-B5	30M	456	169.1	83.6	88.3	73.6
EfficientNet-B6	43M	528	96.9	84.0	88.8	73.9
EfficientNet-B7	66M	600	55.1	84.3	-	-
EfficientNet-B5 RA	30M	456	96.9	83.7	-	-
EfficientNet-B7 RA	66M	600	55.1	84.7	-	-
KDforAA-B8	87M	800	25.2	85.8	-	-
Transformers: training 300 epochs						
ViT-B/16	86M	384	85.9	77.9	83.6	-
ViT-L/16	307M	384	27.3	76.5	82.2	-
DeiT-Ti	5M	224	2536.5	72.2	80.1	60.4
DeiT-S	22M	224	940.4	79.8	85.7	68.5
DeiT-B	86M	224	292.3	81.8	86.7	71.5
DeiT-B $\uparrow$ 384	86M	384	85.9	83.1	87.7	72.4
DeiT-Ti*	6M	224	2529.5	74.5	82.1	62.9
DeiT-S*	22M	224	936.2	81.2	86.8	70.0
DeiT-B*	87M	224	290.9	83.4	88.3	73.2
DeiT-B* $\uparrow$ 384	87M	384	85.8	84.5	89.0	74.8
Transformers: training 1000 epochs						
DeiT-Ti*	6M	224	2529.5	76.6	83.9	65.4
DeiT-S*	22M	224	936.2	82.6	87.8	71.7
DeiT-B*	87M	224	290.9	84.2	88.7	73.9
DeiT-B* $\uparrow$ 384	87M	384	85.8	85.2	89.3	75.2

*: our trained teachers with SGD, whose optimization procedure is closer to DeiT

Outline

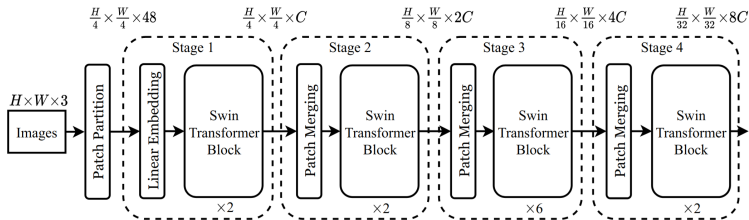
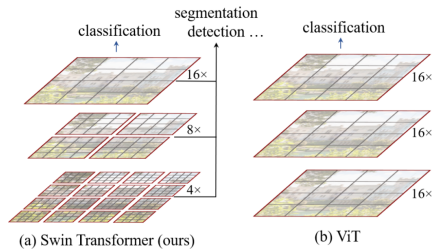
1. Vision Transformer

2. Swin Transformer

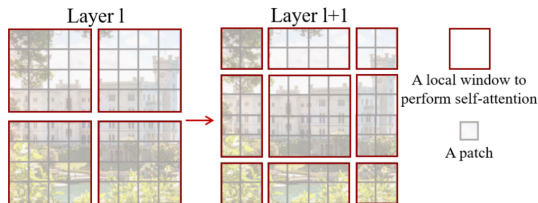
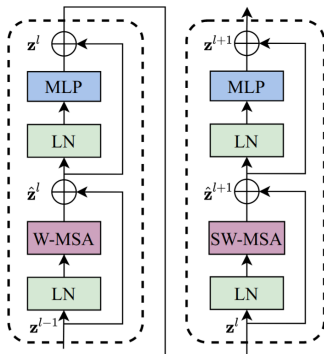
3. ConvNeXt

4. MobileNetV4

Swin Transformer



Shifted window attention



Include positional information in self-attention:

$$SA(q, k, v) = \text{softmax}(qk^t / \sqrt{D_h} + b)v$$

$$b \in \mathbb{R}^{N \times N}$$

Swin variants

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 96, \text{ head } 3 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 96, \text{ head } 3 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 128, \text{ head } 4 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 192, \text{ head } 6 \end{bmatrix} \times 2$
stage 2	8× (28×28)	concat 2×2, 192-d, LN	concat 2×2, 192-d, LN	concat 2×2, 256-d, LN	concat 2×2, 384-d, LN
		$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 192, \text{ head } 6 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 192, \text{ head } 6 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 256, \text{ head } 8 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 384, \text{ head } 12 \end{bmatrix} \times 2$
stage 3	16× (14×14)	concat 2×2, 384-d, LN	concat 2×2, 384-d, LN	concat 2×2, 512-d, LN	concat 2×2, 768-d, LN
		$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 384, \text{ head } 12 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 384, \text{ head } 12 \end{bmatrix} \times 18$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 512, \text{ head } 16 \end{bmatrix} \times 18$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 768, \text{ head } 24 \end{bmatrix} \times 18$
stage 4	32× (7×7)	concat 2×2, 768-d, LN	concat 2×2, 768-d, LN	concat 2×2, 1024-d, LN	concat 2×2, 1536-d, LN
		$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 768, \text{ head } 24 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 768, \text{ head } 24 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 1024, \text{ head } 32 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7 \times 7, \\ \text{dim } 1536, \text{ head } 48 \end{bmatrix} \times 2$

Swin Transformer

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224 ²	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224 ²	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224 ²	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300 ²	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380 ²	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456 ²	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528 ²	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600 ²	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384 ²	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384 ²	307M	190.7G	27.3	76.5
DeiT-S [63]	224 ²	22M	4.6G	940.4	79.8
DeiT-B [63]	224 ²	86M	17.5G	292.3	81.8
DeiT-B [63]	384 ²	86M	55.4G	85.9	83.1
Swin-T	224 ²	29M	4.5G	755.2	81.3
Swin-S	224 ²	50M	8.7G	436.9	83.0
Swin-B	224 ²	88M	15.4G	278.1	83.5
Swin-B	384 ²	88M	47.0G	84.7	84.5

Outline

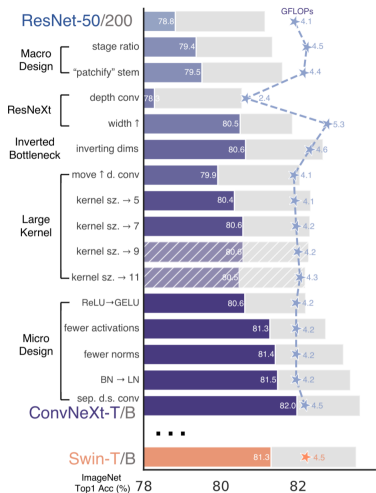
1. Vision Transformer

2. Swin Transformer

3. ConvNeXt

4. MobileNetV4

ConvNeXt

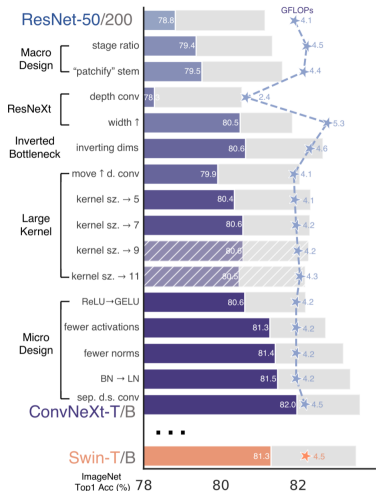


Baseline ResNet-50 with modern training:

- 90 → 300 epochs
- AdamW optimizer
- Augmentations: Mixup, Cutmix, RandAugment, Random Erasing
- Regularization: Stoch. Depth, Label Smoothing

76.1% → 78.8% on ImageNet

ConvNeXt

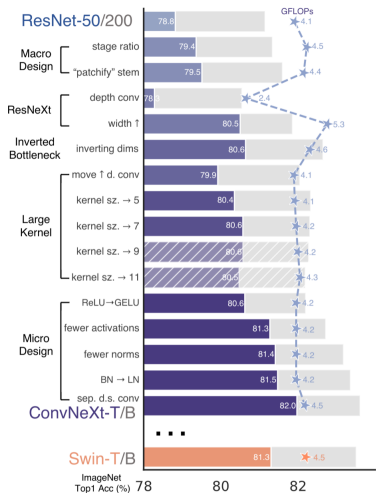


Macro design

Change #blocks
 $(3, 4, 6, 3) \rightarrow (3, 3, 9, 3)$

Change "patchify" stem
 $7 \times 7 \text{ conv}/2, \text{ pool}/2 \rightarrow 4 \times 4 \text{ conv}/4$

ConvNeXt



Inverted bottleneck

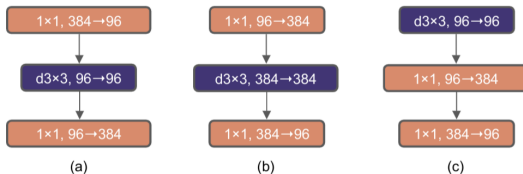
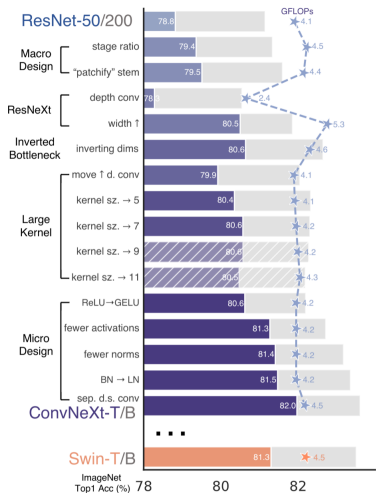


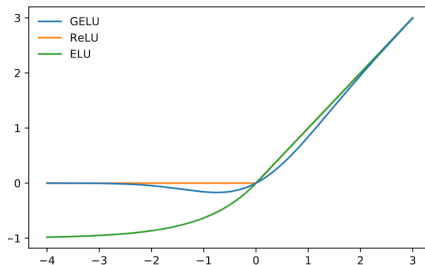
Figure 3. **Block modifications and resulted specifications.** (a) is a ResNeXt block; in (b) we create an inverted bottleneck block and in (c) the position of the spatial depthwise conv layer is moved up.

ConvNeXt

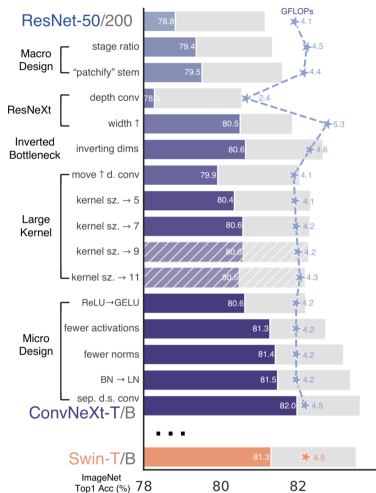


ReLU → GELU

$$GELU(x) = x\Phi(x)$$

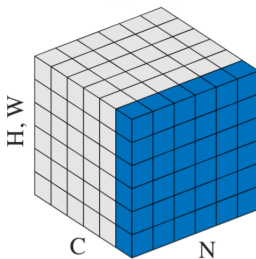


ConvNeXt

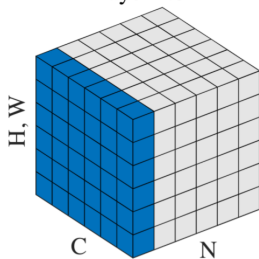


BatchNorm → LayerNorm

Batch Norm

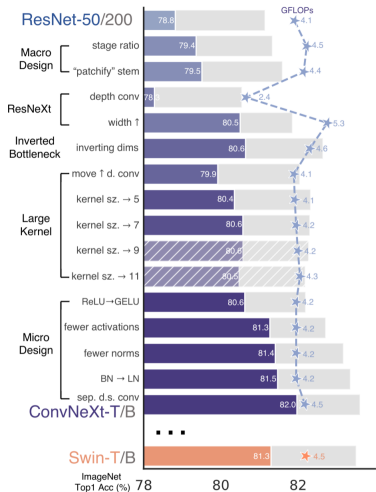


Layer Norm

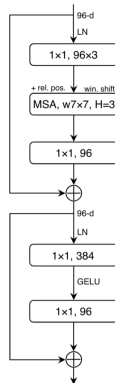


ConvNeXt

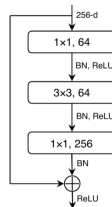
Block design



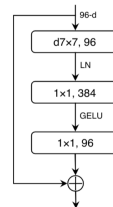
Swin Transformer Block



ResNet Block



ConvNeXt Block



ConvNeXt results

model	image size	#param.	FLOPs	throughput (image / s)	IN-1K top-1 acc.
ImageNet-1K trained models					
● RegNetY-16G [54]	224 ²	84M	16.0G	334.7	82.9
● EffNet-B7 [71]	600 ²	66M	37.0G	55.1	84.3
● EffNetV2-L [72]	480 ²	120M	53.0G	83.7	85.7
○ DeiT-S [73]	224 ²	22M	4.6G	978.5	79.8
○ DeiT-B [73]	224 ²	87M	17.6G	302.1	81.8
○ Swin-T	224 ²	28M	4.5G	757.9	81.3
● ConvNeXt-T	224 ²	29M	4.5G	774.7	82.1
○ Swin-S	224 ²	50M	8.7G	436.7	83.0
● ConvNeXt-S	224 ²	50M	8.7G	447.1	83.1
○ Swin-B	224 ²	88M	15.4G	286.6	83.5
● ConvNeXt-B	224 ²	89M	15.4G	292.1	83.8
○ Swin-B	384 ²	88M	47.1G	85.1	84.5
● ConvNeXt-B	384 ²	89M	45.0G	95.7	85.1
● ConvNeXt-L	224 ²	198M	34.4G	146.8	84.3
● ConvNeXt-L	384 ²	198M	101.0G	50.4	85.5

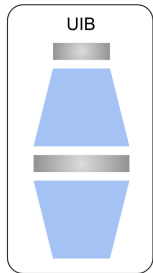
ImageNet-22K pre-trained models					
● R-101x3 [39]	384 ²	388M	204.6G	-	84.4
● R-152x4 [39]	480 ²	937M	840.5G	-	85.4
● EffNetV2-L [72]	480 ²	120M	53.0G	83.7	86.8
● EffNetV2-XL [72]	480 ²	208M	94.0G	56.5	87.3
○ ViT-B/16 (🐼) [67]	384 ²	87M	55.5G	93.1	85.4
○ ViT-L/16 (🐼) [67]	384 ²	305M	191.1G	28.5	86.8
● ConvNeXt-T	224 ²	29M	4.5G	774.7	82.9
● ConvNeXt-T	384 ²	29M	13.1G	282.8	84.1
● ConvNeXt-S	224 ²	50M	8.7G	447.1	84.6
● ConvNeXt-S	384 ²	50M	25.5G	163.5	85.8
○ Swin-B	224 ²	88M	15.4G	286.6	85.2
● ConvNeXt-B	224 ²	89M	15.4G	292.1	85.8
○ Swin-B	384 ²	88M	47.0G	85.1	86.4
● ConvNeXt-B	384 ²	89M	45.1G	95.7	86.8
○ Swin-L	224 ²	197M	34.5G	145.0	86.3
● ConvNeXt-L	224 ²	198M	34.4G	146.8	86.6
○ Swin-L	384 ²	197M	103.9G	46.0	87.3
● ConvNeXt-L	384 ²	198M	101.0G	50.4	87.5
● ConvNeXt-XL	224 ²	350M	60.9G	89.3	87.0
● ConvNeXt-XL	384 ²	350M	179.0G	30.2	87.8

Outline

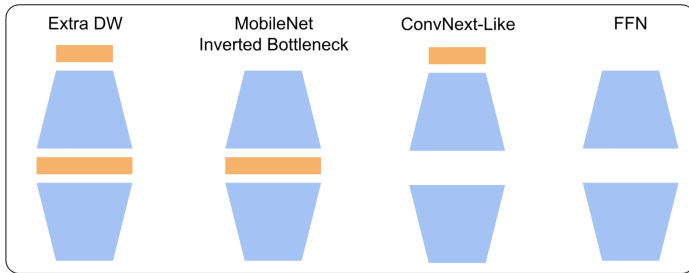
1. Vision Transformer
2. Swin Transformer
3. ConvNeXt
4. MobileNetV4

Universal Inverted Bottleneck

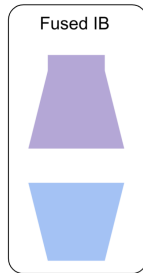
Universal IB block
w/ two optional DW



Possible instantiations of our UIB block



Alternative
Fused IB



Optional Depthwise



DepthWise



PointWise



Conv2D

Mobile MQA

$$\text{Mobile_MQA}(\mathbf{X}) = \text{Concat}(\text{attention}_1, \dots, \text{attention}_n) \mathbf{W}^O$$

$$\text{where } \text{attention}_j = \text{softmax} \left(\frac{(\mathbf{X} \mathbf{W}^{Q_j})(SR(\mathbf{X}) \mathbf{W}^K)^T}{\sqrt{d_k}} \right) (SR(\mathbf{X}) \mathbf{W}^V)$$

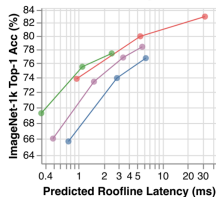
Roofline analysis

$$\text{ModelTime} = \sum_i \max(\text{MACTime}_i, \text{MemTime}_i)$$

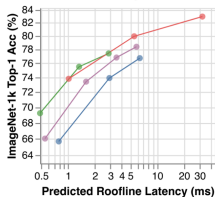
$$\text{MACTime}_i = \frac{\text{LayerMACs}_i}{\text{PeakMACs}}, \quad \text{MemTime}_i = \frac{\text{WeightBytes}_i + \text{ActivationBytes}_i}{\text{PeakMemBW}}$$

Roofline Latency vs Accuracy

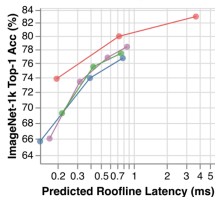
RP: 0.00 MACs/byte - MACs



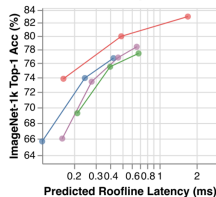
RP: 5.00 MACs/byte - Slow CPU



RP: 50.00 MACs/byte - Fast CPU



RP: 500.00 MACs/byte - Accelerator



Model Family

MobileNetV4

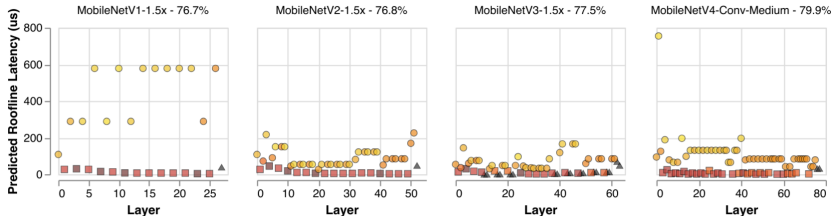
MobileNetV3

MobileNetV2

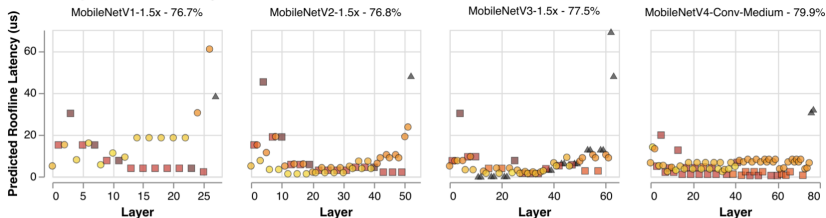
MobileNetV1

Roofline analysis

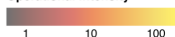
Roofline Ops (5.00 MACs/byte - Slow CPU)



Roofline Ops (500.00 MACs/byte - Accelerator)



Operational Intensity



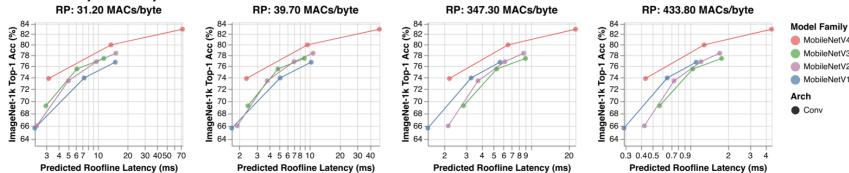
Op Type



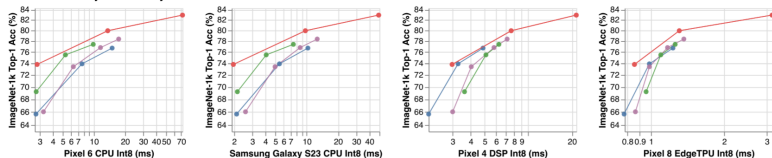
Roofline analysis

Execution Target	Ridge Point (MACs/B)	r_s -Roofline	r_s -MAC
Pixel 6 CPU (Int8)	31.2	0.973	0.962
Samsung Galaxy S23 CPU (Int8)	39.7	0.962	0.940
Pixel 4 DSP (Int8)	347.3	0.962	0.758
Pixel 8 EdgeTPU (Int8)	433.8	0.973	0.857

Roofline Latency vs Accuracy



Measured Latency vs Accuracy



Found architectures

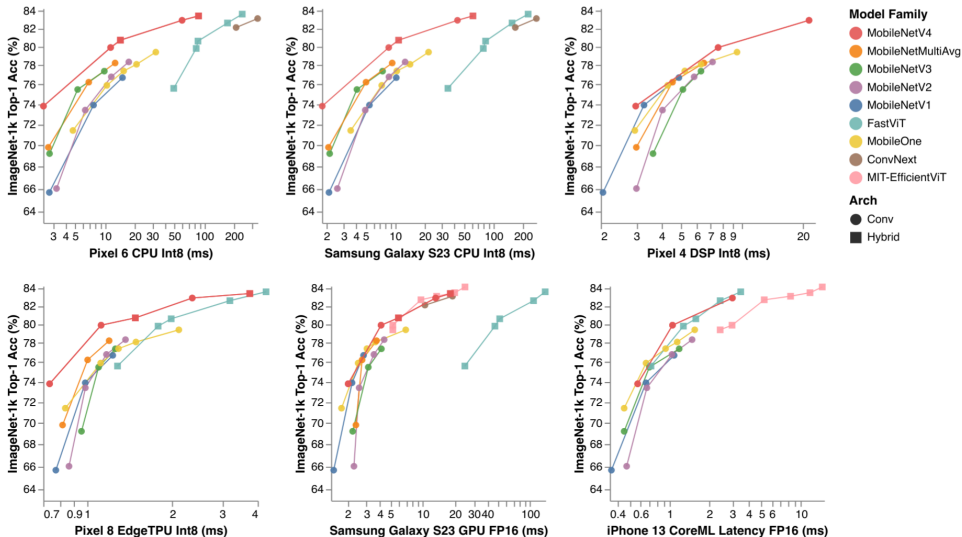
Table 12: Architecture specification of MNv4-Conv-M.

Input	Block	DW K_1	DW K_2	Expanded Dim	Output Dim	Stride
$256^2 \times 3$	Conv2D	-	3×3	-	32	2
$128^2 \times 32$	FusedIB	-	3×3	128	48	2
$64^2 \times 48$	ExtraDW	3×3	5×5	192	80	2
$32^2 \times 80$	ExtraDW	3×3	3×3	160	80	1
$32^2 \times 80$	ExtraDW	3×3	5×5	480	160	2
$16^2 \times 160$	ExtraDW	3×3	3×3	640	160	1
$16^2 \times 160$	ExtraDW	3×3	3×3	640	160	1
$16^2 \times 160$	ExtraDW	3×3	5×5	640	160	1
$16^2 \times 160$	ExtraDW	3×3	3×3	640	160	1
$16^2 \times 160$	ConvNext	3×3	-	640	160	1
$16^2 \times 160$	FFN	-	-	320	160	1
$16^2 \times 160$	ConvNext	3×3	-	640	160	1
$16^2 \times 160$	ExtraDW	5×5	5×5	960	256	2
$8^2 \times 256$	ExtraDW	5×5	5×5	1024	256	1
$8^2 \times 256$	ExtraDW	3×3	5×5	1024	256	1
$8^2 \times 256$	ExtraDW	3×3	5×5	1024	256	1
$8^2 \times 256$	FFN	-	-	1024	256	1
$8^2 \times 256$	ConvNext	3×3	-	1024	256	1
$8^2 \times 256$	ExtraDW	3×3	5×5	512	256	1
$8^2 \times 256$	ExtraDW	5×5	5×5	1024	256	1
$8^2 \times 256$	FFN	-	-	1024	256	1
$8^2 \times 256$	FFN	-	-	1024	256	1
$8^2 \times 256$	ConvNext	5×5	-	512	256	1
$8^2 \times 256$	Conv2D	-	1×1	-	960	1
$8^2 \times 960$	AvgPool	-	8×8	-	960	1
$1^2 \times 960$	Conv2D	-	1×1	-	1280	1
$1^2 \times 1280$	Conv2D	-	1×1	-	1000	1

Table 13: Architecture specification of MNv4-Hybrid-M.

Input	Block	DW K_1	DW K_2	Expanded Dim	Output Dim	Stride
$256^2 \times 3$	Conv2D	-	3×3	-	32	2
$128^2 \times 32$	FusedIB	-	3×3	128	48	2
$64^2 \times 48$	ExtraDW	3×3	5×5	192	80	2
$32^2 \times 80$	ExtraDW	3×3	3×3	160	80	1
$32^2 \times 80$	ExtraDW	3×3	5×5	480	160	2
$16^2 \times 160$	ExtraDW	3×3	3×3	640	160	1
$16^2 \times 160$	ExtraDW	3×3	3×3	640	160	1
$16^2 \times 160$	ExtraDW	3×3	5×5	640	160	1
$16^2 \times 160$	Mobile-MQA	-	-	-	160	1
$16^2 \times 160$	ExtraDW	3×3	3×3	640	160	1
$16^2 \times 160$	Mobile-MQA	-	-	-	160	1
$16^2 \times 160$	ConvNext	3×3	-	640	160	1
$16^2 \times 160$	Mobile-MQA	-	-	-	160	1
$16^2 \times 160$	FFN	-	-	640	160	1
$16^2 \times 160$	Mobile-MQA	-	-	-	160	1
$16^2 \times 160$	ConvNext	3×3	-	640	160	1
$16^2 \times 160$	ExtraDW	5×5	5×5	960	256	2
$8^2 \times 256$	ExtraDW	5×5	5×5	1024	256	1
$8^2 \times 256$	ExtraDW	3×3	5×5	1024	256	1
$8^2 \times 256$	ExtraDW	3×3	5×5	1024	256	1
$8^2 \times 256$	FFN	-	-	1024	256	1
$8^2 \times 256$	ConvNext	3×3	-	1024	256	1
$8^2 \times 256$	ExtraDW	3×3	5×5	512	256	1
$8^2 \times 256$	Mobile-MQA	-	-	-	256	1
$8^2 \times 256$	ExtraDW	5×5	5×5	1024	256	1
$8^2 \times 256$	Mobile-MQA	-	-	-	256	1
$8^2 \times 256$	FFN	-	-	1024	256	1
$8^2 \times 256$	Mobile-MQA	-	-	-	256	1
$8^2 \times 256$	FFN	-	-	1024	256	1
$8^2 \times 256$	Mobile-MQA	-	-	-	256	1
$8^2 \times 256$	ConvNext	5×5	-	1024	256	1
$8^2 \times 256$	Conv2D	-	1×1	-	960	1
$8^2 \times 960$	AvgPool	-	8×8	-	960	1
$1^2 \times 960$	Conv2D	-	1×1	-	1280	1
$1^2 \times 1280$	Conv2D	-	1×1	-	1000	1

Evaluation results



Conclusion

We reviewed three key modern backbones and a mobile network:

1. Vision Transformer (ViT) applies ideas from NLP to images. Key element is attention — mechanism for gathering information across whole image
2. Swin Transformer reintroduces convnet priors to transformers using shifted window attention
3. ConvNeXt modernizes ResNets into a transformer-like fully convolutional architecture
4. MobileNetV4 combines ideas from CNNs and transformers and uses NAS and Roofline Analysis to find fast architectures that works well across various devices