Transformers and large-kernel CNNs

Vlad Shakhuro

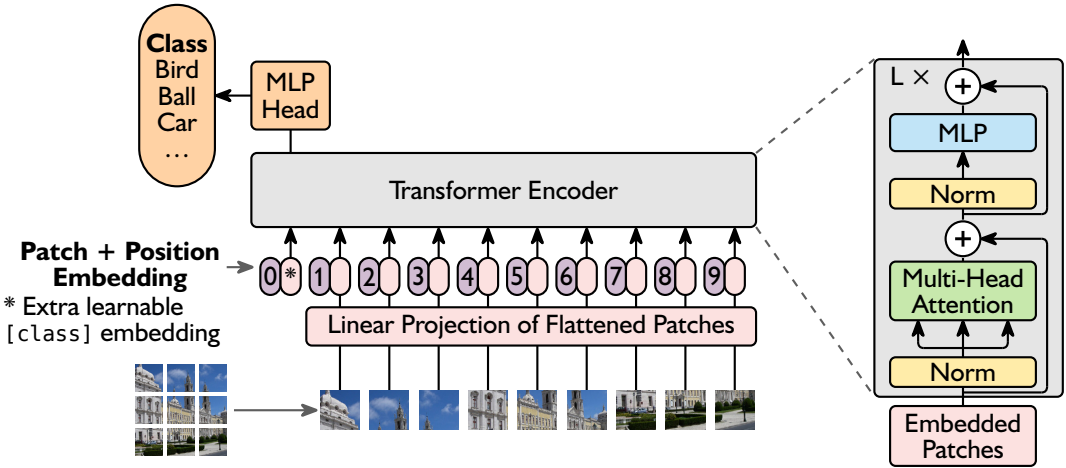


22 October 2026

Outline

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

Vision Transformer



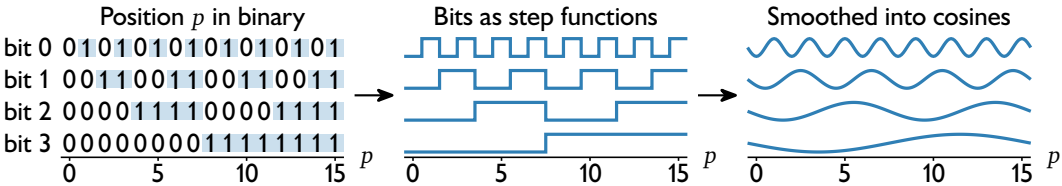
Dosovitskiy et al. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. ICLR 2021

Position embeddings

Attention ignores token order: add a position embedding to each token

Learned (ViT): $E_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$: N patches and class token, width D

Sinusoidal (Transformer): position in binary, bits as steps, steps smoothed



$$\text{PE}(p, 2i) = \sin\left(p/10000^{2i/D}\right), \quad \text{PE}(p, 2i+1) = \cos\left(p/10000^{2i/D}\right), \quad i < D/2$$

Dosovitskiy et al. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. ICLR 2021
Vaswani et al. Attention Is All You Need. NeurIPS 2017

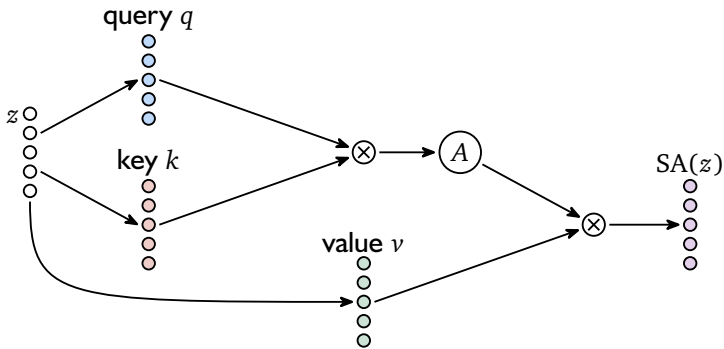
Multi-Head Self-Attention

$$[q, k, v] = zU_{qkv}, \qquad z \in \mathbb{R}^{N\times D},\ U_{qkv} \in \mathbb{R}^{D\times 3D_h}$$

$$A = \text{softmax}(qk^T/\sqrt{D_h}), \qquad A \in \mathbb{R}^{N\times N}$$

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

$$\text{MSA}(z) = [\text{SA}_1(z); \text{SA}_2(z); \ldots; \text{SA}_h(z)]U_{\text{msa}}, \qquad U_{\text{msa}} \in \mathbb{R}^{h\cdot D_h\times D}$$



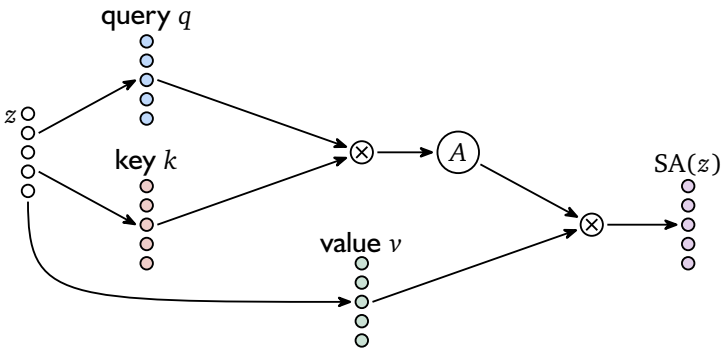
Multi-Head Self-Attention

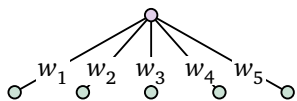
$$[q, k, v] = zU_{qkv}, \qquad z \in \mathbb{R}^{N\times D},\ U_{qkv} \in \mathbb{R}^{D\times 3D_h}$$

$$A = \text{softmax}(qk^T/\sqrt{D_h}), \qquad A \in \mathbb{R}^{N\times N}$$

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

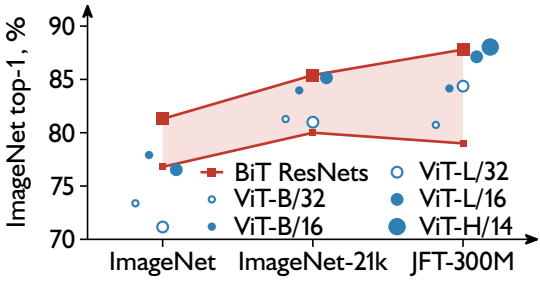
$$\text{MSA}(z) = [\text{SA}_1(z); \text{SA}_2(z); \ldots; \text{SA}_h(z)]U_{\text{msa}}, \qquad U_{\text{msa}} \in \mathbb{R}^{h\cdot D_h\times D}$$

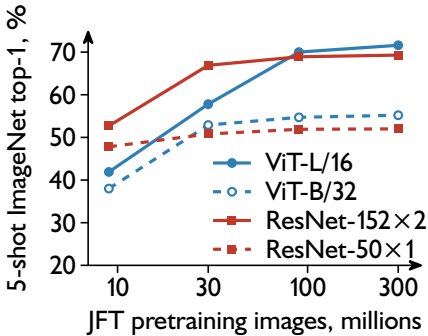




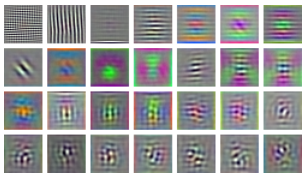
Vision Transformer

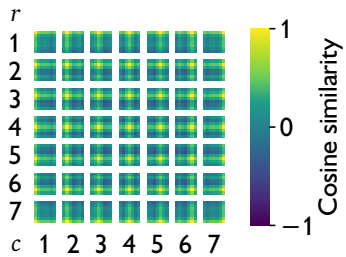
Model	Layers L	Hidden size D	MLP size	Heads h	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

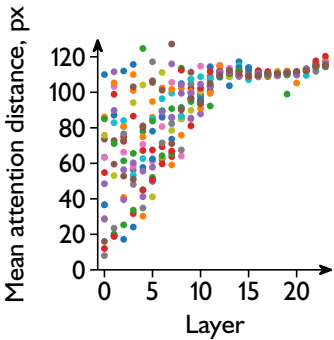




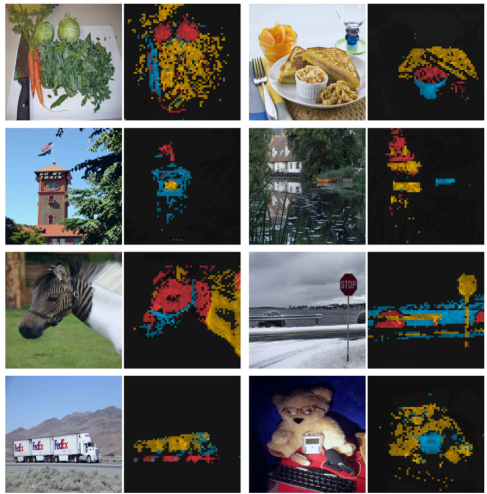
Learned embeddings and attention distance







Attention maps of heads



Caron et al. Emerging Properties in Self-Supervised Vision Transformers. ICCV 2021

Data-Efficient Transformer

	ViT-B	DeiT-B
Epochs	300	300
Batch size	4096	1024
Optimizer	AdamW	AdamW
Learning rate	0.003	$0.0005 \cdot \frac{\text{batch size}}{512}$
Learning rate decay	cosine	cosine
Weight decay	0.3	0.05
Warmup epochs	3.4	5
Label smoothing ε	–	0.1
Dropout	0.1	–
Stochastic depth	–	0.1
Repeated aug.	–	yes
Gradient clipping	yes	–
RandAugment	–	9/0.5
Mixup prob.	–	0.8
CutMix prob.	–	1
Erasing prob.	–	0.25

Mixup



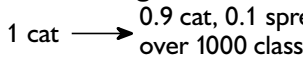
CutMix



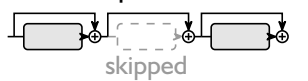
Repeated augmentation



Label smoothing

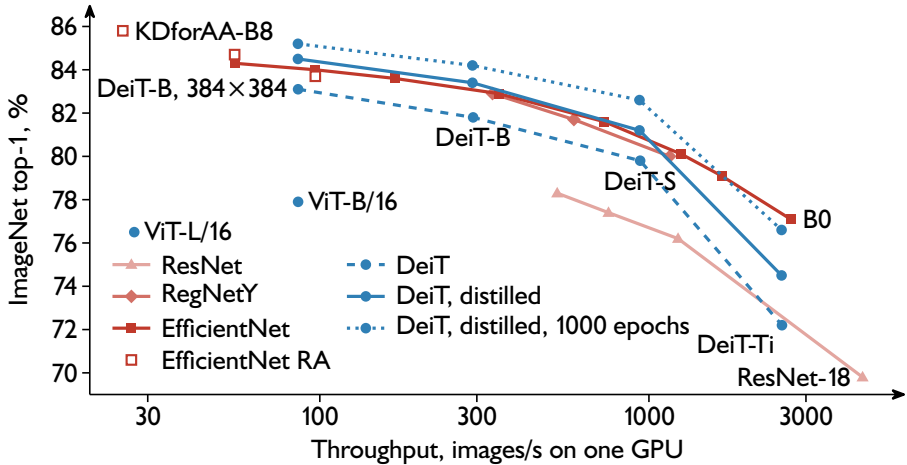


Stochastic depth



Touvron et al. Training data-efficient image transformers & distillation through attention. ICML 2021

DeiT results

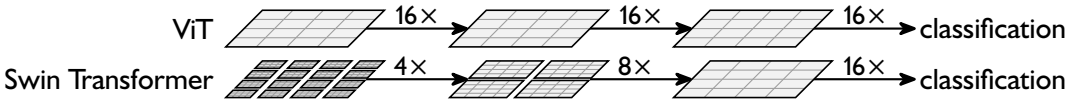


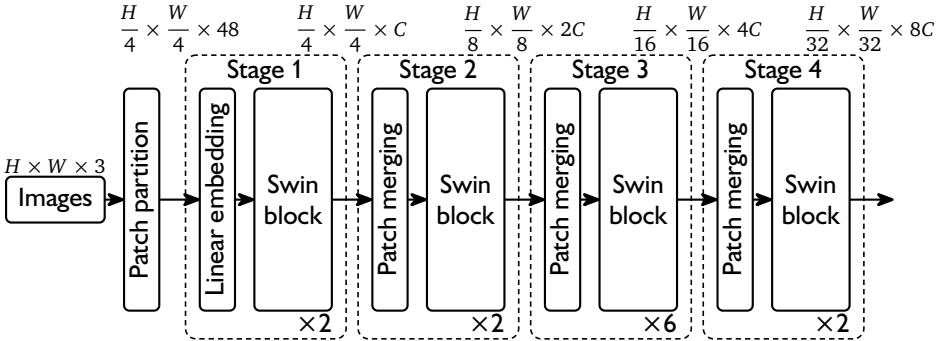
Touvron et al. Training data-efficient image transformers & distillation through attention. ICML 2021

Outline

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

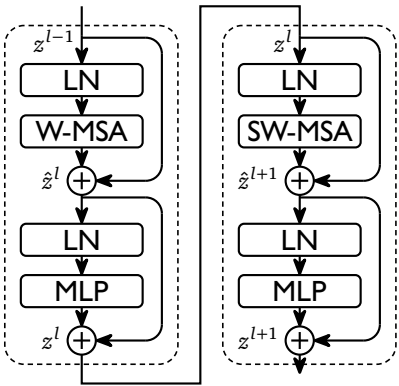
Swin Transformer

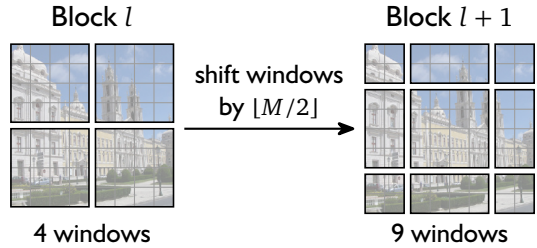




Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. ICCV 2021

Shifted window attention





W-MSA, SW-MSA: MSA in regular and shifted windows of $M \times M$ patches

$$\mathrm{SA}(z) = \mathrm{softmax}(qk^T/\sqrt{D_h} + B)v$$

$B \in \mathbb{R}^{M^2 \times M^2}$: learned relative position bias

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
		[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 128, head 4] × 2	[win. sz. 7×7, dim 192, head 6] × 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
		[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 256, head 8] × 2	[win. sz. 7×7, dim 384, head 12] × 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
		[win. sz. 7×7, dim 384, head 12] × 6	[win. sz. 7×7, dim 384, head 12] × 18	[win. sz. 7×7, dim 512, head 16] × 18	[win. sz. 7×7, dim 768, head 24] × 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
		[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 1024, head 32] × 2	[win. sz. 7×7, dim 1536, head 48] × 2

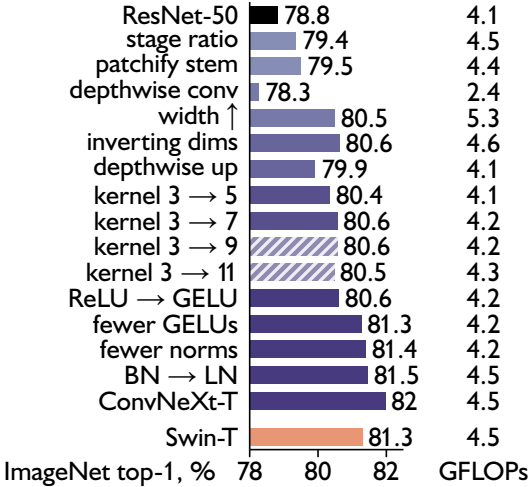
Swin results on ImageNet

Model	Input	Params, M	FLOPs, G	Images/s	Top-1, %
RegNetY-4G	224×224	21	4.0	1156.7	80.0
RegNetY-8G	224×224	39	8.0	591.6	81.7
RegNetY-16G	224×224	84	16.0	334.7	82.9
EfficientNet-B3	300×300	12	1.8	732.1	81.6
EfficientNet-B4	380×380	19	4.2	349.4	82.9
EfficientNet-B5	456×456	30	9.9	169.1	83.6
EfficientNet-B6	528×528	43	19.0	96.9	84.0
EfficientNet-B7	600×600	66	37.0	55.1	84.3
DeiT-S	224×224	22	4.6	940.4	79.8
DeiT-B	224×224	86	17.5	292.3	81.8
Swin-T	224×224	29	4.5	755.2	81.3
Swin-S	224×224	50	8.7	436.9	83.0
Swin-B	224×224	88	15.4	278.1	83.5

Outline

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

ConvNeXt



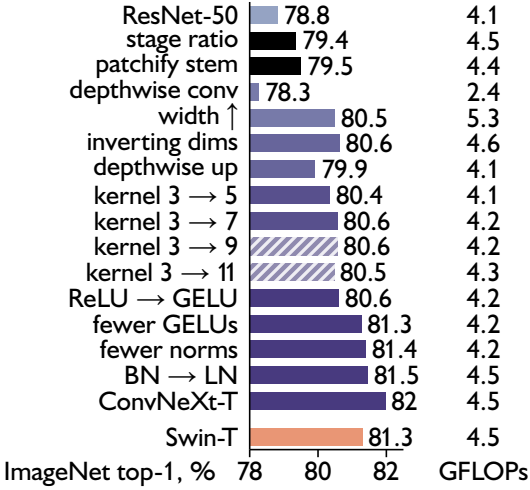
Baseline ResNet-50 with modern training:

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

76.1% → 78.8% on ImageNet

Liu et al. A ConvNet for the 2020s. CVPR 2022

ConvNeXt



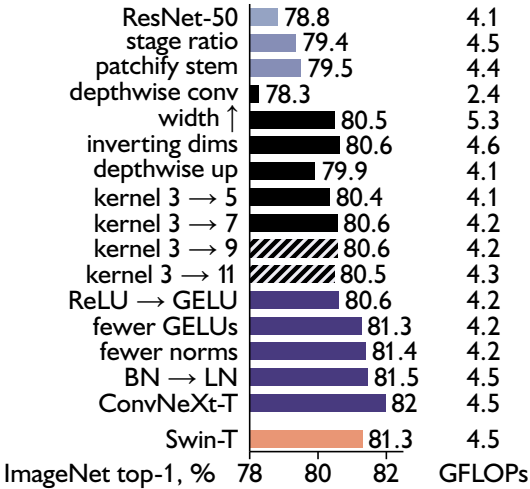
Macro design

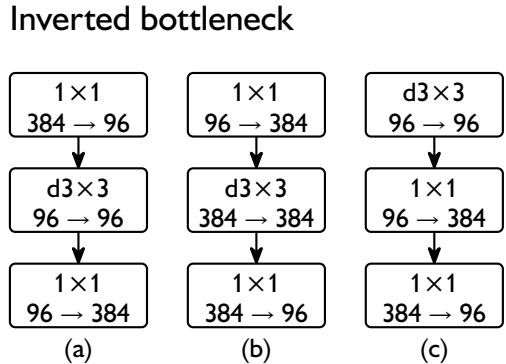
Change #blocks
(3,4,6,3) → (3,3,9,3)

Change "patchify" stem
7×7 conv/2, pool/2 →
4×4 conv/4

Liu et al. A ConvNet for the 2020s. CVPR 2022

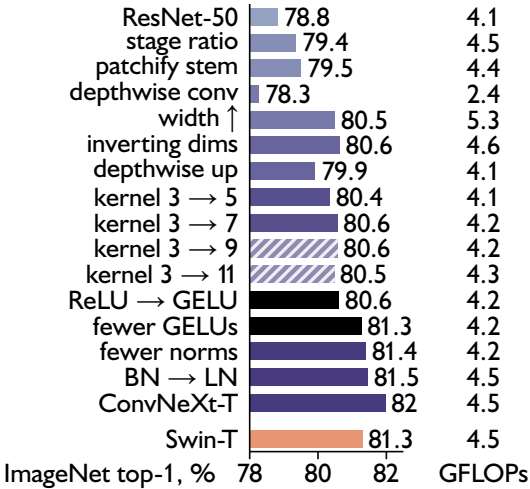
ConvNeXt





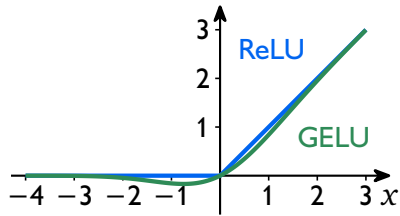
Liu et al. A ConvNet for the 2020s. CVPR 2022

ConvNeXt



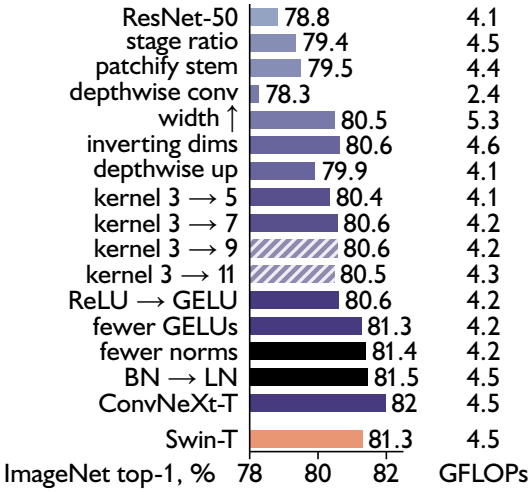
ReLU → GELU

$\mathrm{GELU}(x) = x\Phi(x)$

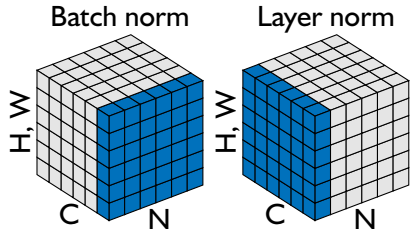


Liu et al. A ConvNet for the 2020s. CVPR 2022
Hendrycks, Gimpel. Gaussian Error Linear Units (GELUs). arXiv 2016

ConvNeXt

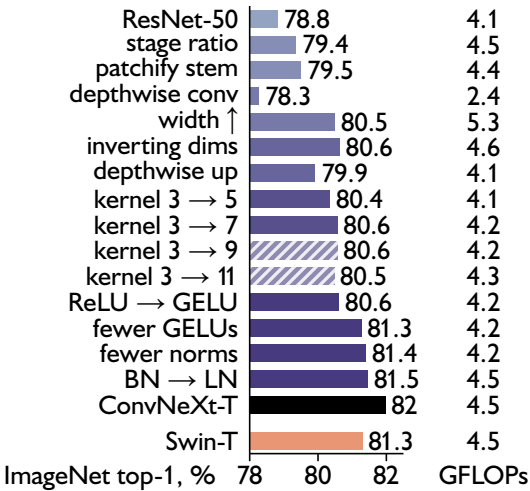


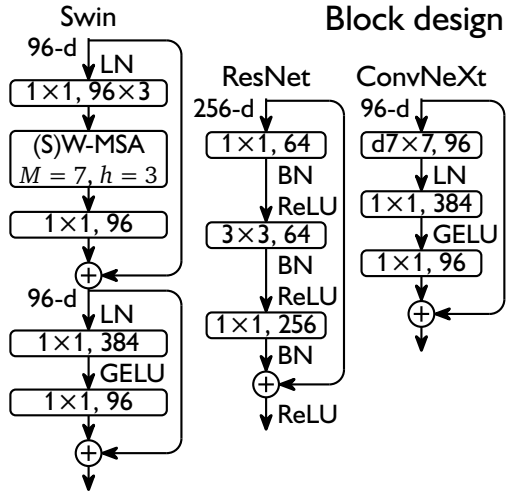
BatchNorm → LayerNorm



Liu et al. A ConvNet for the 2020s. CVPR 2022
Wu, He. Group Normalization. ECCV 2018

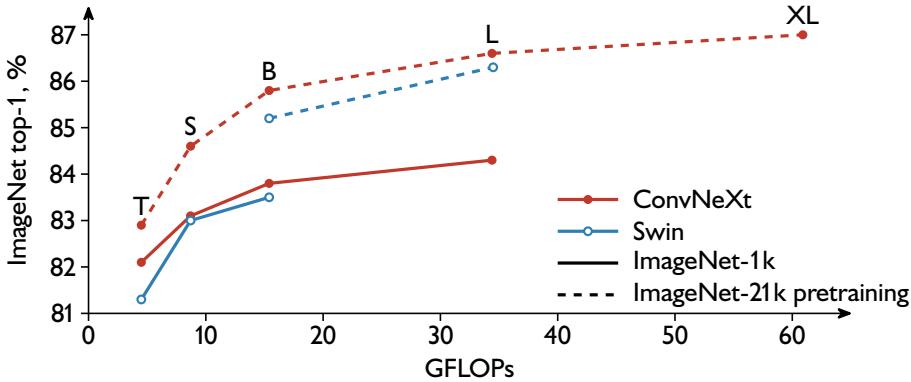
ConvNeXt





Liu et al. A ConvNet for the 2020s. CVPR 2022

ConvNeXt results

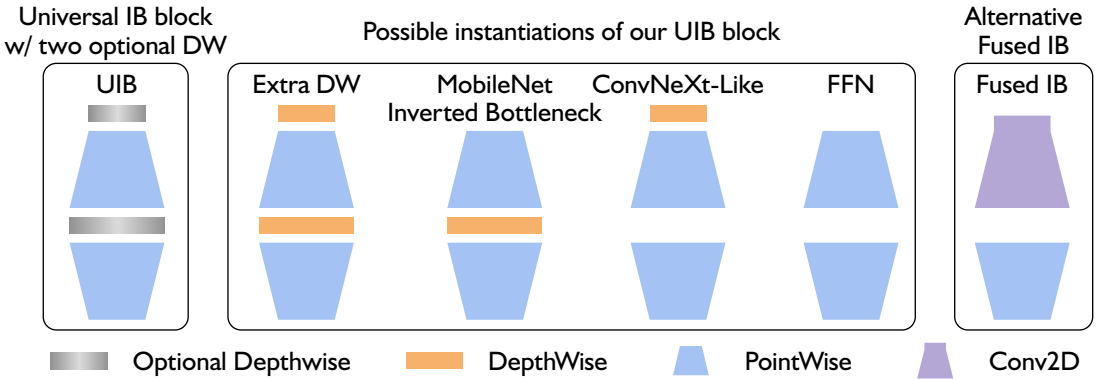


At equal FLOPs ConvNeXt is as accurate as Swin or better

Outline

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

Universal Inverted Bottleneck



Qin et al. MobileNetV4 — Universal Models for the Mobile Ecosystem. ECCV 2024

Mobile MQA

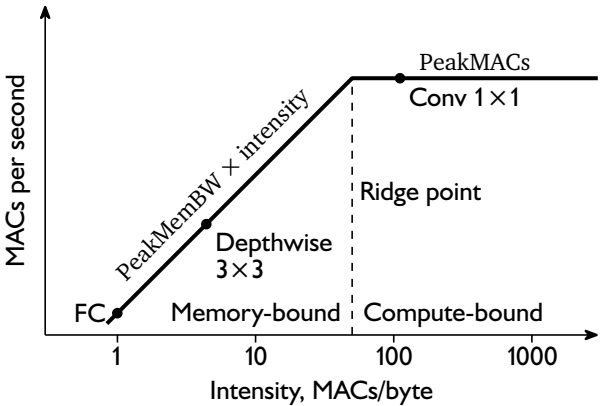
Multi-query attention (MQA): n heads with own queries share $\mathbf{W}^K$, $\mathbf{W}^V$
Mobile MQA downsamples $\mathbf{X}$ with SR: 3×3 depthwise conv, stride 2

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

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

				EdgeTPU, ms		GPU, ms
MNv4-Conv-L	Top-1, %	MACs, G	Params, M	Pixel 7	Pixel 8	Galaxy S23
base	84.9	6.0	30.9	4.3	2.4	13.2
+3 MSA blocks	85.3	6.7	36.0	9.7	2.8	16.5
+3 MQA blocks	85.2	6.5	34.7	5.2	2.6	15.1
MQA vs MSA blocks, %	(−0.03)	**(−28.6)**	**(−25.5)**	**(−84.2)**	**(−39.0)**	**(−41.1)**

Roofline model



Ridge point

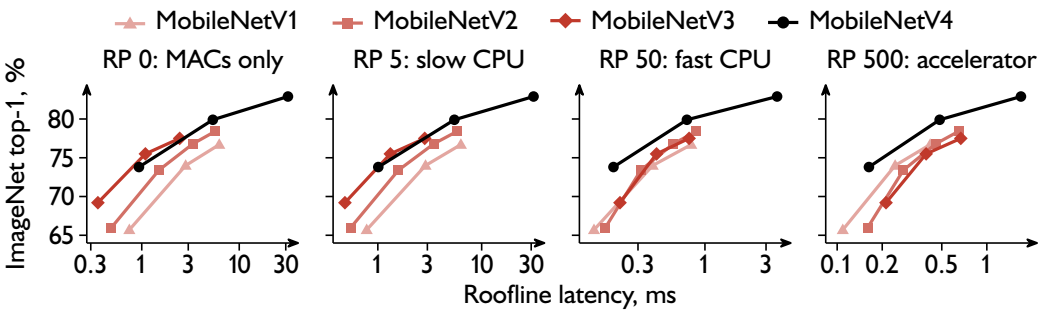
$$\mathrm{RP} = \frac{\mathrm{PeakMACs}}{\mathrm{PeakMemBW}}$$

Williams et al. Roofline: An Insightful Visual Performance Model for Multicore Architectures. CACM 2009

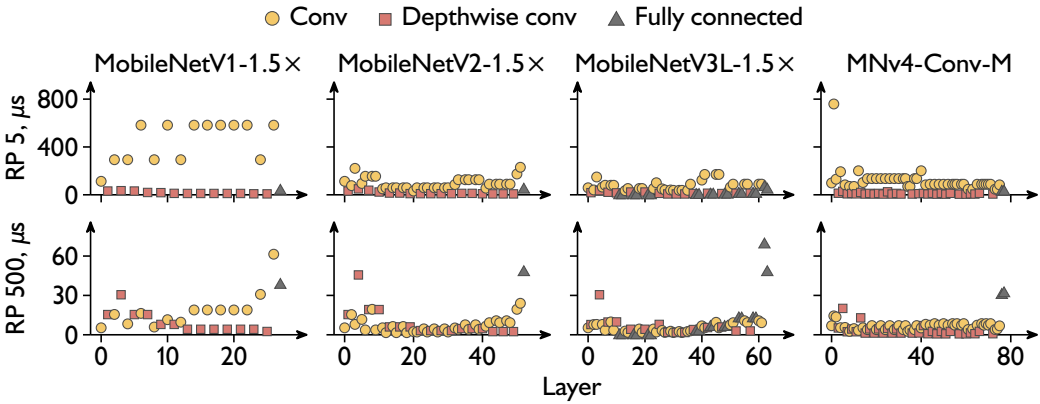
Roofline latency

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

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

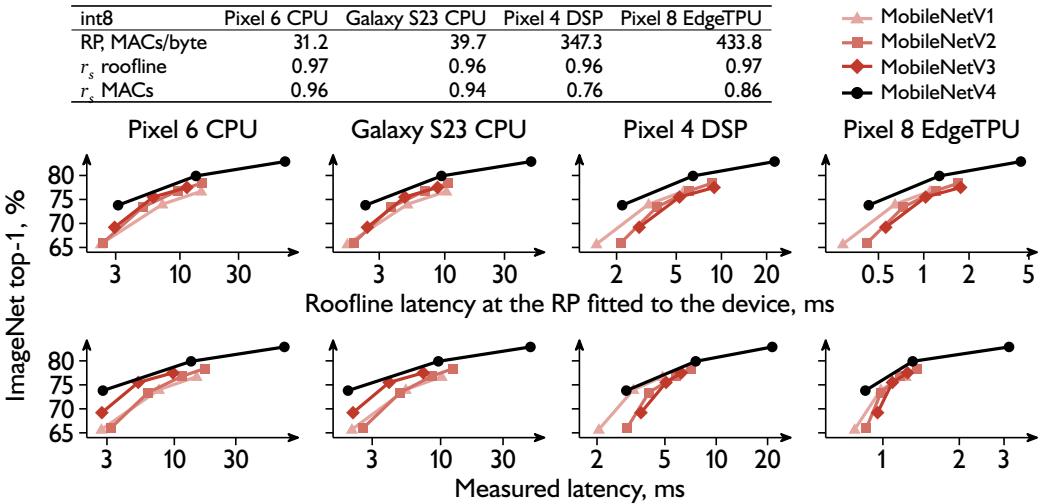


Roofline latency per layer



Roofline vs measured latency

int8	Pixel 6 CPU	Galaxy S23 CPU	Pixel 4 DSP	Pixel 8 EdgeTPU
RP, MACs/byte	31.2	39.7	347.3	433.8
r_s roofline	0.97	0.96	0.96	0.97
r_s MACs	0.96	0.94	0.76	0.86



Found architectures

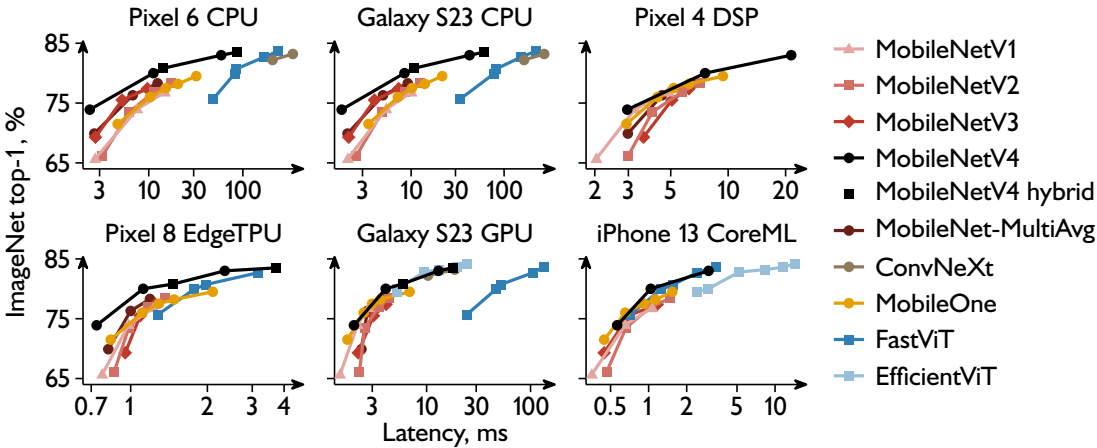
Table 12: Architecture specification of MNv4-Conv-M

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 work well across various devices