Faster and Lighter Vision Transformer

- 2022 동계 세미나 -

심 재 헌

Vision & Display Systems Lab.

Dept. of Electronic Engineering, Sogang University

Outline

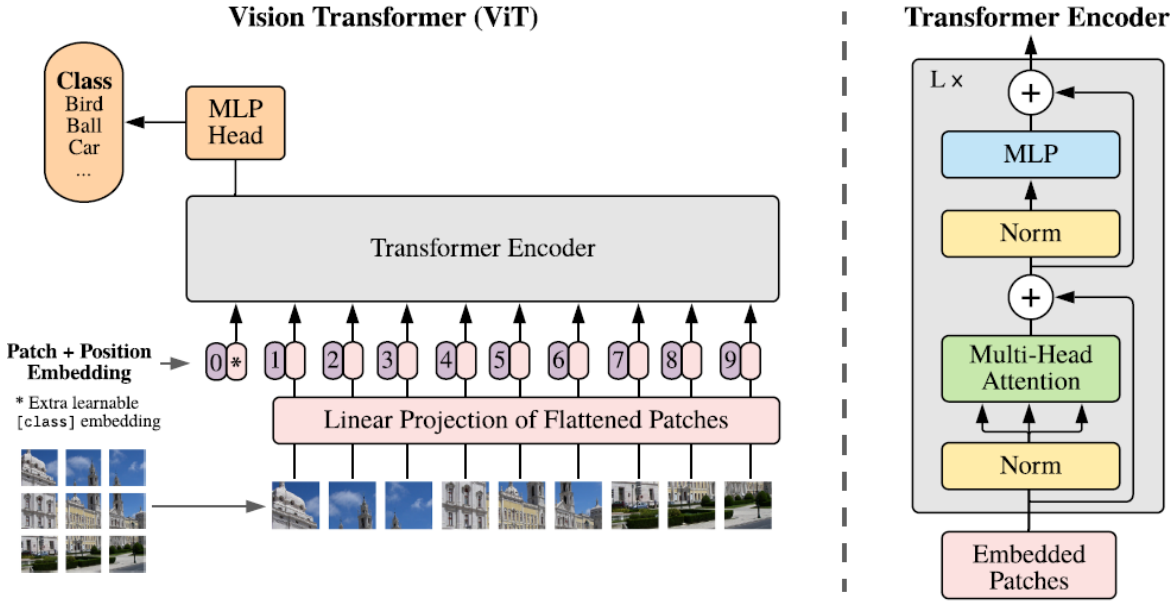
- Vision Transformer
- LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference
- MobileViT: Light-weight, General-Purpose, and Mobile-friendly Vision Transformer
- Conclusion

Vision Transformer

- An Image is Worth 16x16 Words
  - Transformer for Vision Task
    - Originally designed for NLP sequential data
  - Convolution not necessary
    - No dependency on CNNs

Procedure

1. Split an image into patches (16x16)
2. Linear Embedding
3. Add positional embeddings
4. Feed sequence into Transformer



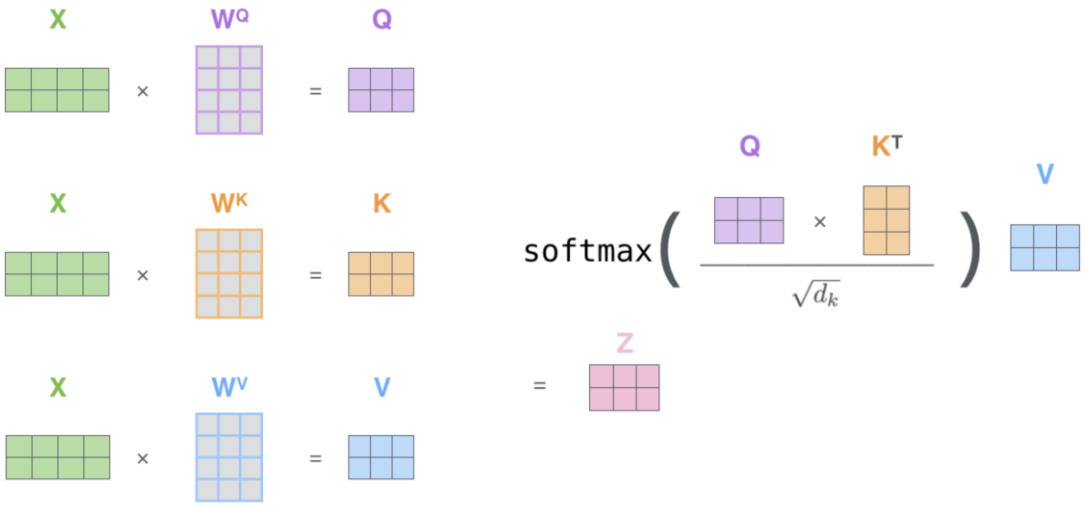
Vision Transformer

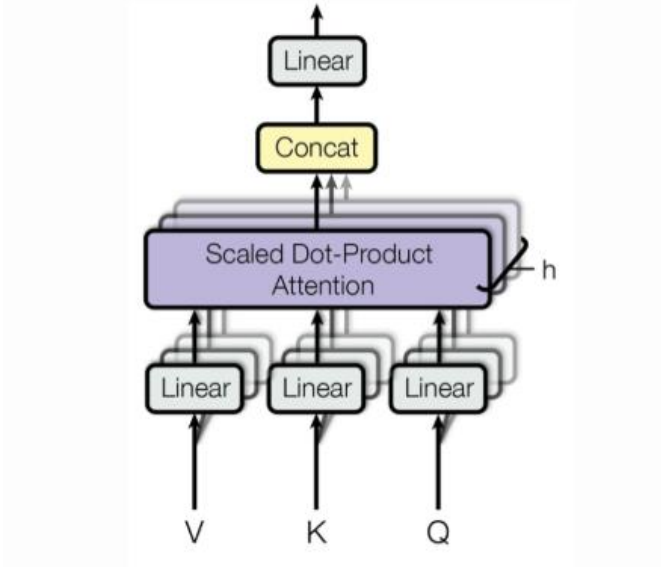
- Self-attention
  - Scaled Dot-Product
  - Multiple Scaled Dot-Product Attention → Multi-head attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

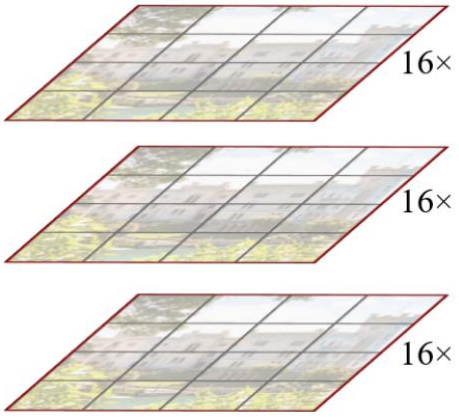
Q: query, K: key, V: value, X: input
W^Q, W^K, W^V: weight matrix



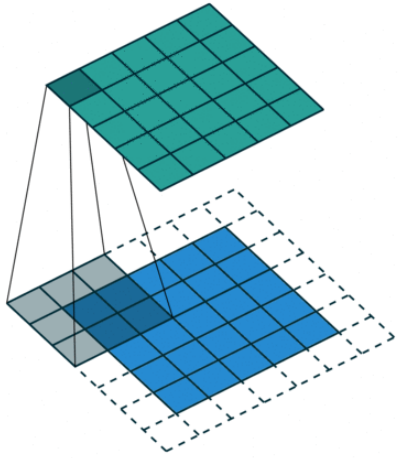


Vision Transformer

- Problems
  - No overlapping regions
    - No locality
    - Not translationally equivariant (No inductive bias)
      - Large dataset & thorough augmentation needed
  - Unified resolution
    - Equal resolution features for all layers
    - Dense prediction (segmentation, detection…) difficult
  - Computationally inefficient
    - Large model size
    - Slow inference speed
    - Large FLOPs (floating-point operations)



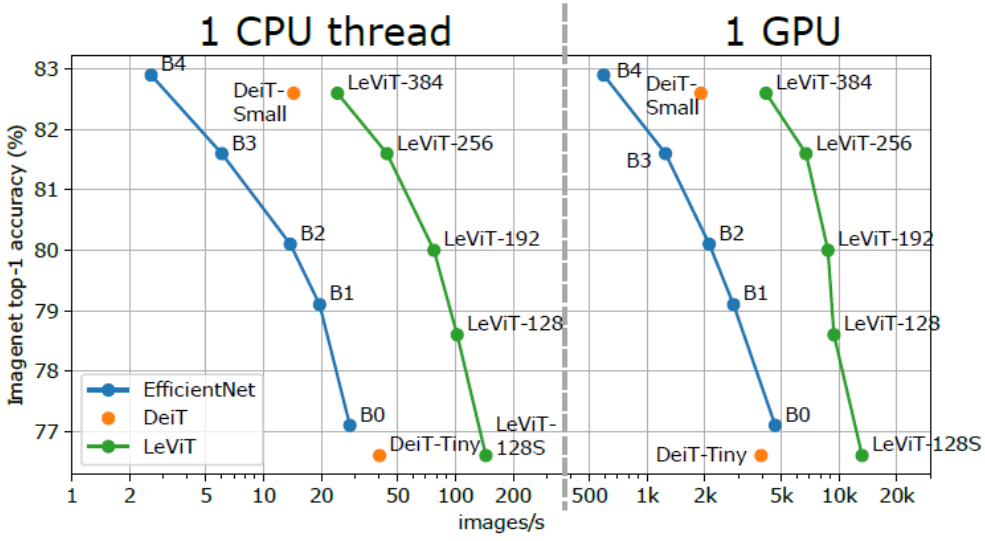
Vision Transformer



Convolution Neural Network

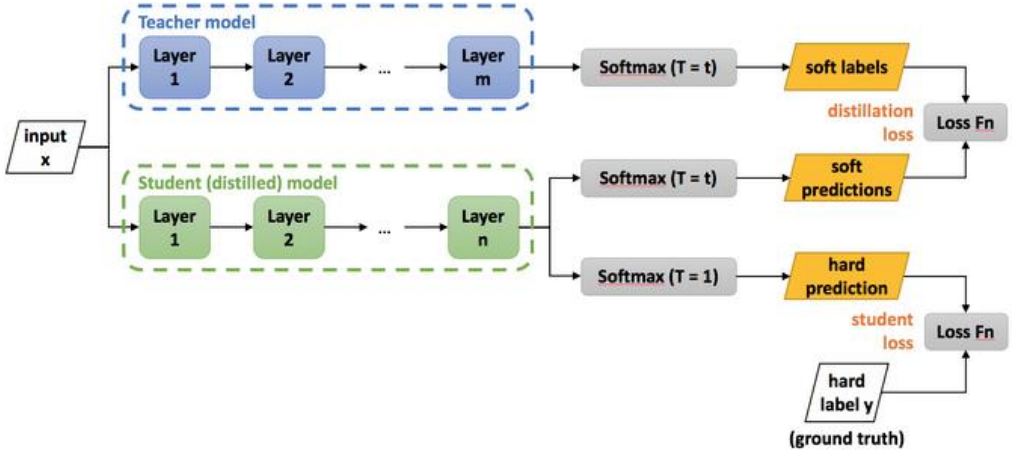
LeViT: Introduction

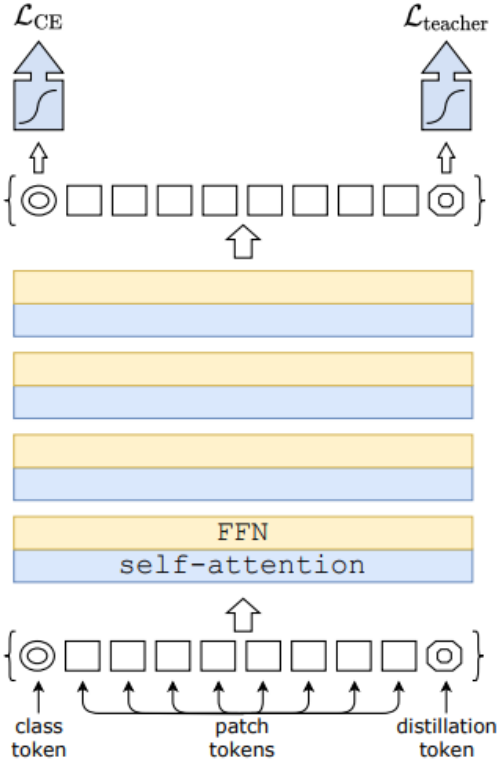
- LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference
  - Faster Vision transformer
    - Recent works focus only on decreasing model size & theoretical computation (FLOPs)
  - Re-introduce convolutional components into Vision Transformer
    - Trade off between attention computation and convolutions



LeViT: Background

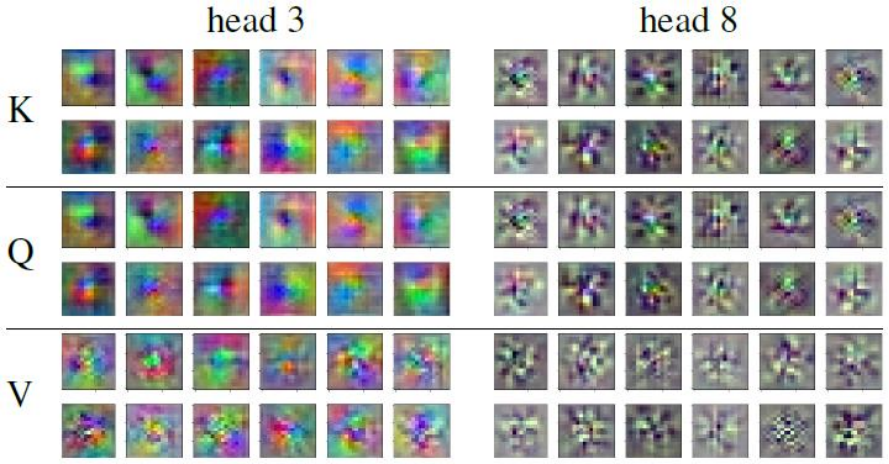
- DeiT: Training data-efficient image transformers & distillation through attention
  - Using only ImageNet-1k to train ViT
    - Original ViT trained on JFT-300M
    - Use knowledge distillation for ViT
      - Distillation Token instead of class token
      - RegNetV-16FG as teacher





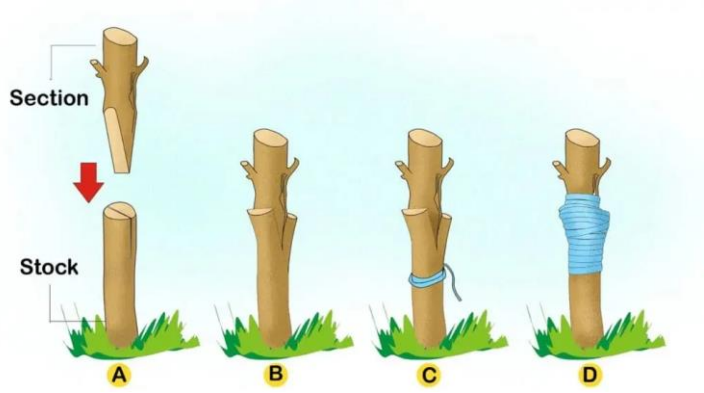
LeViT: Method

- Convolutions in the ViT architecture
  - Key, query, values formed by multiplying learned weights
    - Similar to convolutional functions to the input
    - Attention weights resemble that of convolutional layers
  - Spatial smoothness
    - Overlapping properties of convolutions
      - Smoothness property applied to ViT through data augmentation



LeViT: Method

- Grafting
  - Combination of ResNet & ViT
    - Original paper claims that it is not very effective
      - ⁘ Only ViT is better
    - Extensive experiments necessary

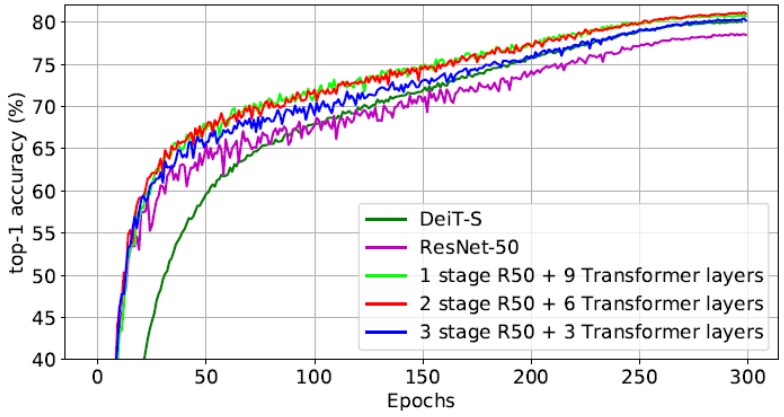


name	Epochs	ImageNet	ImageNet ReaL	CIFAR-10	CIFAR-100	Pets	Flowers	exaFLOPs
ViT-B/32	7	80.73	86.27	98.61	90.49	93.40	99.27	55
ViT-B/16	7	84.15	88.85	99.00	91.87	95.80	99.56	224
ViT-L/32	7	84.37	88.28	99.19	92.52	95.83	99.45	196
ViT-L/16	7	86.30	89.43	99.38	93.46	96.81	99.66	783
ViT-L/16	14	87.12	89.99	99.38	94.04	97.11	99.56	1567
ViT-H/14	14	88.08	90.36	99.50	94.71	97.11	99.71	4262
R50x1+ViT-B/32	7	84.90	89.15	99.01	92.24	95.75	99.46	106
R50x1+ViT-B/16	7	85.58	89.65	99.14	92.63	96.65	99.40	274
R50x1+ViT-L/32	7	85.68	89.04	99.24	92.93	96.97	99.43	246
R50x1+ViT-L/16	7	86.60	89.72	99.18	93.64	97.03	99.40	859
R50x1+ViT-L/16	14	87.12	89.76	99.31	93.89	97.36	99.11	1668

LeViT: Method

- Grafting
  - Combination of ResNet-50 and DeiT-Small
    - Mixture of ResNet and DeiT block under similar computational budgets
  - Results
    - Mixing two stages of ResNet with DeiT showed best results
    - Combined model showed fast convergence

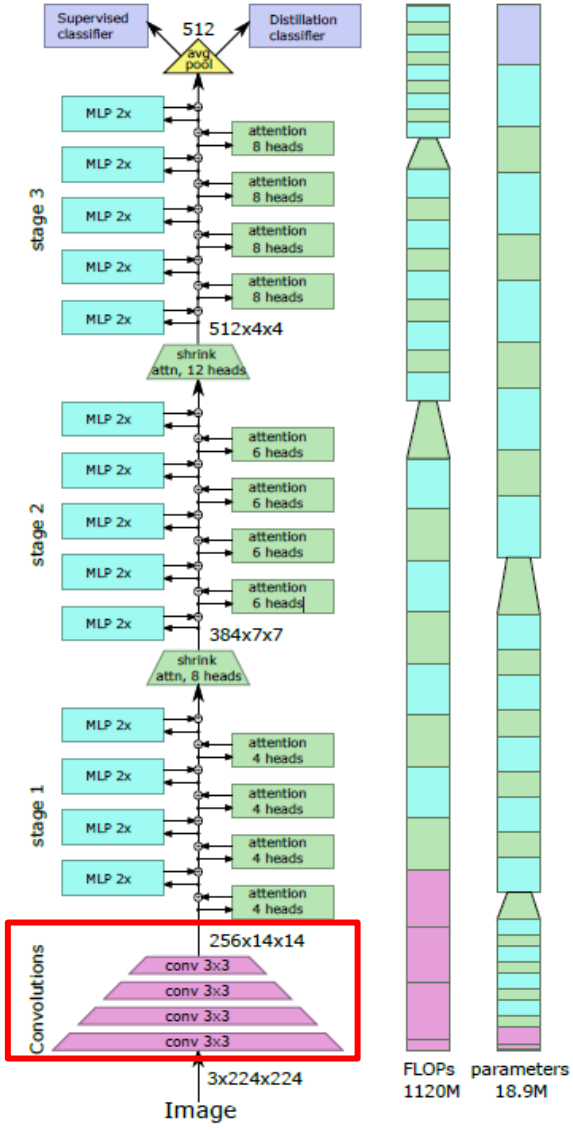
#ResNet stages	#DeiT-S layers	nb. of Params	FLOPs (M)		Speed im/s	IMNET top-1
			conv	transformer		
0	12	22.0M	57	4519	966	79.9
1	9	17.1M	820	3389	995	80.6
2	6	13.1M	1876	2260	1048	80.9
3	3	15.1M	3385	1130	1054	80.1
4	0	25.5M	4119	0	1254	78.4



LeViT: Method

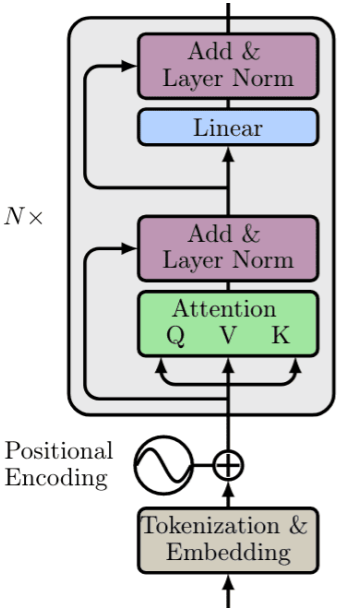
- Model Design
  - VGG style Convolution blocks added to DeiT
    - Less computation compared to ResNet
      - ResNet : 1042M FLOPs
      - VGG : 1084M FLOPs
    - Fast resolution shrink
  - No classification token
    - BCHW tensor used → positional information preserved
  - Attention bias instead of positional embedding
    - Positional embedding important to ViT performance
    - Include positional embedding to every attention block in the form of bias

$$A^h_{(x,y),(x',y')} = Q_{(x,y),:} \bullet K_{(x',y'),:} + B^h_{|x-x'|,|y-y'|}$$

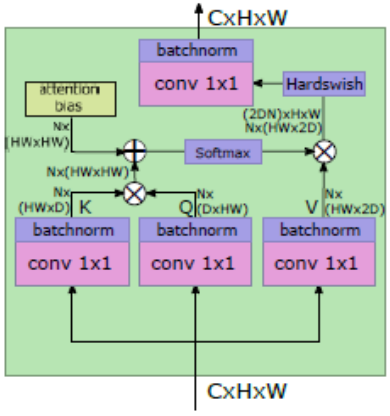


LeViT: Method

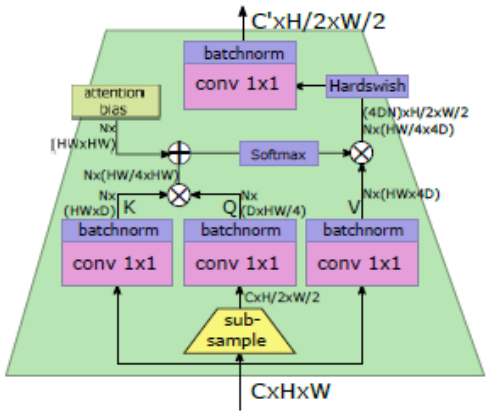
- Model Design
  - Reduction block
    - Following ResNet structure
    - Pyramid structure



  - Convolution instead of MLP
    - No classification token
      - Average pooling used instead
    - Linear embedding → 1x1 conv
      - Layer Norm → BatchNorm
    - Feed forward MLP also to conv



Normal block



Reduction block

LeViT: Experiments

- Experiment Settings
  - DeiT as Baseline
  - Test on various hardware platforms
    - 16GB NVIDIA Volta GPU
    - Intel Xeon 6138 CPU
    - ARM Graviton2 CPU

Model	LeViT-128S ($D=16, p=0$)	LeViT-128 ($D=16, p=0$)	LeViT-192 ($D=32, p=0$)	LeViT-256 ($D=32, p=0$)	LeViT-384 ($D=32, p=0.1$)
Stage 1: 14×14	$2\times \begin{bmatrix} C=128 \\ N=4 \end{bmatrix}$	$4\times \begin{bmatrix} C=128 \\ N=4 \end{bmatrix}$	$4\times \begin{bmatrix} C=192 \\ N=3 \end{bmatrix}$	$4\times \begin{bmatrix} C=256 \\ N=4 \end{bmatrix}$	$4\times \begin{bmatrix} C=384 \\ N=6 \end{bmatrix}$
Subsample	$[\ N=8\ ]$	$[\ N=8\ ]$	$[\ N=6\ ]$	$[\ N=8\ ]$	$[N=12]$
Stage 2: 7×7	$3\times \begin{bmatrix} C=256 \\ N=6 \end{bmatrix}$	$4\times \begin{bmatrix} C=256 \\ N=8 \end{bmatrix}$	$4\times \begin{bmatrix} C=288 \\ N=5 \end{bmatrix}$	$4\times \begin{bmatrix} C=384 \\ N=6 \end{bmatrix}$	$4\times \begin{bmatrix} C=512 \\ N=9 \end{bmatrix}$
Subsample	$[N=16]$	$[N=16]$	$[N=9]$	$[N=12]$	$[N=18]$
Stage 3: 4×4	$4\times \begin{bmatrix} C=384 \\ N=8 \end{bmatrix}$	$4\times \begin{bmatrix} C=384 \\ N=12 \end{bmatrix}$	$4\times \begin{bmatrix} C=384 \\ N=6 \end{bmatrix}$	$4\times \begin{bmatrix} C=512 \\ N=8 \end{bmatrix}$	$4\times \begin{bmatrix} C=768 \\ N=12 \end{bmatrix}$

Model structures. First four convolution layers are applied equally to all models.

LeViT: Experiments

- Results on ImageNet-1k

Architecture	# params (M)	FLOPs (M)	inference speed top-1 %	GPU im/s	Intel im/s	ARM im/s	ImageNet -Real %	-V2. %
LeViT-128S (**ours**)	7.8	305	76.6	12880	131.1	39.1	83.1	64.3
EfficientNet B0	5.3	390	77.1	4754	30.1	3.5	83.5	64.3
LeViT-128 (**ours**)	9.2	406	78.6	9266	94.0	30.8	84.7	66.6
LeViT-192 (**ours**)	10.9	658	80.0	8601	65.0	24.2	85.7	68.0
EfficientNet B1	7.8	700	79.1	2882	20.0	2.3	84.9	66.9
EfficientNet B2	9.2	1000	80.1	2149	13.1	1.3	85.9	68.8
LeViT-256 (**ours**)	18.9	1120	81.6	6582	42.5	16.4	86.8	70.0
DeiT-Tiny	5.9	1220	76.6	3973	39.1	16.8	83.9	65.4
EfficientNet B3	12	1800	81.6	1272	5.9	0.8	86.8	70.6
LeViT-384 (**ours**)	39.1	2353	82.6	4165	23.1	9.4	87.6	71.3
EfficientNet B4	19	4200	82.9	606	2.5	0.5	88.0	72.3
DeiT-Small	22.5	4522	82.6	1931	13.7	7.6	87.8	71.7

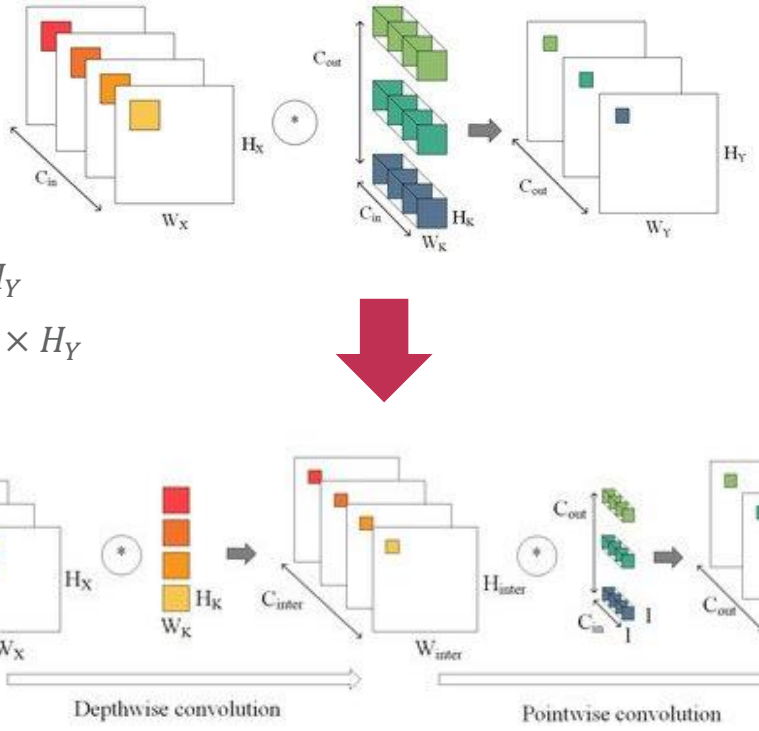
MobileViT: Introduction

- MobileViT: Light-weight, General-Purpose, and Mobile-friendly Vision Transformer
  - Combine the strengths of CNNs and ViTs to build a light-weight and low latency network for mobile devices
    - Performance of light-weight ViTs are not enough
      - DeIT is 3% less accurate compared to MobileNetv3
    - FLOPs do not accurately reflect inference speed
      - PiT has 3x less FLOPs than DeIT. But similar inference speed on mobile device
    - Complex training recipes of previous ViTs
      - Aim for better performance than conventional light-weight CNNs with basic training recipes
      - Good generalization & robust to hyper-parameters (data augmentation and L2 regularization)

MobileViT: Background

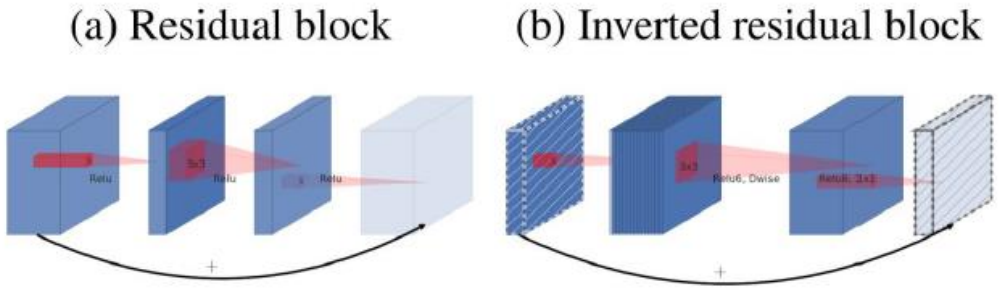
- MobileNetV1: Depthwise Separable Convolution
  - Divide Standard Convolution into Depthwise & Pointwise Convolution

  - Parameters
    - Standard: $W_K \times H_K \times C_{out} \times C_{in}$
    - Depthwise: $(W_K \times H_K + C_{out}) \times C_{in}$
  - Computation
    - Standard: $W_K \times H_K \times C_{out} \times C_{in} \times W_Y \times H_Y$
    - Depthwise: $(W_K \times H_K + C_{out}) \times C_{in} \times W_Y \times H_Y$

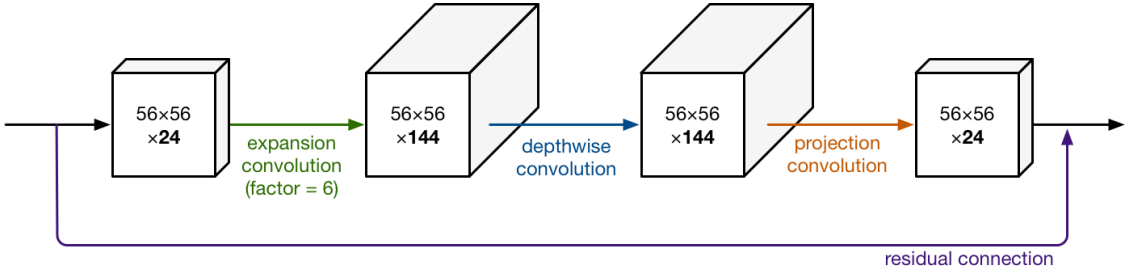


MobileVit: Background

- MobileNetV2: Inverted Residual Block
  - Residual (Bottleneck) vs Inverted Residual
    - Reverse classical Residual Block

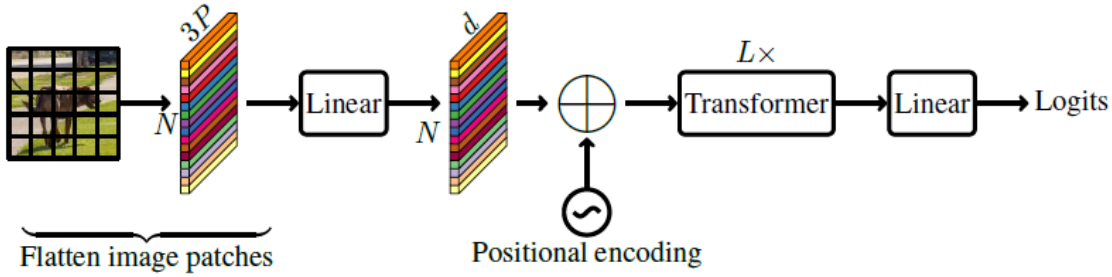


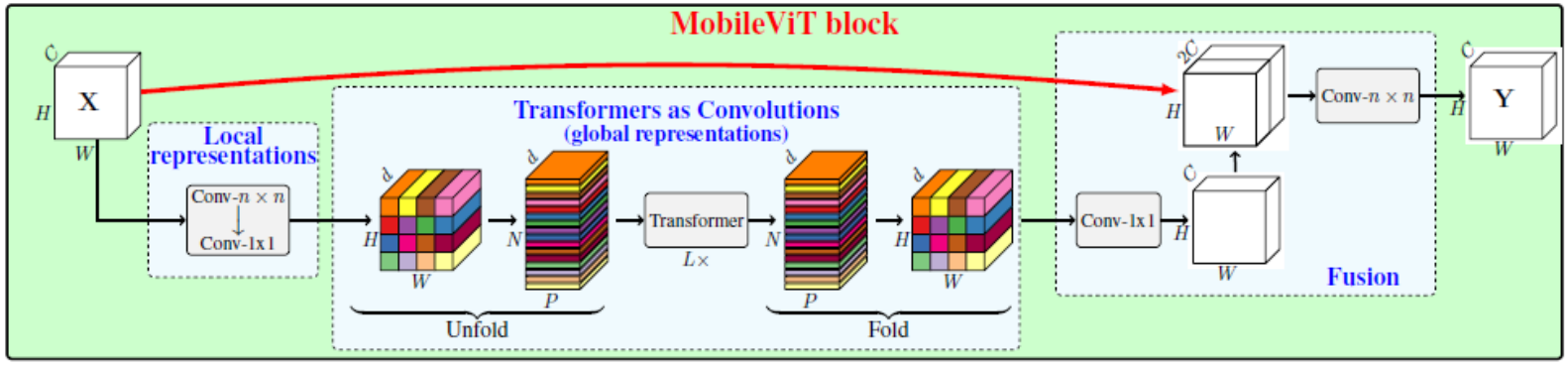
  - Inverted Residual Block Structure
    - Now a standard for many light-weight models (ex. ShuffleNet, EfficientNet, MobileNetV3 etc.)



MobileVit: Method

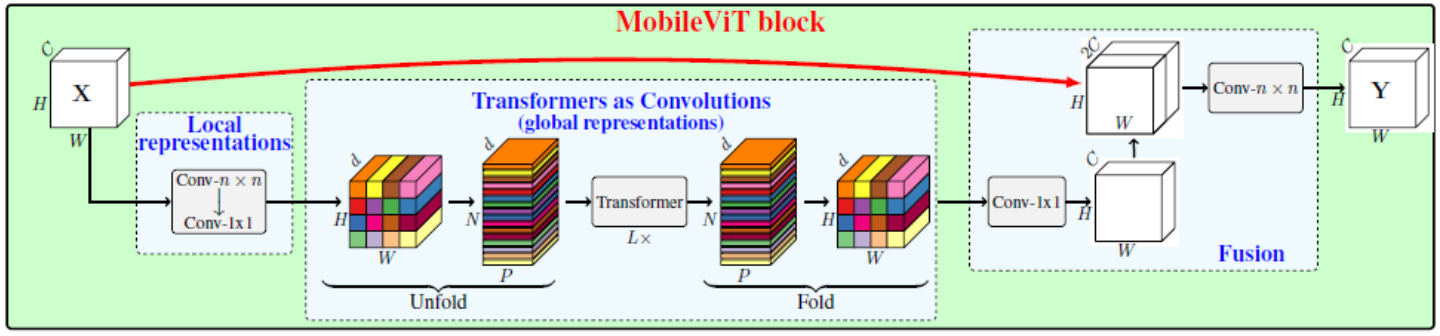
- MobileViT block





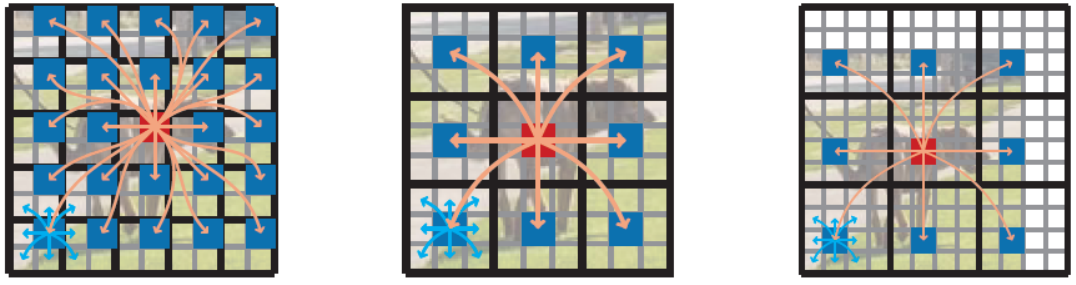
MobileVit: Method

- MobileViT block
  - Local spatial information
    - n x n conv + pointwise conv
  - Global information
    - Using transformer
    - 1 conv = unfold → matrix multiplication → fold
    - Transformers as convolutions
  - Absence of linear projection
    - No need for positional embeddings



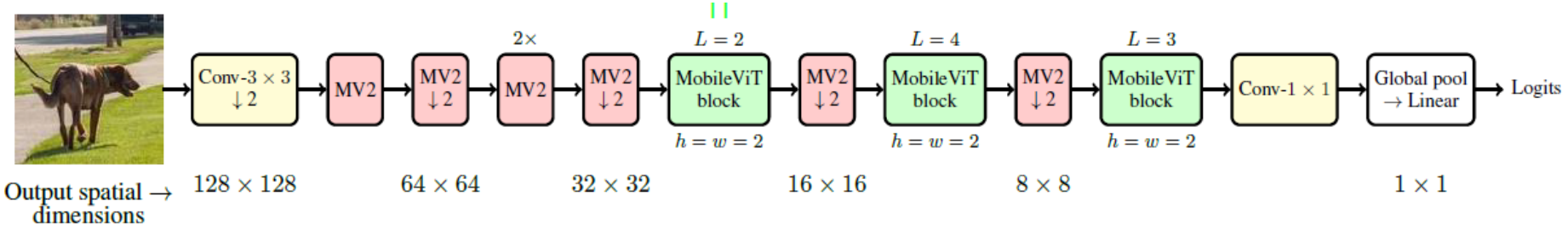
MobileVit: Method

- Effect of convolution + transformer
  - Kernel size (n x n) & patch size (h x w)



(a) $h = w = 2 < n = 3$ (b) $h = w = n = 3$ (c) $h = w = 4 > n = 3$

- Overall architecture

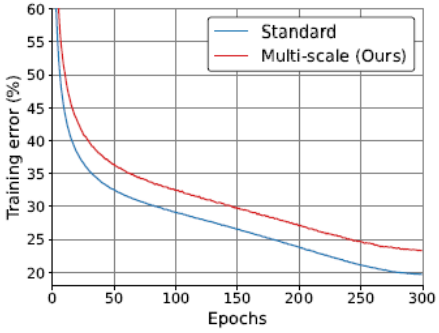


Conv-n x n : standard n x conv, MV2 : MobileNetV2 block, ↓2 : downsample by 2

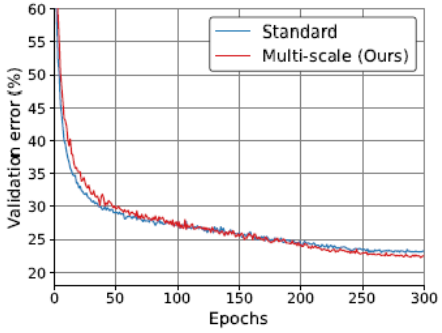
MobileVit: Method

- Multi-scale Sampler for Training Efficiency
  - Multi-scale learning only possible through fine-tuning for standard ViTs
    - Position embeddings need to be interpolated based on the input size
    - No position embedding necessary for MobileViT
  - Multi-scale learning with variably-size batch sizes
    - For randomly sampled spatial resolution of (H_t, W_t)
    - Batch size for t-th iteration: $b_t = \frac{H_n \, W_n b}{H_t \, W_t}$
    - Faster training & better generalization performance

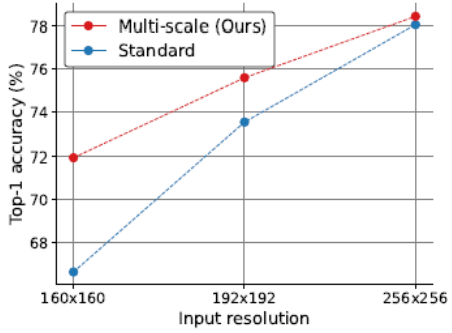
Sampler	# Updates	Epoch time
Standard	375 k	380 sec
Multi-scale (ours)	**232 k**	**270 sec**



(a) Training error



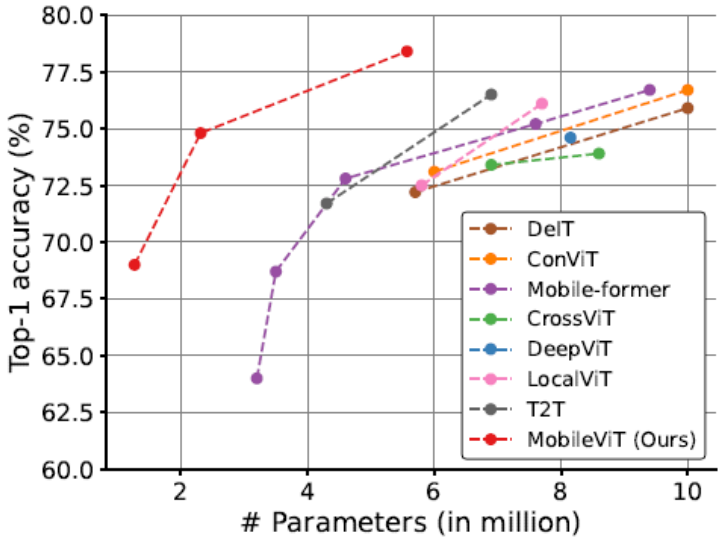
(b) Validation error



(c) Validation acc. vs. input resolution

MobileVit: Experiments

- Results on ImageNet-1k
  - Augmentation
    - Basic : Only standard augmentation techniques for CNN (e.g. ResNet)
    - Advanced : Combination of augmentation methods (e.g. MixUp, CutMix)



Row #	Model	Augmentation	# Params. ⇓	Top-1 ⇑
R1	DeIT	Basic	5.7 M	68.7
R2	T2T	Advanced	4.3 M	71.7
R3	DeIT	Advanced	5.7 M	72.2
R4	PiT	Basic	10.6 M	72.4
R5	Mobile-former	Advanced	4.6 M	72.8
R6	PiT	Advanced	4.9 M	73.0
R7	CrossViT	Advanced	6.9 M	73.4
R8	MobileViT-XS (Ours)	Basic	2.3 M	**74.8**
R9	CeiT	Advanced	6.4 M	76.4
R10	DeIT	Advanced	10 M	75.9
R11	T2T	Advanced	6.9 M	76.5
R12	ViL	Advanced	6.7 M	76.7
R13	LocalVit	Advanced	7.7 M	76.1
R14	Mobile-former	Advanced	9.4 M	76.7
R15	PVT	Advanced	13.2 M	75.1
R16	ConViT	Advanced	10 M	76.7
R17	PiT	Advanced	10.6 M	78.1
R18	BoTNet	Basic	20.8 M	77.0
R19	BoTNet	Advanced	20.8 M	78.3
R20	MobileViT-S (Ours)	Basic	5.6 M	**78.4**

MobileVit: Experiments

- Results on Detection
  - SSDLite on MS-COCO

Feature backbone	# Params. ⇓	mAP ⇑
MobileNetv3	4.9 M	22.0
MobileNetv2	4.3 M	22.1
MobileNetv1	5.1 M	22.2
MixNet	4.5 M	22.3
MNASNet	4.9 M	23.0
MobileViT-XS (Ours)	**2.7 M**	24.8
MobileViT-S (Ours)	5.7 M	**27.7**

(a) Comparison w/ light-weight CNNs

Feature backbone	# Params. ⇓	mAP ⇑
VGG	35.6 M	25.1
ResNet50	22.9 M	25.2
MobileViT-S (Ours)	**5.7 M**	**27.7**

(b) Comparison w/ heavy-weight CNNs

- Results on Segmentation
  - DeepLabv3 on PASCAL VOC 2012

Feature backbone	# Params. ⇓	mIOU ⇑
MobileNetv1	11.2 M	75.3
MobileNetv2	4.5 M	75.7
MobileViT-XXS (Ours)	1.9 M	73.6
MobileViT-XS (Ours)	2.9 M	**77.1**
ResNet-101	58.2 M	**80.5**
MobileViT-S (Ours)	6.4 M	79.1

MobileVit: Experiments

- Results on mobile device
  - Iphone 12 neural engine

Model	# Params. ⇓	FLOPs ⇓	Time ⇓	Top-1 ⇑
(R1) DeIT	5.7 M	1.3 G	10.99 ms	72.2
(R2) PiT	4.9 M	**0.7 G**	10.56 ms	73.0
(R3) MobileViT-XS (Ours; 8,4,2)	**2.3 M**	**0.7 G**	**5.93 ms**	73.8
(R4) MobileViT-XS (Ours; 2,2,2)	**2.3 M**	**0.7 G**	7.28 ms	**74.8**

(a) Comparisons with other light-weight ViTs

Model	# Params. ⇓	FLOPs ⇓	Time ⇓	Top-1 ⇑
MobileNetv2	3.5 M	**0.3 G**	**0.92 ms**	73.3
MobileViT-XS (Ours; 8,4,2)	**2.3 M**	0.7 G	5.93 ms	73.8
MobileViT-XS (Ours; 2,2,2)	**2.3 M**	0.7 G	7.28 ms	**74.8**

(a) ImageNet-1k classification

Backbone	# Params. ⇓	FLOPs ⇓	Time ⇓	mAP ⇑
MobileNetv2	4.3 M	**0.8 G**	**2.3 ms**	22.1
MobileViT-XS (Ours; 8,4,2)	**2.7 M**	1.6 G	10.7 ms	23.1
MobileViT-XS (Ours;2,2,2)	**2.7 M**	1.6 G	12.6 ms	**24.8**

(b) Object detection w/ SSDLite.

Backbone	# Params. ⇓	FLOPs ⇓	Time ⇓	mIOU ⇑
MobileNetv2	4.3 M	5.8 G	**6.5 ms**	75.7
MobileViT-XS (Ours)	**2.9 M**	**5.7 G**	25.1 ms	75.4
MobileViT-XS (Ours)	**2.9 M**	**5.7 G**	32.3 ms	**77.1**

(c) Semantic segmentation w/ DeepLabv3.

MobileVit: Experiments

- Discussion
  - Lack of hardware optimization for ViTs
    - ViTs are slower than CNNs on mobile platform
    - CUDA kernels exists for GPU but not for mobile
    - CPU is optimized for variable devices

Kernel size	Theoretical FLOPs (B)	Time usage (ms)	Theoretical TFLOPS
1×1	420.9	84.5	9.96
3×3	3788.1	198.8	**38.10**
5×5	10522.6	2092.5	10.57
7×7	20624.4	4394.3	9.38

FLOPs & Time of Convolution Operators on 1080Ti GPU

Model	# Params ⇓	FLOPs ⇓	Top-1 ⇑	Inference time ⇓		
				iPhone12 - CPU	iPhone12 - Neural Engine	NVIDIA V100 GPU
MobileNetv2	3.5 M	**0.3 G**	73.3	**7.50 ms**	**0.92 ms**	**0.31 ms**
DeIT	5.7 M	1.3 G	72.2	28.15 ms	10.99 ms	0.43 ms
PiT	4.9 M	0.7 G	73.0	24.03 ms	10.56 ms	0.46 ms
MobileViT (Ours)	**2.3 M**	0.7 G	**74.8**	17.86 ms	7.28 ms	0.62 ms/0.47 ms†

Comparison of ViT models and MobileNetv2 inference speed on different hardware platforms

Conclusion

- LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference
  - Incorporation of CNN into Vision Transformers for faster inference
    - Replacement of MLP based operations to convolution-based operations
      - MLP → 1x1 conv, layer norm → batchnorm
    - Trade-off between CNN + ViT mixture model
      - Applying convolutional layers to early stages of ViT
- MobileViT: Light-weight, General-Purpose, and Mobile-friendly Vision Transformer
  - MobileViT block
    - Combination of convolution and self-attention to consider both local and global context
  - Incorporation of MobileViT block within pre-defined MobileNetV2
    - Hight performance of ViT + computational efficiency of MobileNet

Reference

- Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).
- Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
- Dosovitskiy, Alexey, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." *International Conference on Learning Representations*. 2020.
- Touvron, Hugo, et al. "Training data-efficient image transformers & distillation through attention." *International Conference on Machine Learning*. PMLR, 2021.
- Graham, Ben, et al. "LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference." *arXiv preprint arXiv:2104.01136* (2021).
- Mehta, Sachin, and Mohammad Rastegari. "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer." *arXiv preprint arXiv:2110.02178* (2021).