Recent neural architectures for vision I: Discriminative models

AI602: Recent Advances in Deep Learning

Lecture 1

Jinwoo Shin

KAIST AI

Basic knowledge in **machine learning & classic model design** are assumed:
(e.g., **AI501, AI502, AI601 course**)

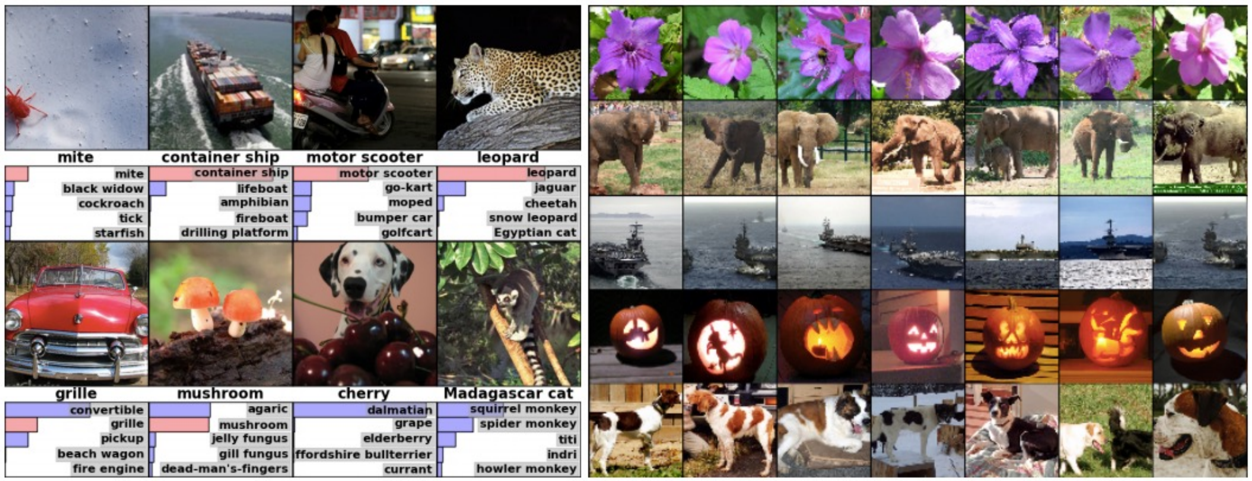
- **Machine Learning**
  - Problems: classification, regression, etc.
  - Optimization: stochastic gradient descent (SGD), regularizations, etc.
  - Deep Neural Networks: basic structures, representation learning, etc.

- **Classic model designs**
  - **Convolutional Neural Networks (CNNs)**
    - Basic operations: convolution, spatial pooling, etc.
    - Design techniques: skip-connection, normalization, etc.
    - Some notable models: AlexNet, Inception, ResNet, etc.
  - **Transformers**
    - Transformer architecture: token data structure, self-attention, etc.

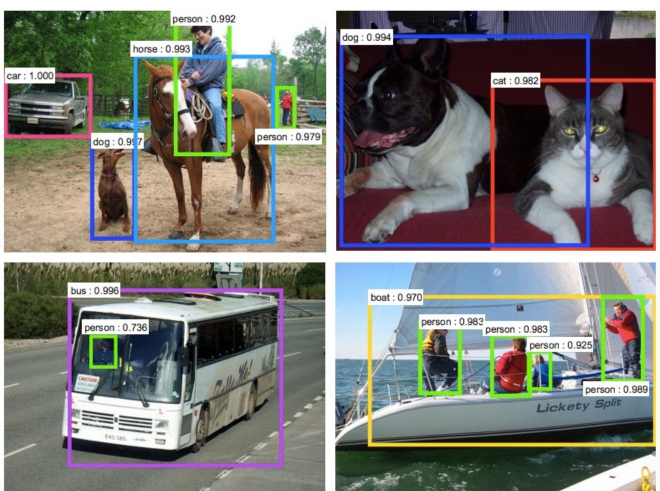
Overview: Convolutional Neural Networks

Convolutional neural networks have been tremendously successful in practical applications;

Classification and retrieval [Krizhevsky et al., 2012]



Detection [Ren et al., 2015]



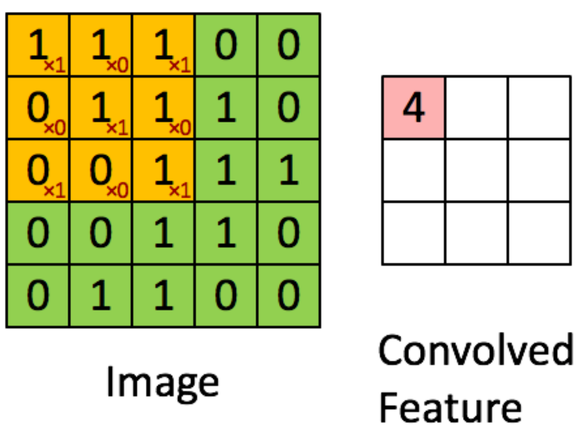
Segmentation [Farabet et al., 2013]

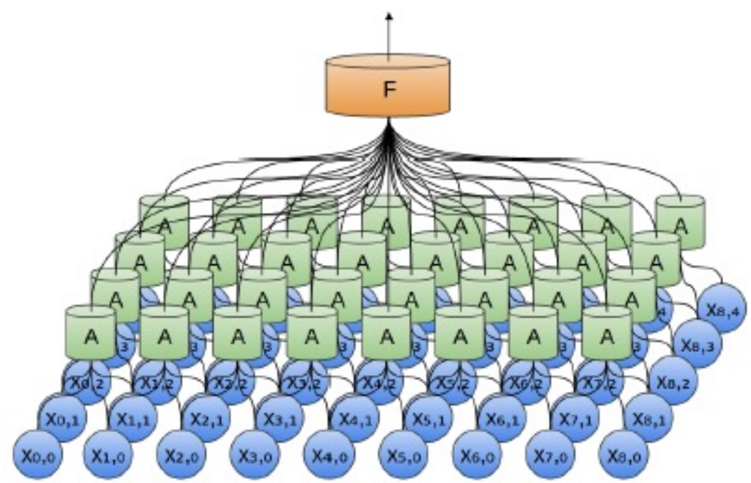
Neural networks that use convolution in place of general matrix multiplication

- Sharing parameters across **multiple image locations**
- Translation equivariant (invariant with **pooling**) operation

Specialized for processing data that has a known, grid-like topology

- e.g., time-series data (1D grid), image data (2D grid)

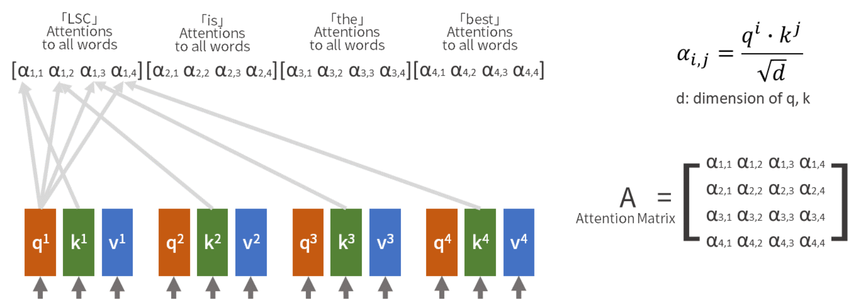




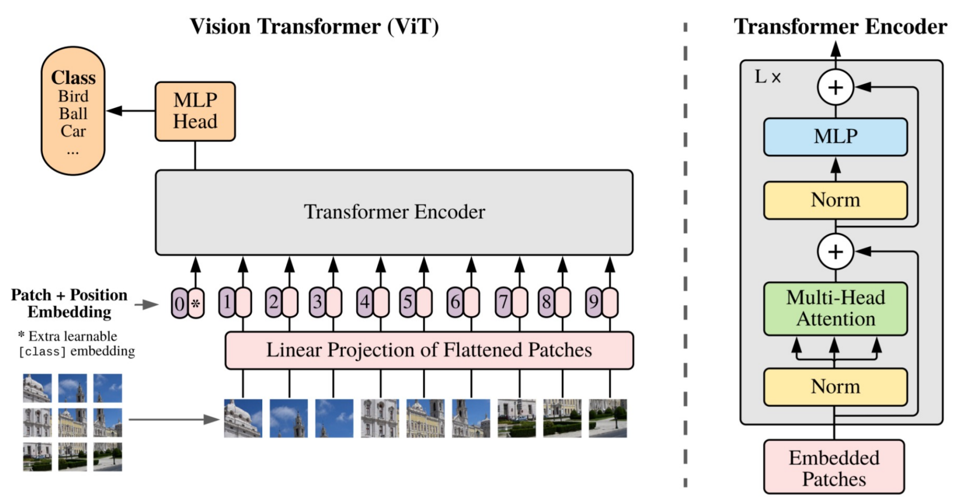
*sources :
- https://www.cc.gatech.edu/~san37/post/dlhc-cnn/
- http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Vision transformers with self-attention for 2D spatial data also emerged recently

- Shares parameters across **multiple image locations**
  - However, self-attention adapts different weights per each location



- Very **small inductive-bias** towards image data; **everything is learned from data**!



Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

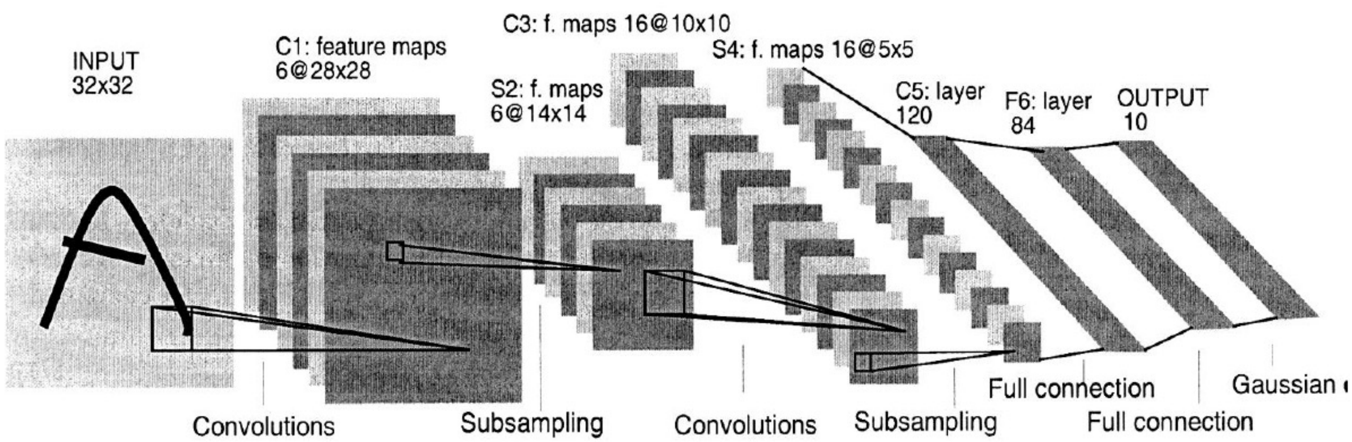
Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Typically, designing a CNN model requires some effort

- There are a lot of **design choices**: # layers, # filters, sizes of kernel, pooling, …
- It is **costly** to measure the performance of each model and choose the best one

Example: LeNet for handwritten digits recognition [LeCun et al., 1998]

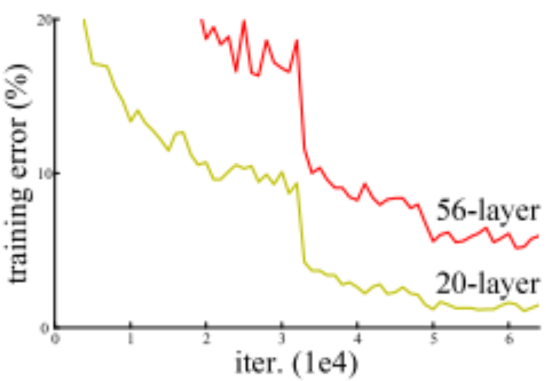


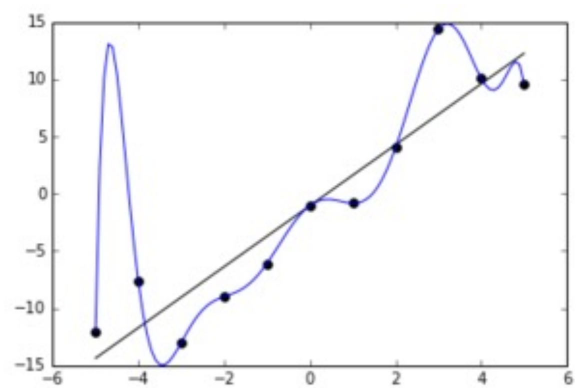
- However, **LeNet** is not enough to solve real-world problems in AI domain
  - CNNs are typically applied to extremely complicated domains, e.g. raw RGB images
  - We need to design **a larger model** to solve them adequately

Problem: The larger the network, the more difficult it is to design

1. **Optimization difficulty**
   - When the training loss is degraded
   - Deeper networks are typically much harder to optimize
   - Related to gradient vanishing and exploding
2. **Generalization difficulty**
   - The training is done well, but the testing error is degraded
   - Larger networks are more likely to over-fit, i.e., regularization is necessary

- Good architectures should be **scalable** that solves both of these problems

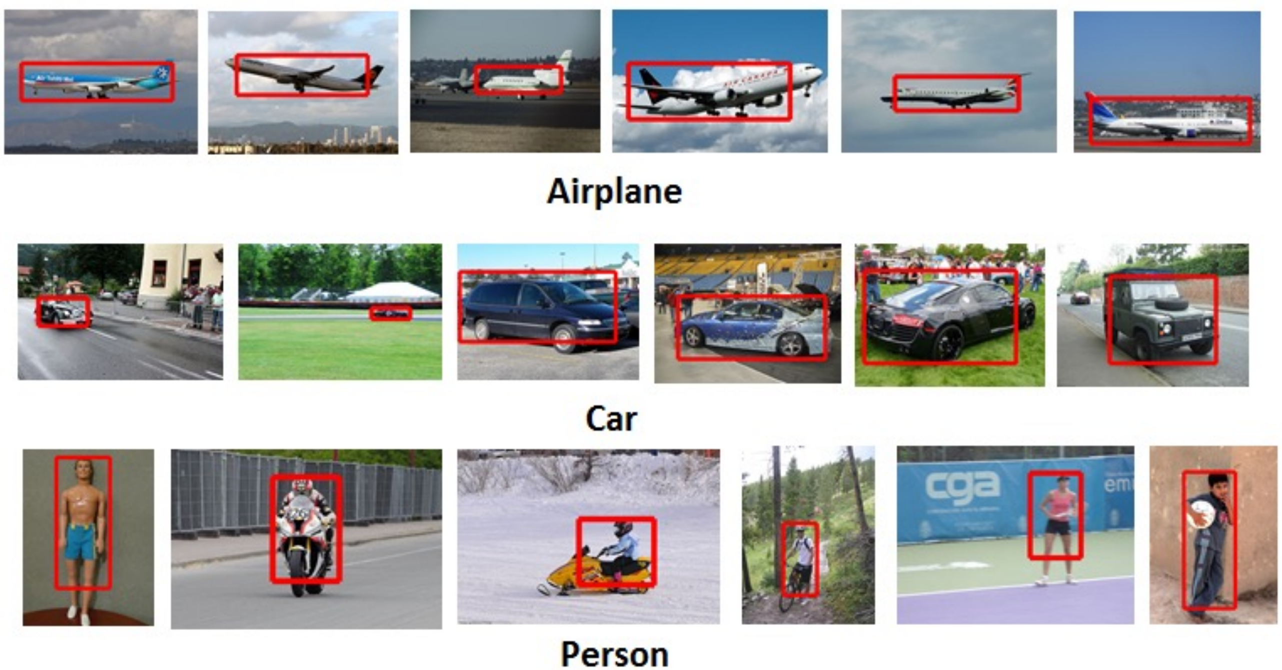




*sources :
- He et al. "Deep residual learning for image recognition". CVPR 2016.
- https://upload.wikimedia.org/wikipedia/commons/thumb/6/68/Overfitted_Data.png/300px-Overfitted_Data.png

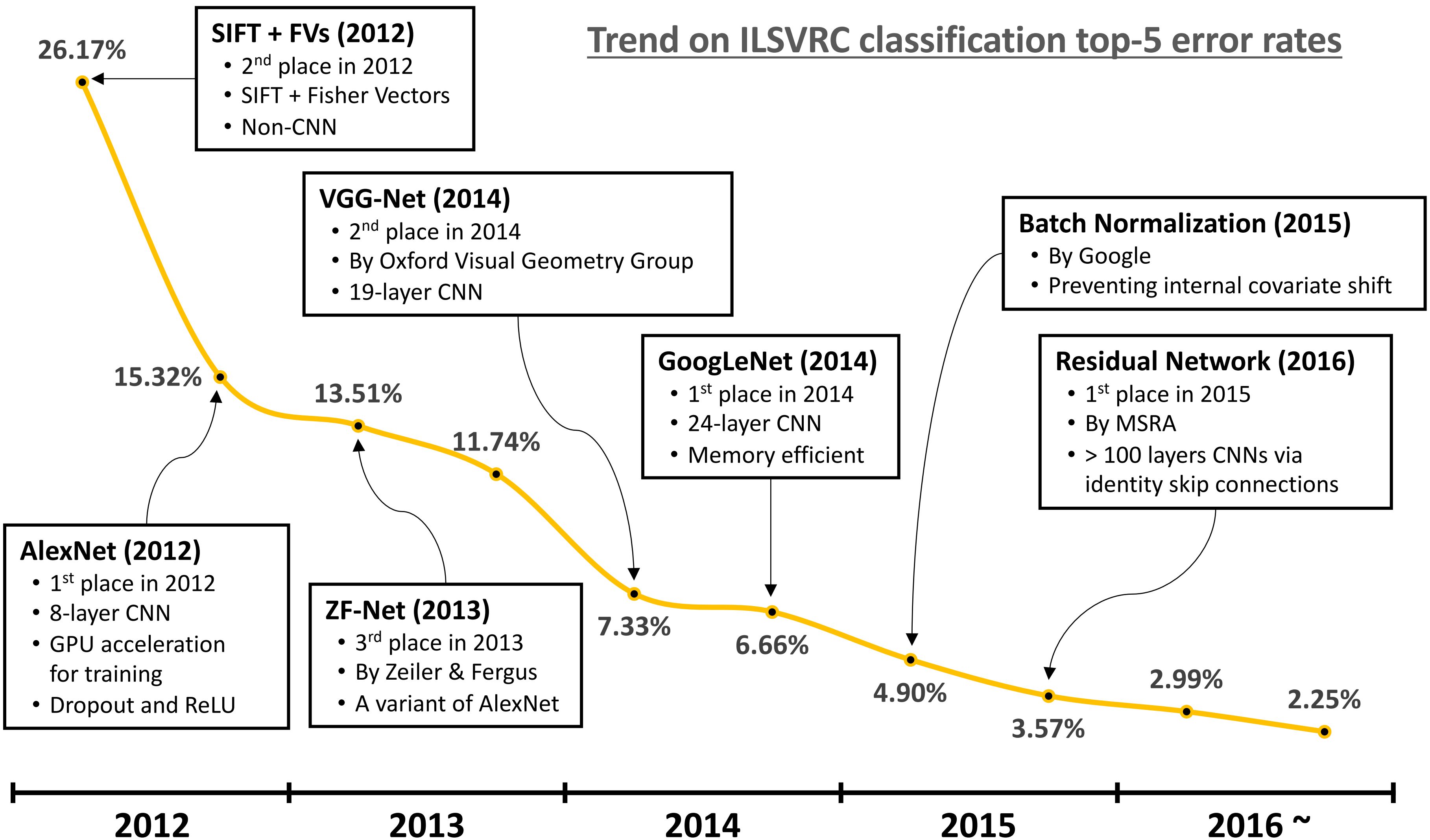
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- ImageNet dataset: a large database of visual objects
  - ~14M labeled images, 20K classes
  - Human labels via Amazon MTurk
- Classification: 1,281,167 images for training / 1,000 categories
- Annually ran from 2010 to 2017, and now hosted by Kaggle
- For details, see [Russakovsky et al., 2015]

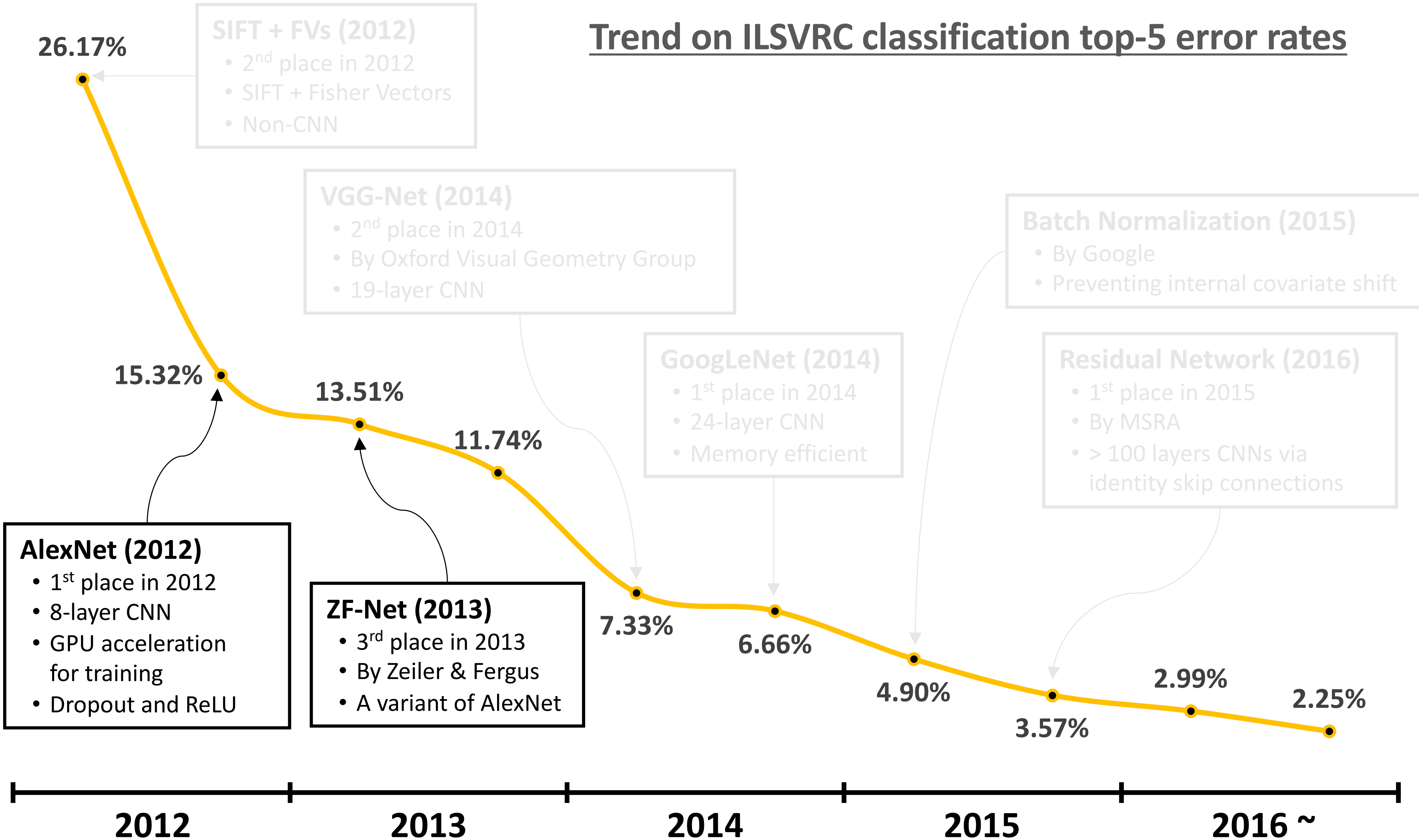


*source : http://visgraph.cse.ust.hk/ilsvrc/files/tesor.png

ILSVRC contributed greatly to development of CNN architectures



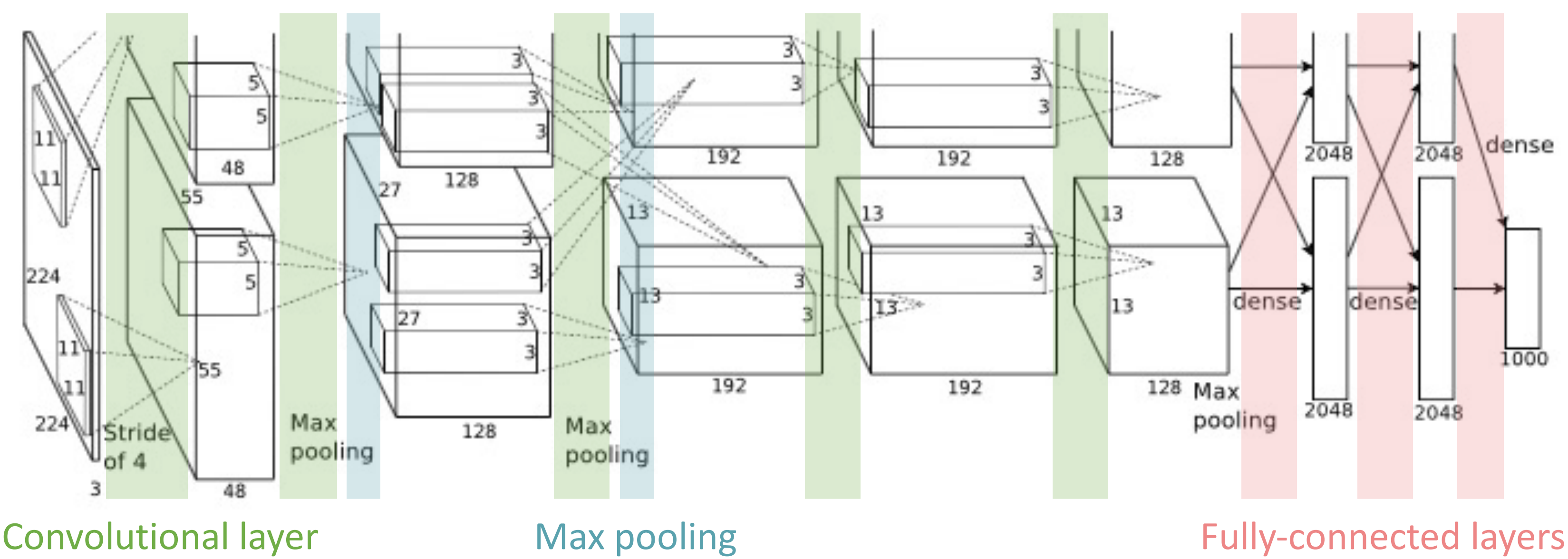
ILSVRC contributed greatly to development of CNN architectures



The first winner to use CNN in ILSVRC, with an astounding improvement

- Top-5 error is largely improved: 25.8% → **15.3%**
- The 2nd best entry at that time was **26.2%**

- 8-layer CNN (5 Conv + 3 FC)
- Utilized 2 GPUs (GTX-580 × 2) for training the network
  - Split a single network into 2 parts to distribute them into each GPU



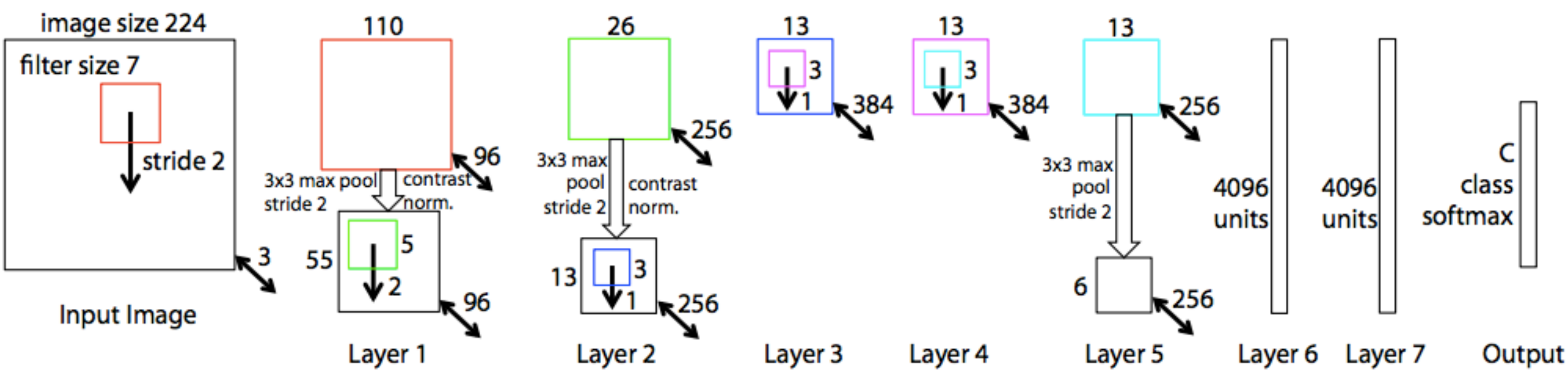
*source : Krizhevsky et al. "Imagenet classification with deep convolutional neural networks". NIPS 2012

A simple variant of AlexNet, placing the 3rd in ILSVRC'13 (15.3% → 13.5%)

- Smaller kernel at input: $11 \times 11 \rightarrow 7 \times 7$
- Smaller stride at input: $4 \rightarrow 2$
- The # of hidden filters are doubled

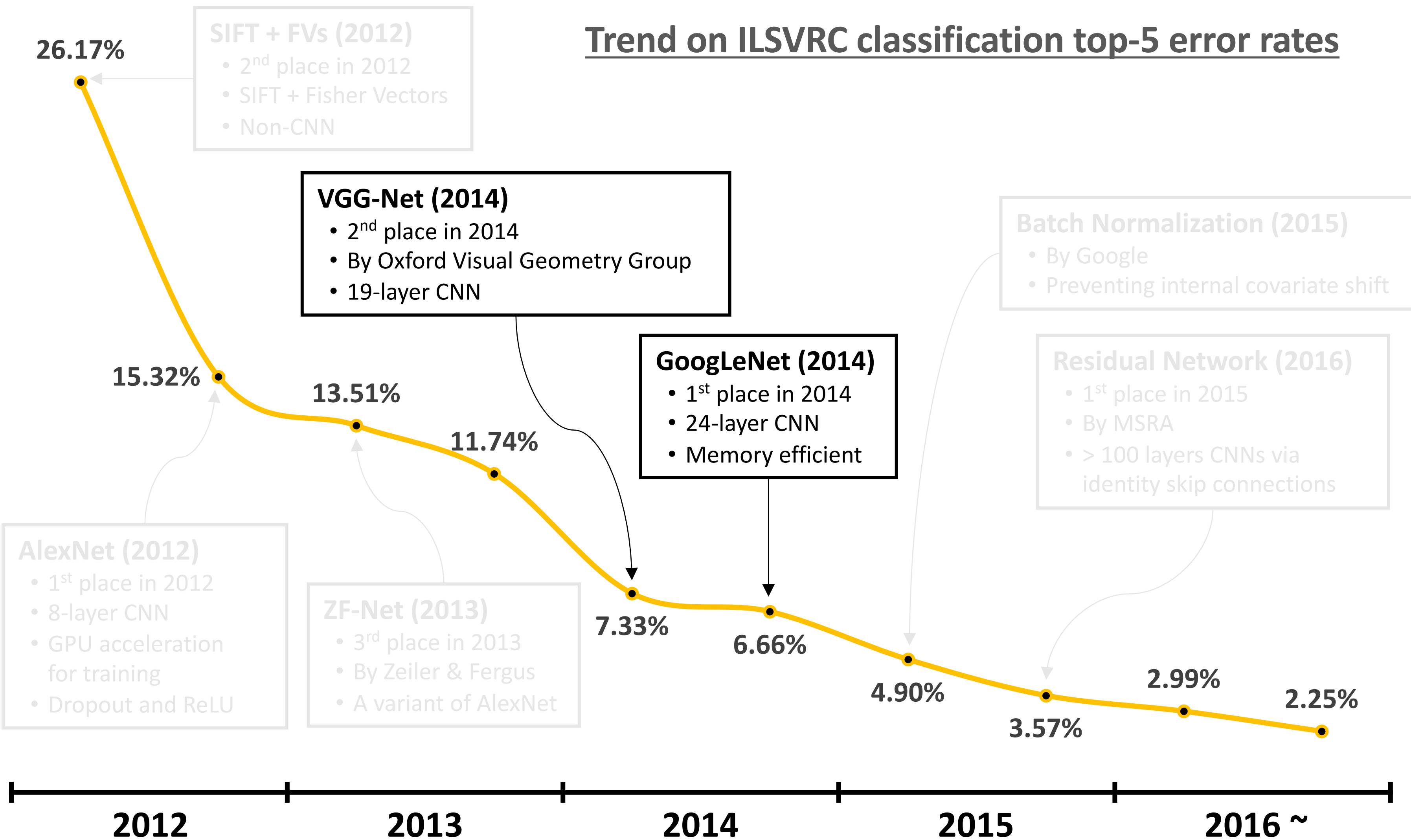
Lessons

1. Design principle: Use smaller kernel, and smaller stride
2. CNN architectures can be very sensitive on hyperparameters



*source : Zeiler et al., "Visualizing and understanding convolutional networks". ECCV 2014

ILSVRC contributed greatly to development of CNN architectures



Evolution of CNN Architectures: VGGNet and GoogleNet

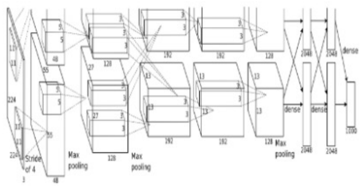
Networks were getting deeper

- AlexNet: 8 layers
- VGGNet: **19** layers
- GoogleNet: **24** layers

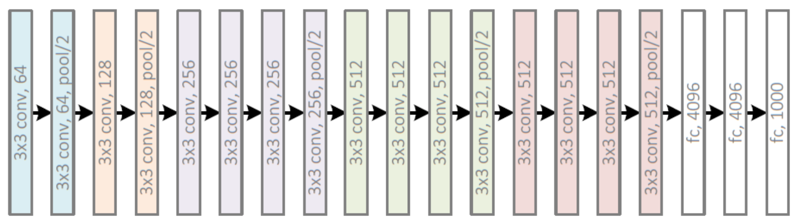
Both focused on parameter efficiency of each block

- Mainly to allow larger networks computable at that time

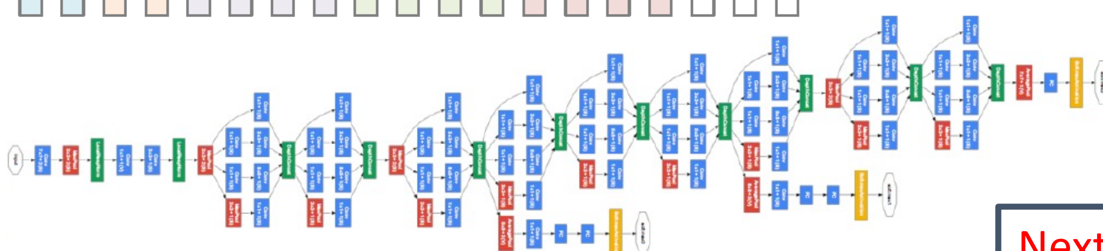
AlexNet



VGGNet



GoogLeNet



Next, VGGNet

*sources :
- Krizhevsky et al. “Imagenet classification with deep convolutional neural networks”. NIPS 2012
- Simonyan et al., “Very deep convolutional networks for large-scale image recognition”. arXiv 2014.
- Szegedy et al., “Going deeper with convolutions”. CVPR 2015

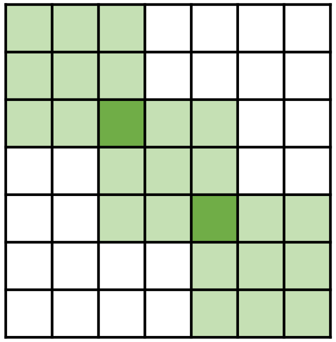
The 2nd place in ILSVRC'14 (11.7% → 7.33%)

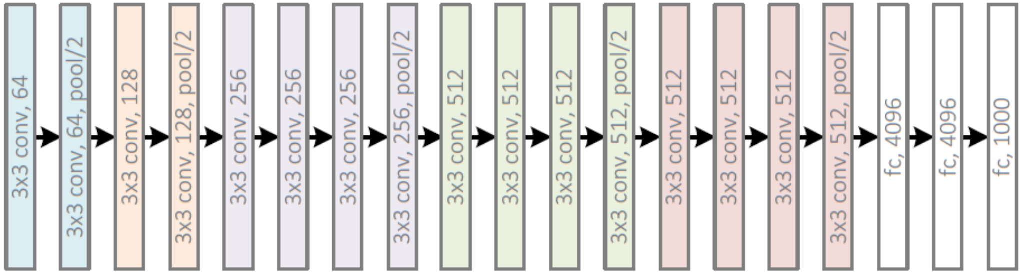
- Designed using only 3×3 kernels for convolutions

Lesson: **Stacking multiple 3×3** is advantageous than using other kernels

Example: $((3\times3)\times3)$ v.s. (7×7)

- Essentially, they get the same receptive field
- $((3\times3)\times3)$ have less # parameters
  - $3\times\left(\mathrm{C}\times\left((3\times3)\times\mathrm{C}\right)\right) = \mathbf{27C^2}$
  - $\mathrm{C}\times\left((7\times7)\times\mathrm{C}\right) = \mathbf{49C^2}$
- $((3\times3)\times3)$ gives more non-linearities





Next, GoogLeNet

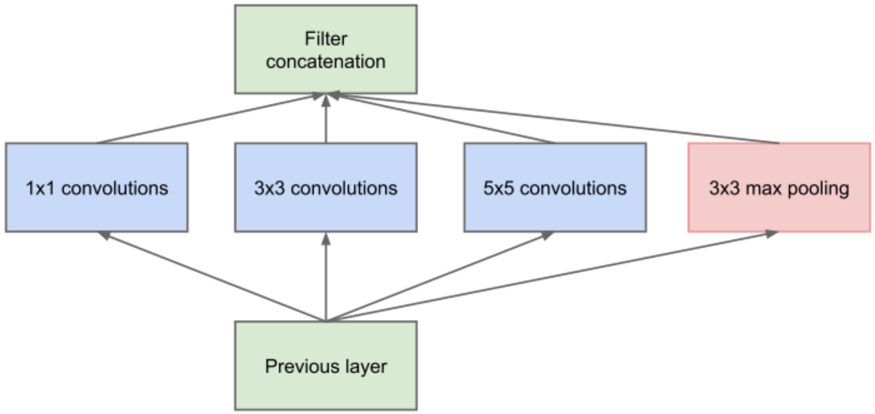
*source : Simonyan et al., "Very deep convolutional networks for large-scale image recognition". arXiv 2014.

The winner of ILSVRC'14 (11.7% → 6.66%)

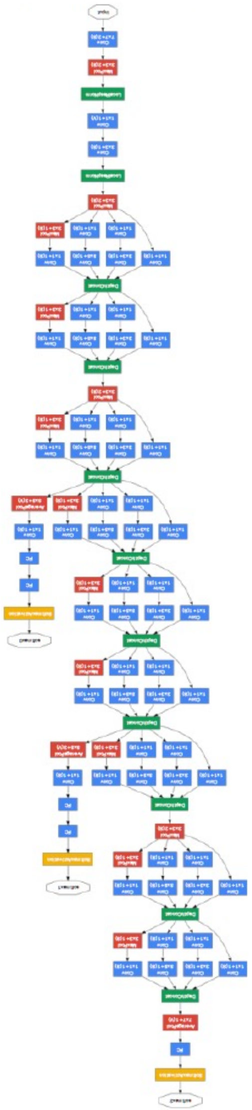
- Achieved 12× fewer parameters than AlexNet

Inception module

- Multiple operation paths with different receptive fields
- Each of the outputs are **concatenated** in filter-wise
- Capturing sparse patterns in a stack of features

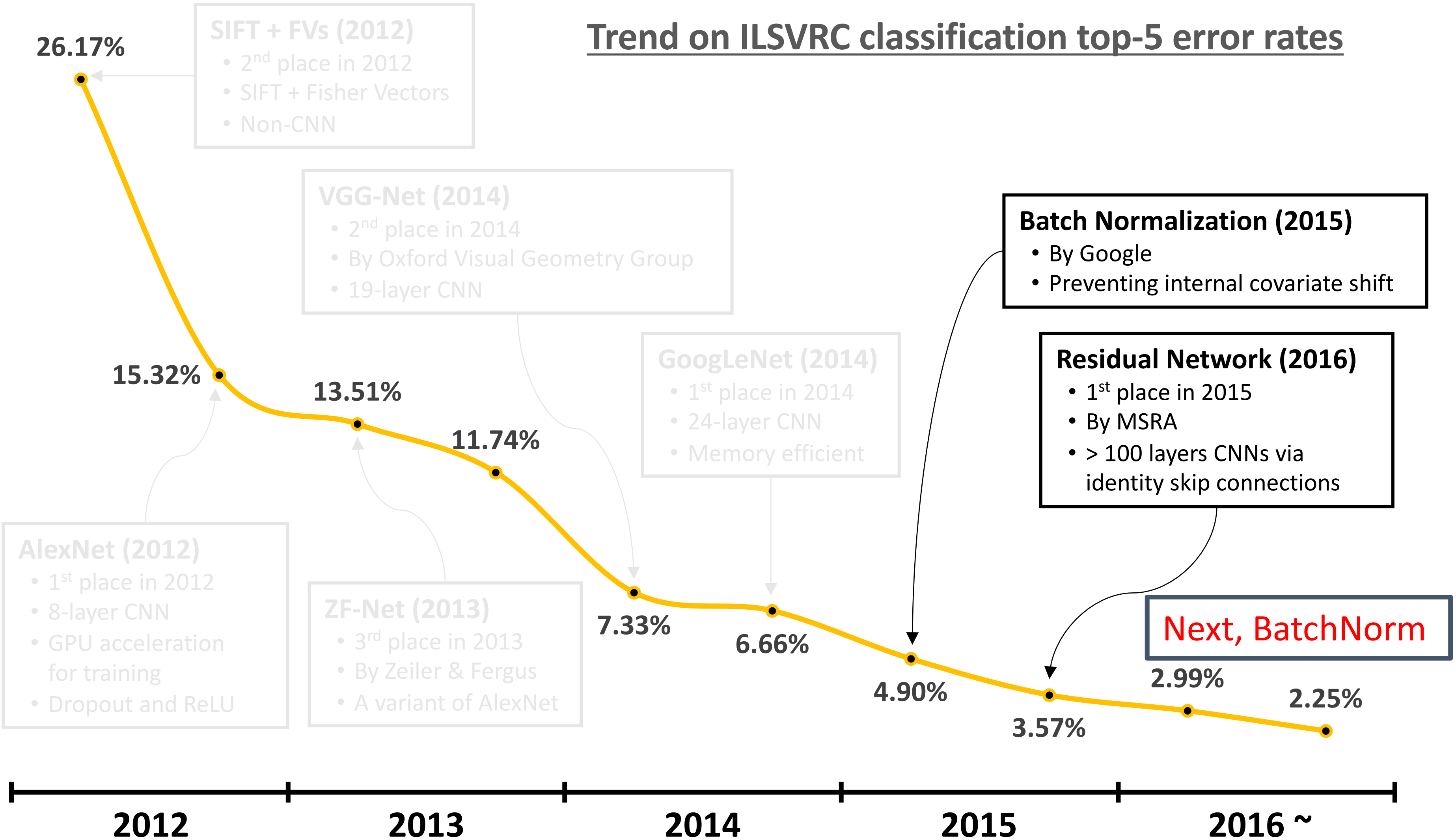


(a) Inception module, naïve version



*source : Szegedy et al., "Going deeper with convolutions". CVPR 2015

ILSVRC contributed greatly to development of CNN architectures



Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

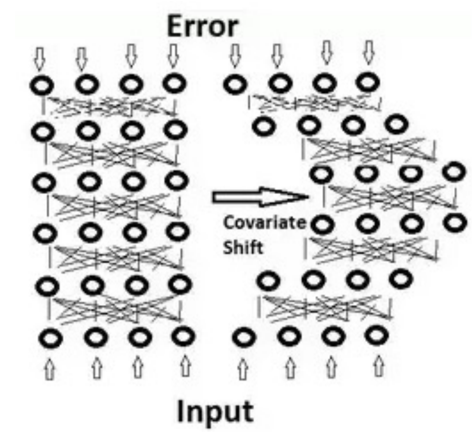
- Patch-based architectures for vision
- New design paradigms

Training a deep network well had been a delicate task

- It requires a careful initialization, with adequately low learning rate
- **Gradient vanishing**: networks containing saturating non-linearity

Ioffe et al. (2015): Such difficulties are come from **internal covariate shift**

Motivation: “The cup game analogy”

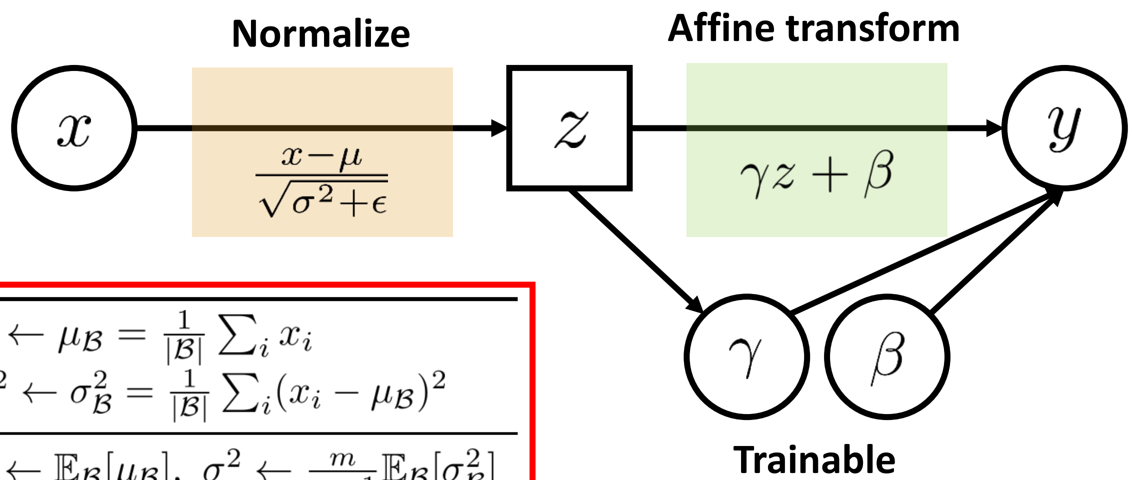


- Similar problem happens during training of deep neural networks
- Updates in early layers may shift the inputs of later layers too much

*sources :
- Ioffe et al., “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. ICML 2015
- http://pages.cs.wisc.edu/~shavlik/cs638/lectureNotes/Batch_Normalization.pptx
- https://www.quora.com/Why-does-batch-normalization-help

Batch normalization (BN) accelerates neural network training by eliminating internal covariate shift inside the network

Idea: A normalization layer that behaves differently in training and testing



Training	$\mu \leftarrow \mu_{\mathcal{B}} = \frac{1}{\vert\mathcal{B}\vert}\sum_i x_i$ $\sigma^2 \leftarrow \sigma^2_{\mathcal{B}} = \frac{1}{\vert\mathcal{B}\vert}\sum_i (x_i - \mu_{\mathcal{B}})^2$
Testing	$\mu \leftarrow \mathbb{E}_{\mathcal{B}}[\mu_{\mathcal{B}}],\ \sigma^2 \leftarrow \frac{m}{m-1}\mathbb{E}_{\mathcal{B}}[\sigma^2_{\mathcal{B}}]$

1. During training, input distribution of y only depends on γ and β
   - Training mini-batches are always normalized into mean 0, variance 1
2. There is some gap between $\mu_{\mathcal{B}}$ and $\mathbb{E}[\mu_{\mathcal{B}}]$ ($\sigma^2_{\mathcal{B}}$, resp.)
   - Noise injection effect for each mini-batch ⇒ Regularization effect

*source : Ioffe et al., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

Batch normalization (BN) accelerates neural network training by eliminating internal covariate shift inside the network

- BN allows much **higher learning rates**, i.e. faster training
- BN **stabilizes** gradient vanishing on saturating non-linearities
- BN also has its own **regularization effect**, so that it allows to reduce weight decay, and to remove dropout layers

- BN makes GoogLeNet much easier to train with great improvements

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***

Next, ResNet

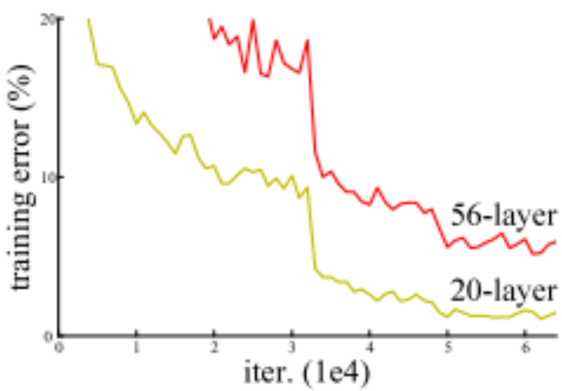
*source : Ioffe et al., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". ICML 2015

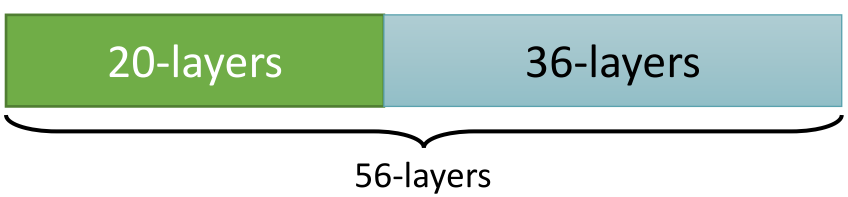
The winner of ILSVRC’15 (6.66% → 3.57%)

- **ResNet** is the first architecture succeeded to train >100-layer networks
  - Prior works could until ~30 layers, but failed for the larger nets

What was the problem?

- 56-layer net gets higher training error than 20-layers network
- Deeper networks are much harder to optimize even if we use BNs
- It’s not due to overfitting, but optimization difficulty
- **Quiz**: Why is that?





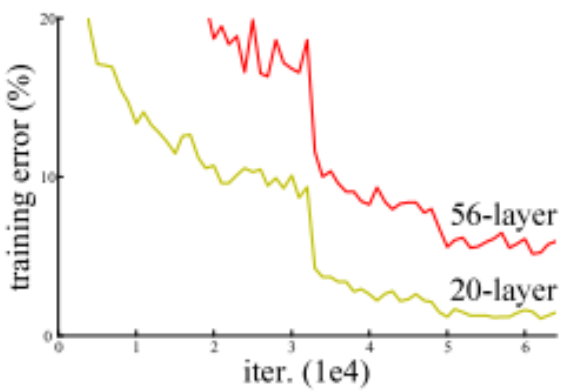
*source : He et al., “Deep residual learning for image recognition”. CVPR 2016

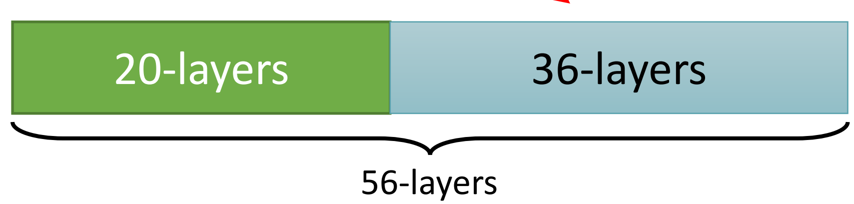
The winner of ILSVRC'15 (6.66% → 3.57%)

- **ResNet** is the first architecture succeeded to train >100-layer networks
    - Prior works could until ~30 layers, but failed for the larger nets

What was the problem?

- It's not due to overfitting, but optimization difficulty
- **Quiz**: Why is that?
- If the 56-layer model optimized well, then it **must be better** than the 20-layer
    - There is a trivial solution for the 36-layer: identity





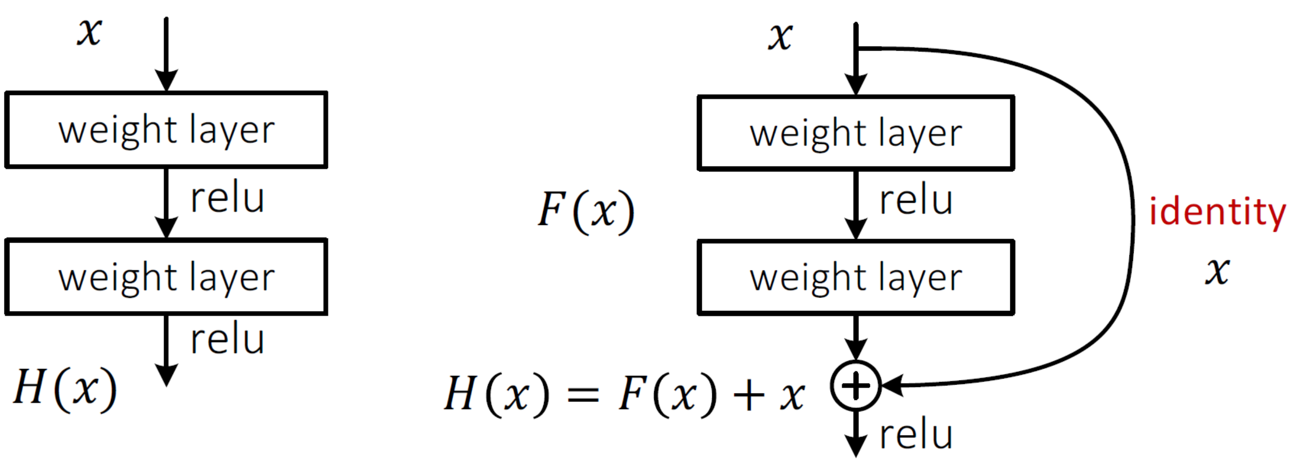
*source : He et al., "Deep residual learning for image recognition". CVPR 2016

Motivation: A non-linear layer may struggle to represent an identity function

- Due to its internal non-linearities, e.g. ReLU
- This may cause the optimization difficulty on large networks

Idea: Reparametrize each layer to make them easy to represent an *identity*

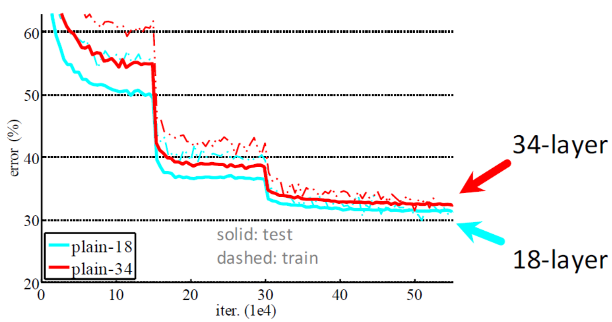
- When all the weights are set to zero, the layer represents an identity



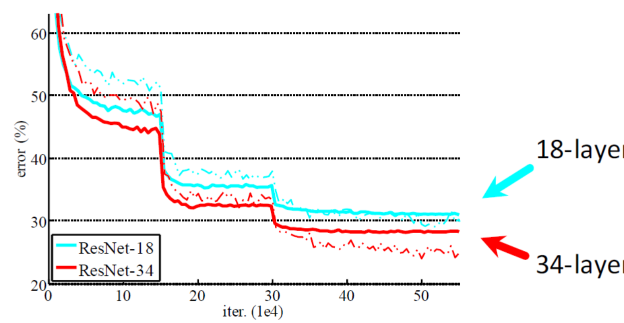
*source : He et al., "Deep residual learning for image recognition". CVPR 2016

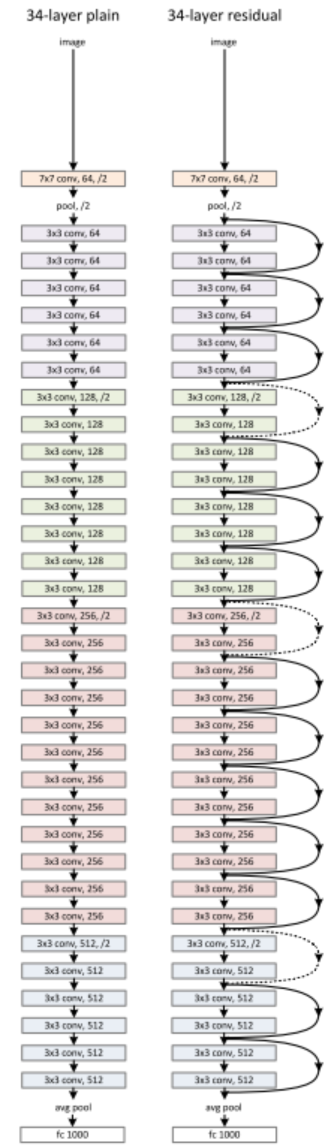
Plain nets v.s. ResNets

ImageNet plain nets

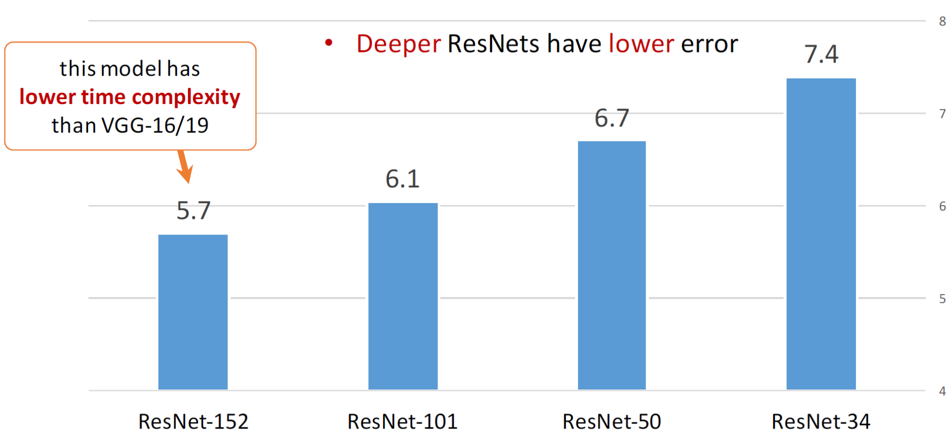


ImageNet ResNets





- Deeper ResNets can be trained without any difficulty



10-crop testing, top-5 val error (%)

*sources :
- He et al., “Deep residual learning for image recognition”. CVPR 2016
- He, Kaiming, "Deep Residual Networks: Deep Learning Gets Way Deeper.” 2016.

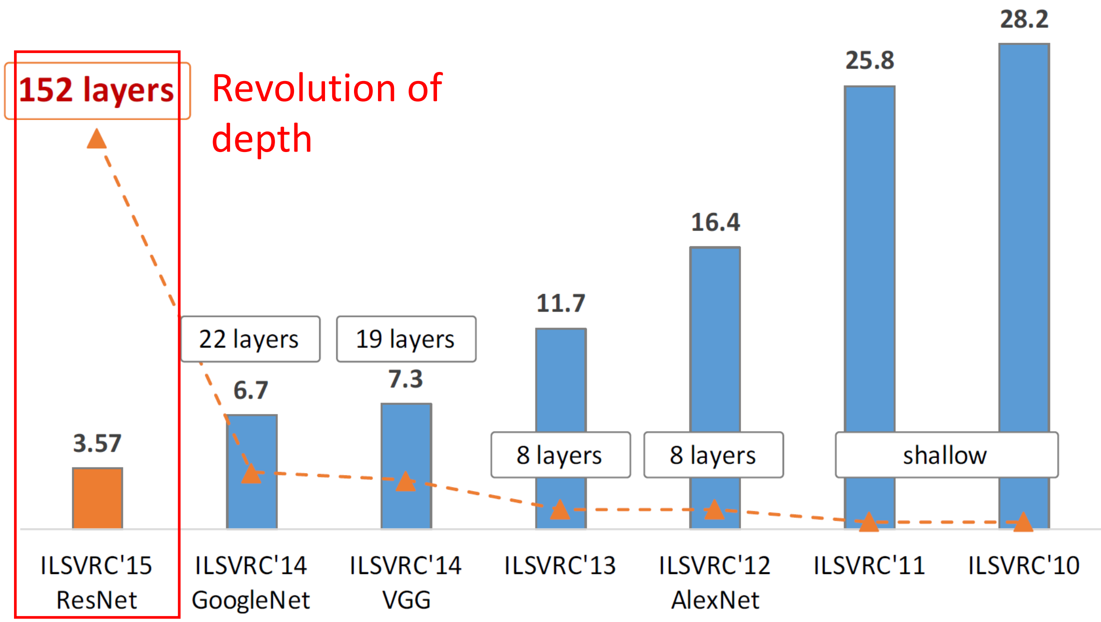
ResNet [He et al., 2016a]

- Identity connection resolved a major difficulty on optimizing large networks

Revolution of depth: Training >100-layer network without difficulty

  - Later, ResNet is revised to allow to train up to >1000 layers [He et al., 2016b]

- ResNet also shows good generalization ability as well

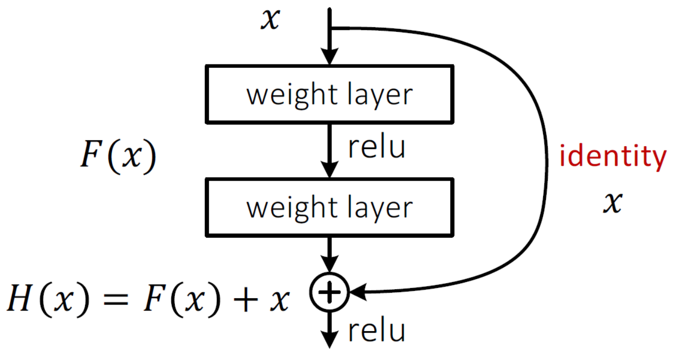


ImageNet Classification top-5 error (%)

*sources :
- He et al., "Deep residual learning for image recognition". CVPR 2016
- Kaiming He, "Deep Residual Networks: Deep Learning Gets Way Deeper." 2016.
- He et al. "Identity mappings in deep residual networks.", ECCV 2016

Various architectures now are based on ResNet

- ResNet with stochastic depth [Huang et al., 2016]
- Wide ResNet [Zagoruyko et al., 2016]
- ResNet in ResNet [Targ et al., 2016]
- ResNeXt [Xie et al., 2016]
- PyramidNet [Han et al., 2016]
- Inception-v4 [Szegedy et al., 2017]
- DenseNet [Huang et al., 2017]
- Dual Path Network [Chen et al., 2017]



Transition of design paradigm: Optimization ⇒ Generalization

- People are now less concerned about optimization problems in a model
- Instead, they now focus more on its generalization ability
- "How well does an architecture generalize as its scale grows?"

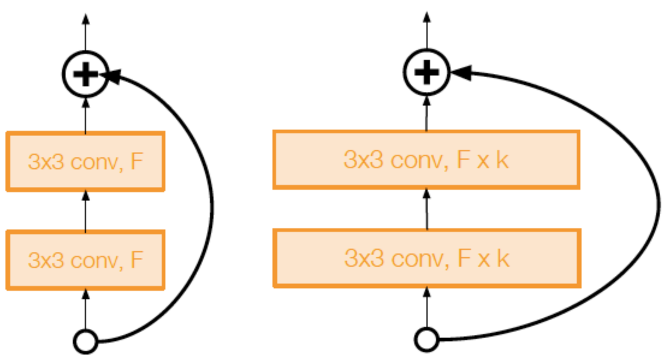
*source : He et al., "Deep residual learning for image recognition". CVPR 2016

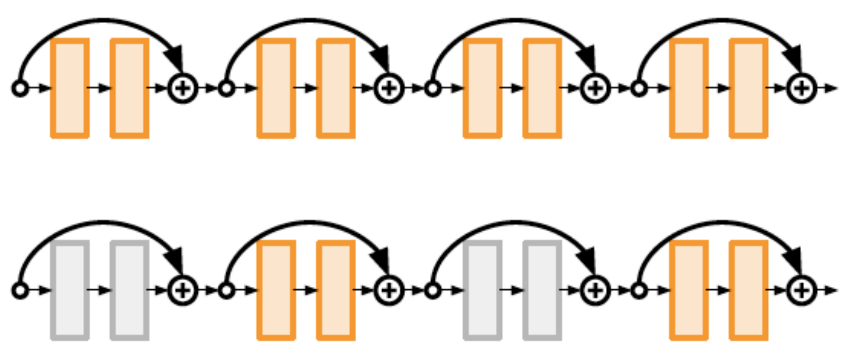
Wide Residual Networks [Zagoruyko et al., 2016]

- Residuals can also work to enlarge the width, not only its depth
- Residual blocks with $\times k$ wider filters
- Increasing width instead of depth can be more computationally efficient
  - GPUs are much better on handling "**wide-but-shallow**" than "thin-but-deep“
- WRN-50 outperforms ResNet-152

Deep Networks with Stochastic Depth [Huang et al., 2016]

- Randomly drop a subset of layers during training
- Bypassing via identity connections
- Reduces gradient vanishing, and training time as well

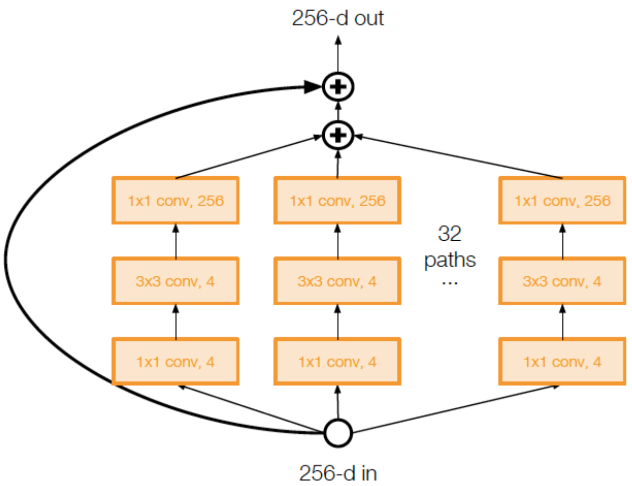




*source : Fei-Fei Li et al. (2018), CS231n Lecture 9, Stanford University

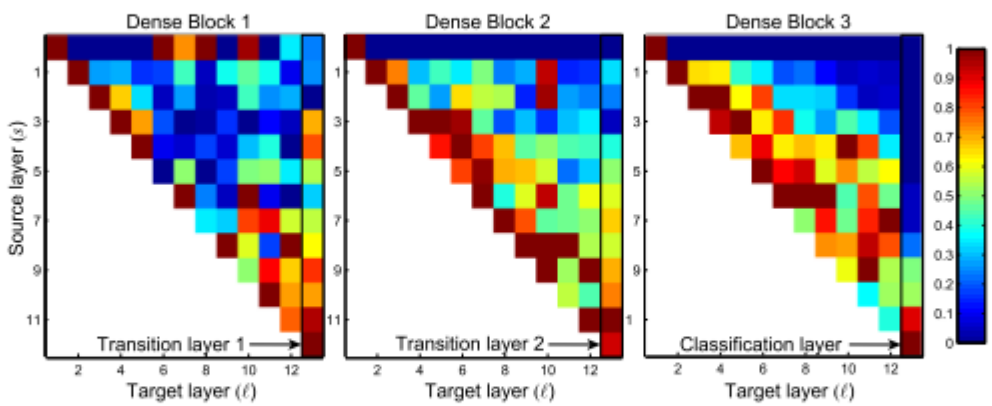
ResNeXt [Xie et al., 2016]

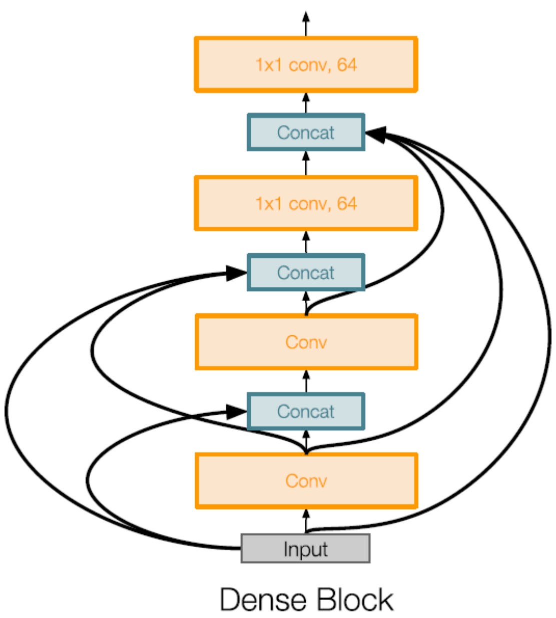
- Aggregating multiple parallel paths inside a residual block ("**cardinality**")
- Increasing cardinality is more effective than going deeper or wider



DenseNet [Huang et al. 2017]

- Passing all the previous representation directly via concatenation of features
- Strengthens **feature propagation** and **feature reuse**





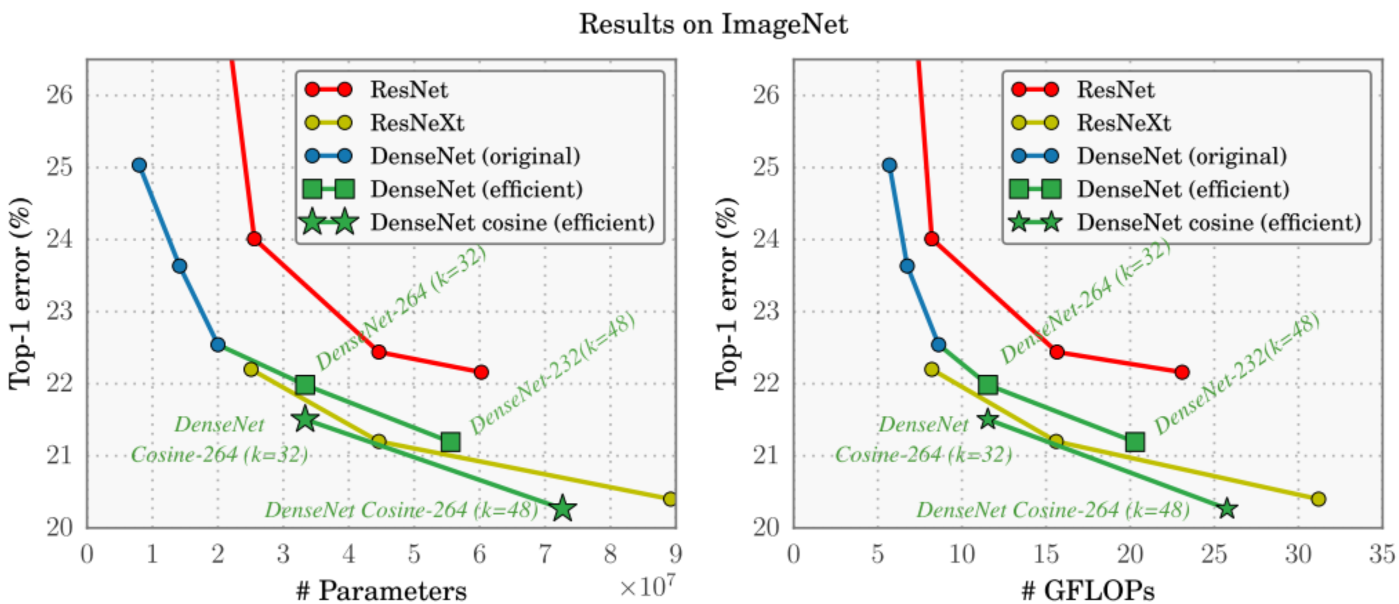
*source : Fei-Fei Li et al. (2018), CS231n Lecture 9, Stanford University

ResNeXt [Xie et al., 2016]

- Aggregating multiple parallel paths inside a residual block ("**cardinality**")
- Increasing cardinality is more effective than going deeper or wider

DenseNet [Huang et al. 2017]

- Passing all the previous representation directly via concatenation of features
- Strengthens **feature propagation** and **feature reuse**



*source : Fei-Fei Li et al. (2018), CS231n Lecture 9, Stanford University

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

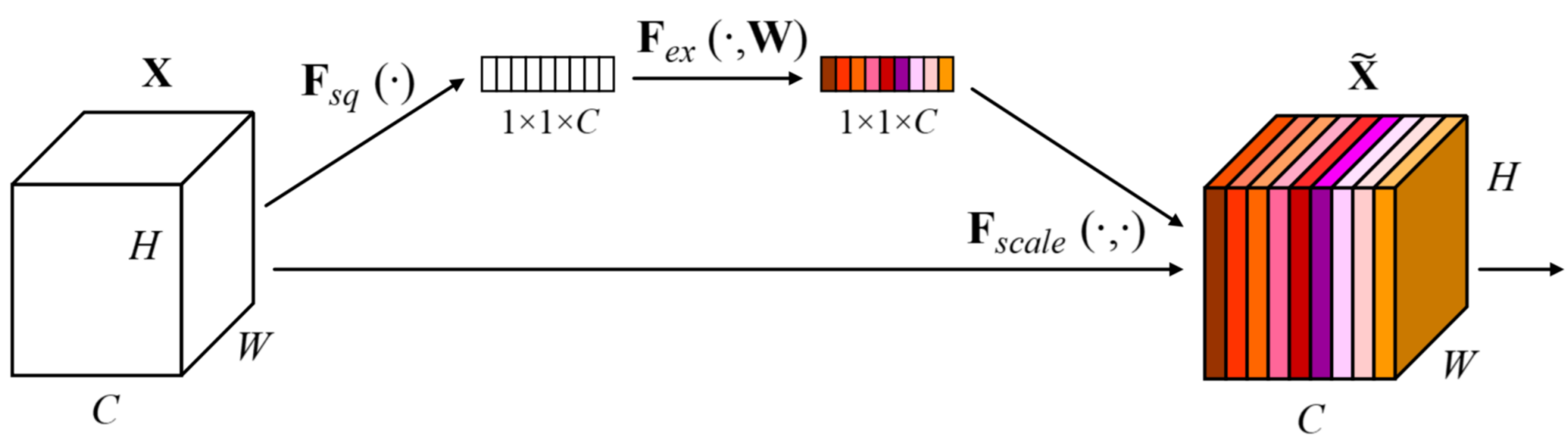
- Patch-based architectures for vision
- New design paradigms

Motivation: The deeper the model, the more feature maps are generated

- Many of them might be important for classification task
- Others might redundant or less important

Squeeze and Excitation Network [Hu et al., 2018]

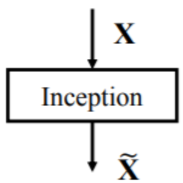
- It selectively emphasizes informative feature maps and suppress less useful ones via global information in two steps
- **Squeeze** step: obtaining global information by shrinking feature maps
  - Global average pooling
- **Excitation** step: recalibrating weights of features by learning channel-wise weights
  - MLP of two fully-connected layers



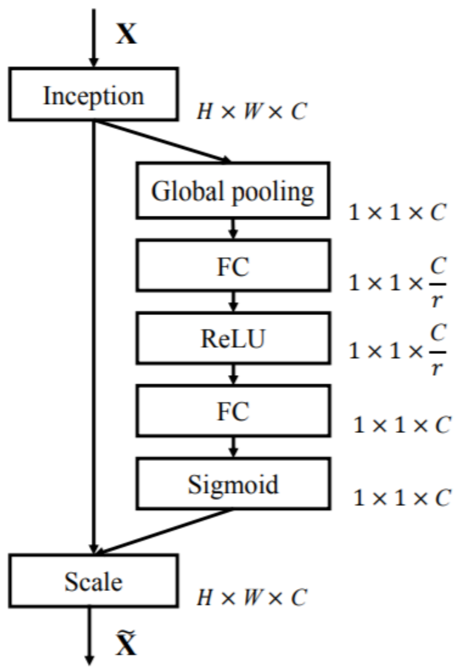
*source: Hu et al., "Squeeze-and-Excitation Networks", CVPR, 2018

Motivation: The deeper the model, the more feature maps are generated
- Many of them might be important for classification task
- Others might redundant or less important

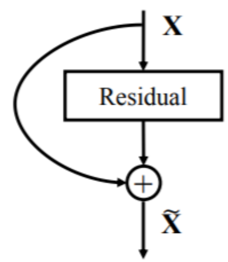
SE block integrates to Inception and ResNet module
- SENet ranked first in the ILSVRC'17 (2.99% → **2.25%**)



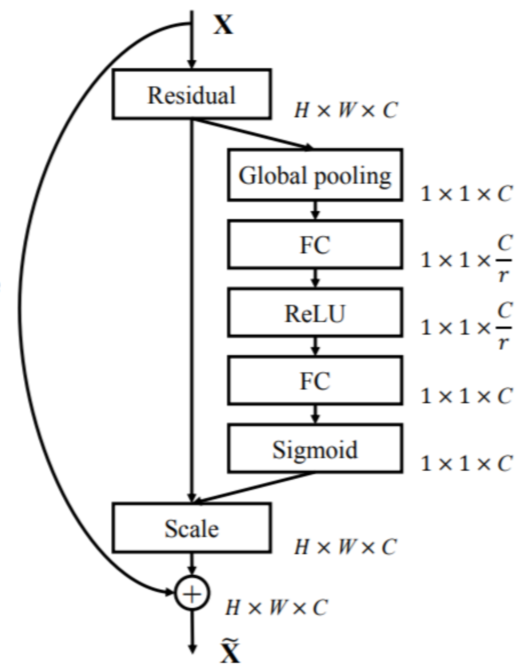
Inception Module



SE-Inception Module



ResNet Module



SE-ResNet Module

*source: Hu et al., "Squeeze-and-Excitation Networks", CVPR, 2018

Motivation: The deeper the model, the more feature maps are generated

- Many of them might be important for classification task
- Others might redundant or less important

SE block integrates to Inception and ResNet module

- SENet ranked first in the ILSVRC'17 (2.99% → **2.25%**)

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [21]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Next, Convolutional Block Attention Module

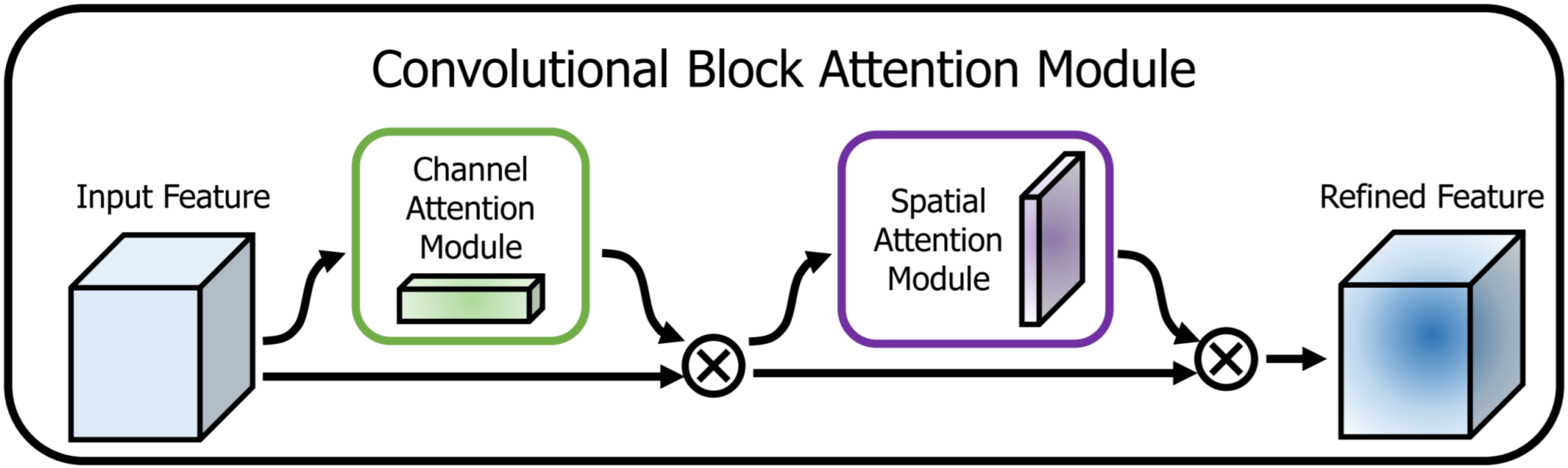
*source: Hu et al., "Squeeze-and-Excitation Networks", CVPR, 2018

Motivation: SENet only considers the contribution of feature maps

- It ignores the spatial locality of the object in image
- The spatial location of the object has a vital role in understanding image

Convolutional Block Attention Module (CBAM) [Woo et al., 2018]

- Learning 'what' and 'where' to attend in the channel and spatial axes respectively
- **Channel** and **Spatial** attention modules



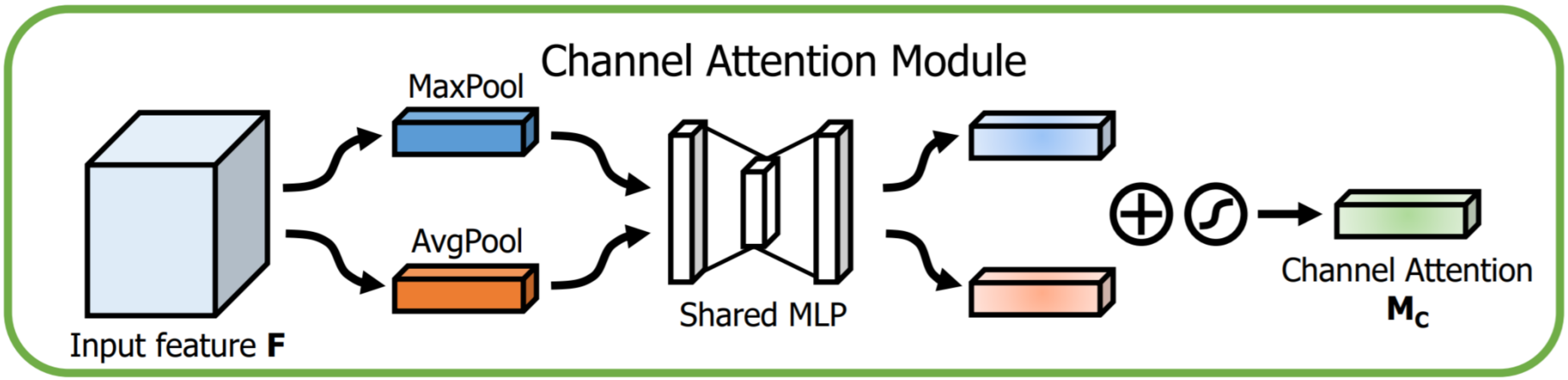
*source: Woo et al., "CBAM: Convolutional block attention module", ECCV, 2018

Motivation: SENet only considers the contribution of feature maps

- It ignores the spatial locality of the object in image
- The spatial location of the object has a vital role in understanding image

Channel attention module: It helps "what" to focus

- Both **average-pooling** and **max-pooling** are important
- **Max-pooling** provides the information of distinctive object features
- Both pooled features share a MLP with two fully-connected layers



$$\mathbf{M_c}(\mathbf{F}) = \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F})))$$

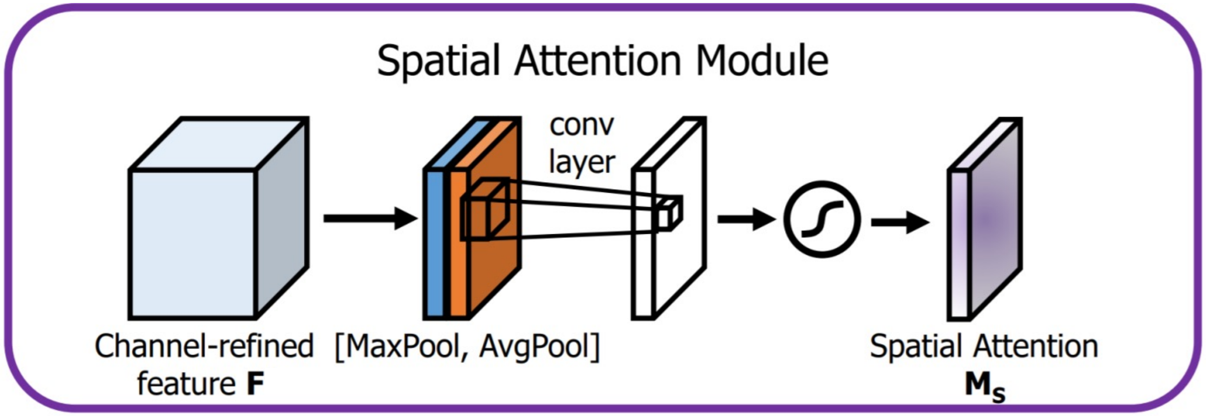
*source: Woo et al., "CBAM: Convolutional block attention module", ECCV, 2018

Motivation: SENet only considers the contribution of feature maps

- It ignores the spatial locality of the object in image
- The spatial location of the object has a vital role in understanding image

Spatial attention module: It helps "where" to focus

- Again, Both **average-pooling** and **max-pooling** are important
- It aggregates channel information of feature maps by using two pooling operations
- Capturing **spatial locality** via convolution



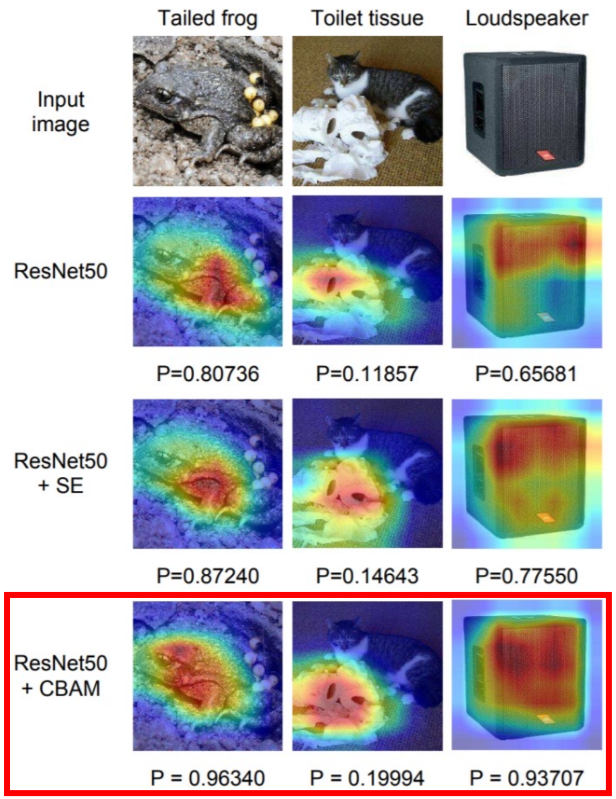
$$\mathbf{M_s}(\mathbf{F}) = \sigma(Conv([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})]))$$

*source: Woo et al., "CBAM: Convolutional block attention module", ECCV, 2018

Motivation: SENet only considers the contribution of feature maps

- It ignores the spatial locality of the object in image
- The spatial location of the object has a vital role in understanding image

- **CBAM** module integrated with ResNet outperforms SE module



Grad-CAM visualization

Architecture	Param.	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
ResNet18 [5]	11.69M	1.814	29.60	10.55
ResNet18 [5] + SE [28]	11.78M	1.814	29.41	10.22
ResNet18 [5] + CBAM	11.78M	1.815	**29.27**	**10.09**
ResNet34 [5]	21.80M	3.664	26.69	8.60
ResNet34 [5] + SE [28]	21.96M	3.664	26.13	8.35
ResNet34 [5] + CBAM	21.96M	3.665	**25.99**	**8.24**
ResNet50 [5]	25.56M	3.858	24.56	7.50
ResNet50 [5] + SE [28]	28.09M	3.860	23.14	6.70
ResNet50 [5] + CBAM	28.09M	3.864	**22.66**	**6.31**
ResNet101 [5]	44.55M	7.570	23.38	6.88
ResNet101 [5] + SE [28]	49.33M	7.575	22.35	6.19
ResNet101 [5] + CBAM	49.33M	7.581	**21.51**	**5.69**
WideResNet18 [6] (widen=1.5)	25.88M	3.866	26.85	8.88
WideResNet18 [6] (widen=1.5) + SE [28]	26.07M	3.867	26.21	8.47
WideResNet18 [6] (widen=1.5) + CBAM	26.08M	3.868	**26.10**	**8.43**
WideResNet18 [6] (widen=2.0)	45.62M	6.696	25.63	8.20
WideResNet18 [6] (widen=2.0) + SE [28]	45.97M	6.696	24.93	7.65
WideResNet18 [6] (widen=2.0) + CBAM	45.97M	6.697	**24.84**	**7.63**
ResNeXt50 [7] (32x4d)	25.03M	3.768	22.85	6.48
ResNeXt50 [7] (32x4d) + SE [28]	27.56M	3.771	**21.91**	6.04
ResNeXt50 [7] (32x4d) + CBAM	27.56M	3.774	21.92	**5.91**
ResNeXt101 [7] (32x4d)	44.18M	7.508	21.54	5.75
ResNeXt101 [7] (32x4d) + SE [28]	48.96M	7.512	21.17	5.66
ResNeXt101 [7] (32x4d) + CBAM	48.96M	7.519	**21.07**	**5.59**

*source: Woo et al., “CBAM: Convolutional block attention module”, ECCV, 2018

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

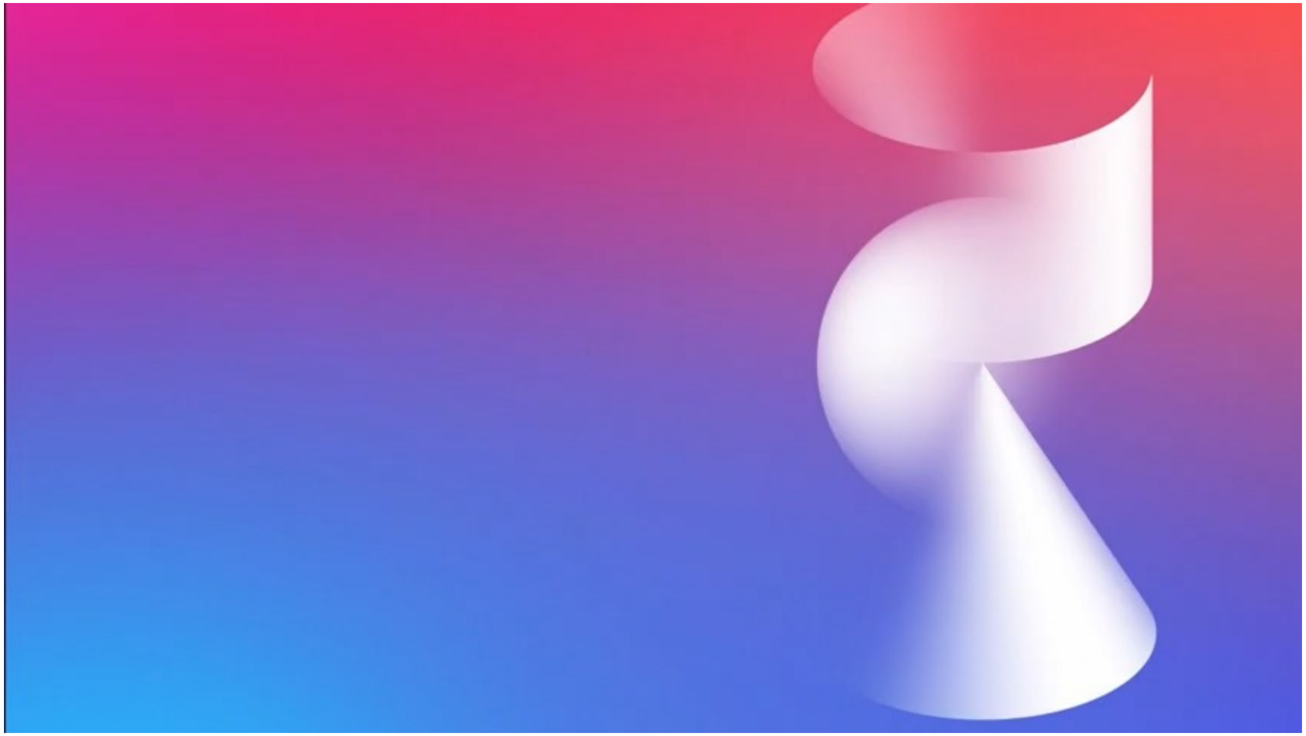
- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Success of Transformer in Language: GPT-3

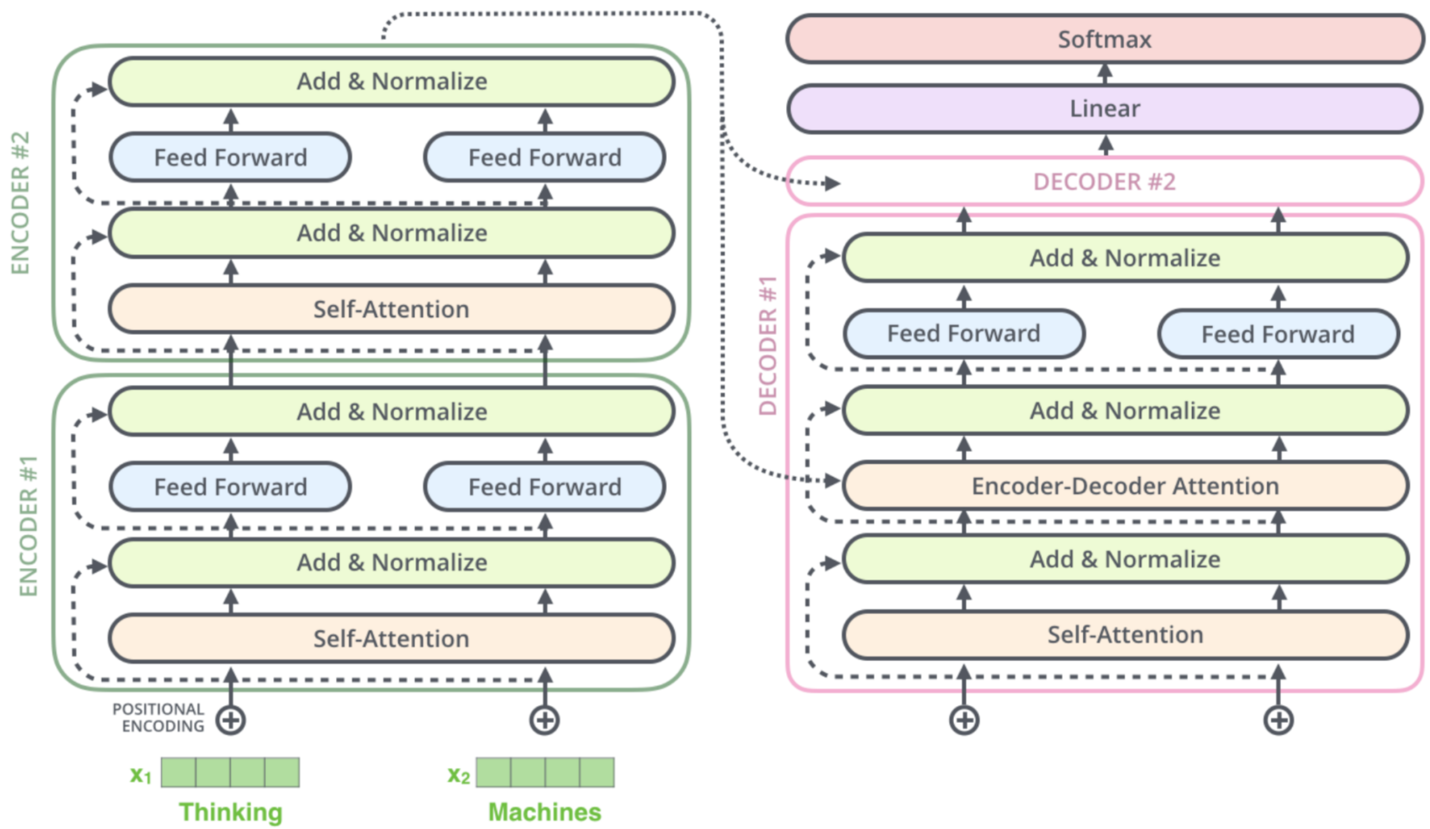
- In 2020, **GPT-3** achieved near-human results in various tasks
- OpenAI even trained a model with **175 billion** parameters (**350 GB** of memory) and showed near-human performance on various **few-shot** tasks



*source : https://youtu.be/CSe3_u9P-RM
Draxler et al., "Essentially no barriers in neural network energy landscape", ICML 2018

What is Transformer?

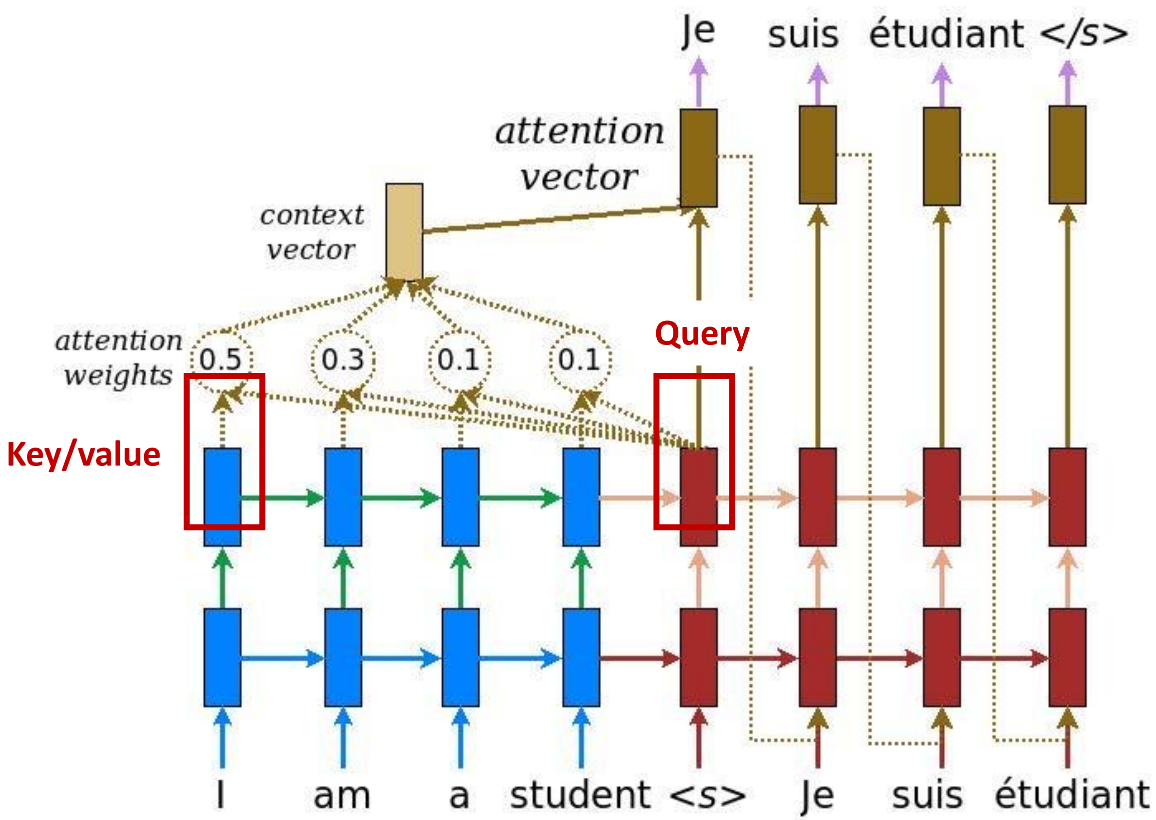
- Transformer [Vaswani et al., 2017] has an **encoder-decoder** structure and they are composed of multiple block with **self-attention** module



*source: http:// http://jalammar.github.io/illustrated-transformer

What is Transformer?

- Transformer [Vaswani et al., 2017] has an **encoder-decoder** structure and they are composed of multiple block with **self-attention** module
- The self-attention is a function of **query** (e.g., "Je") and **key/value** (e.g., "I")
  - It shows powerful performances in learning **sequential input-output relations**



Attention mechanism can be used for other type of input data, e.g. image

- Self-attention operation scales **quadratically** with the sequence length

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Question: How to transform an image to **sequence data**?

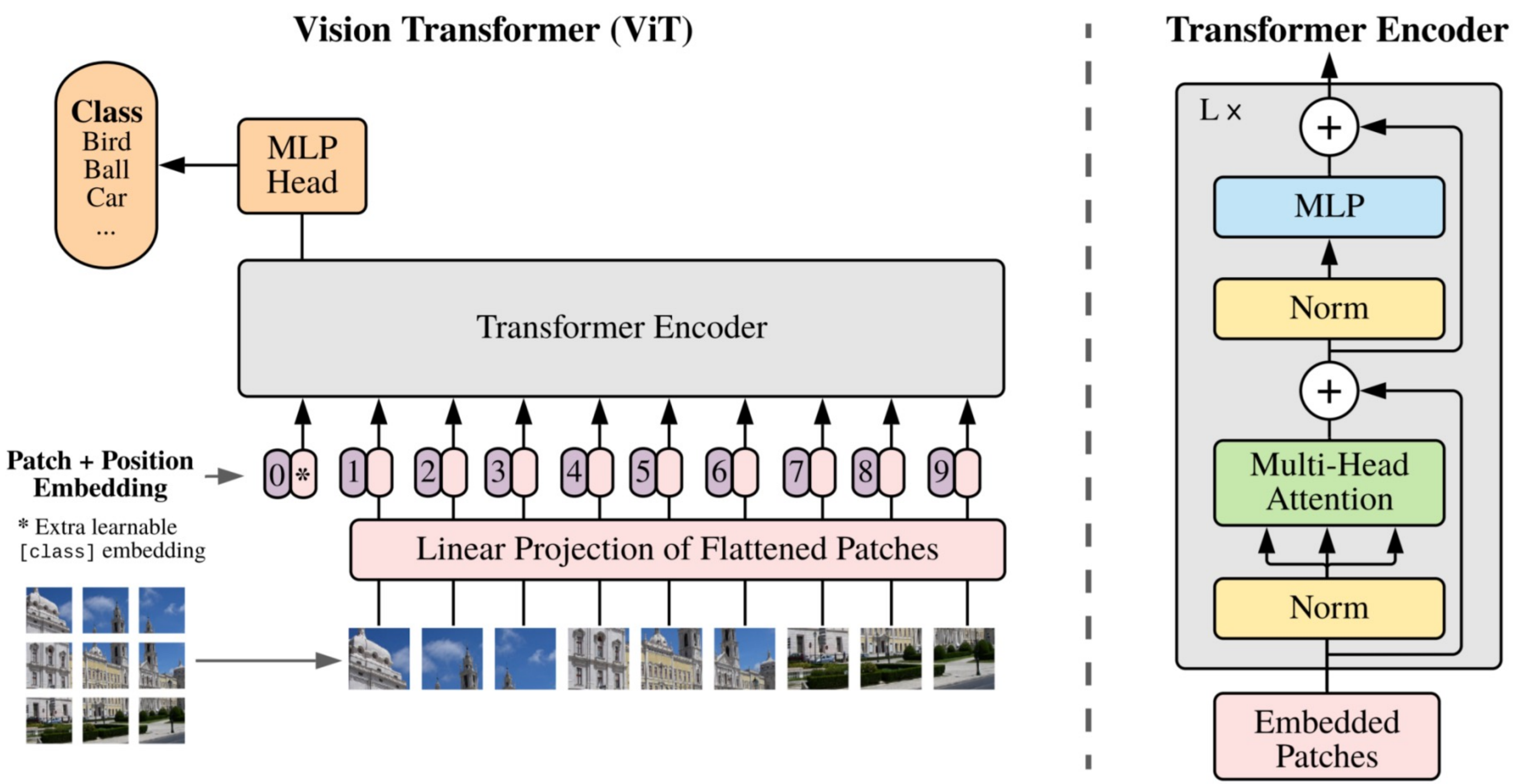
- Dosovitskiy et al. (2021): splits an **image** into **patches**

Sequence of patch images

*source: [Chen et al. 2020] Generative Pretraining from Pixels, ICML 2020
[Dosovitskiy et al. 2021] An image is worth 16x16 words: Transformers for image recognition at scale, ICLR 2021

Vision Transformer [Dosovitskiy et al., 2021]

- Splitting an image into fixed-size patches (16x16)
    - Linearly embedding each of them
- Adding position embedding & [class] token



*source: [Dosovitskiy et al. 2021] An image is worth 16x16 words: Transformers for image recognition at scale, ICLR 2021

Vision Transformer [Dosovitskiy et al., 2021]

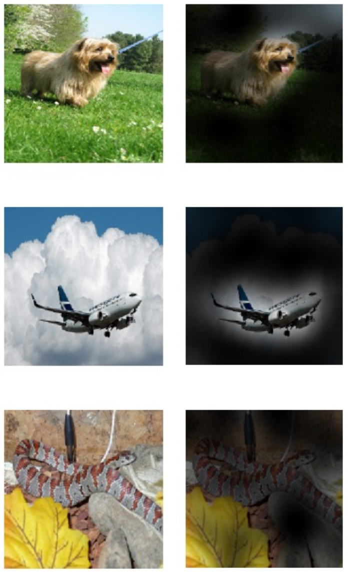
- Splitting an image into fixed-size patches (16x16)
  - Linearly embedding each of them
- Adding position embedding & [class] token

- **Dosovitskiy et al.** (2021) pre-trains models on larger datasets (14M-300M images)
  - Vision Transformer achieves **competitive performances** compared to CNNs

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Vision Transformer **CNNs**

Input Attention



*source: [Dosovitskiy et al. 2021] An image is worth 16x16 words: Transformers for image recognition at scale, ICLR 2021

Various architectures now are based on Vision Transformer

1. **Modification for patch splitting**
    - Token-to-Token Vision Transformer [Li et al., 2021]
    - Swin Transformer [Liu et al., 2021]
2. **Modification for hierarchical structure**
    - Pooling-based Vision Transformer [Heo et al., 2021]
    - Swin Transformer [Liu et al., 2021]

Question: What's a good way to split an **image** into a **sequence of patches**?

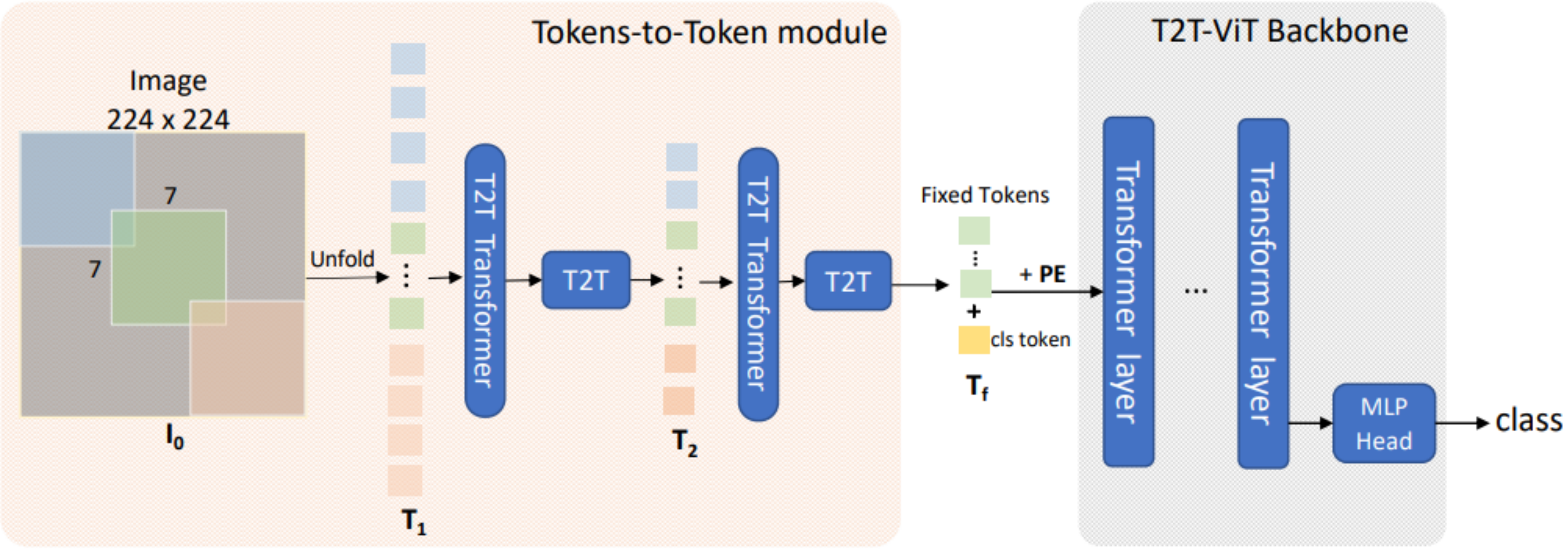
- Vision Transformer splits an image into a **fixed grid-shape** of **non-overlapping** patches

Sequence of patch images

*source : He et al., "Deep residual learning for image recognition". CVPR 2016

Token-to-Token Vision Transformer [Li et al., 2021]

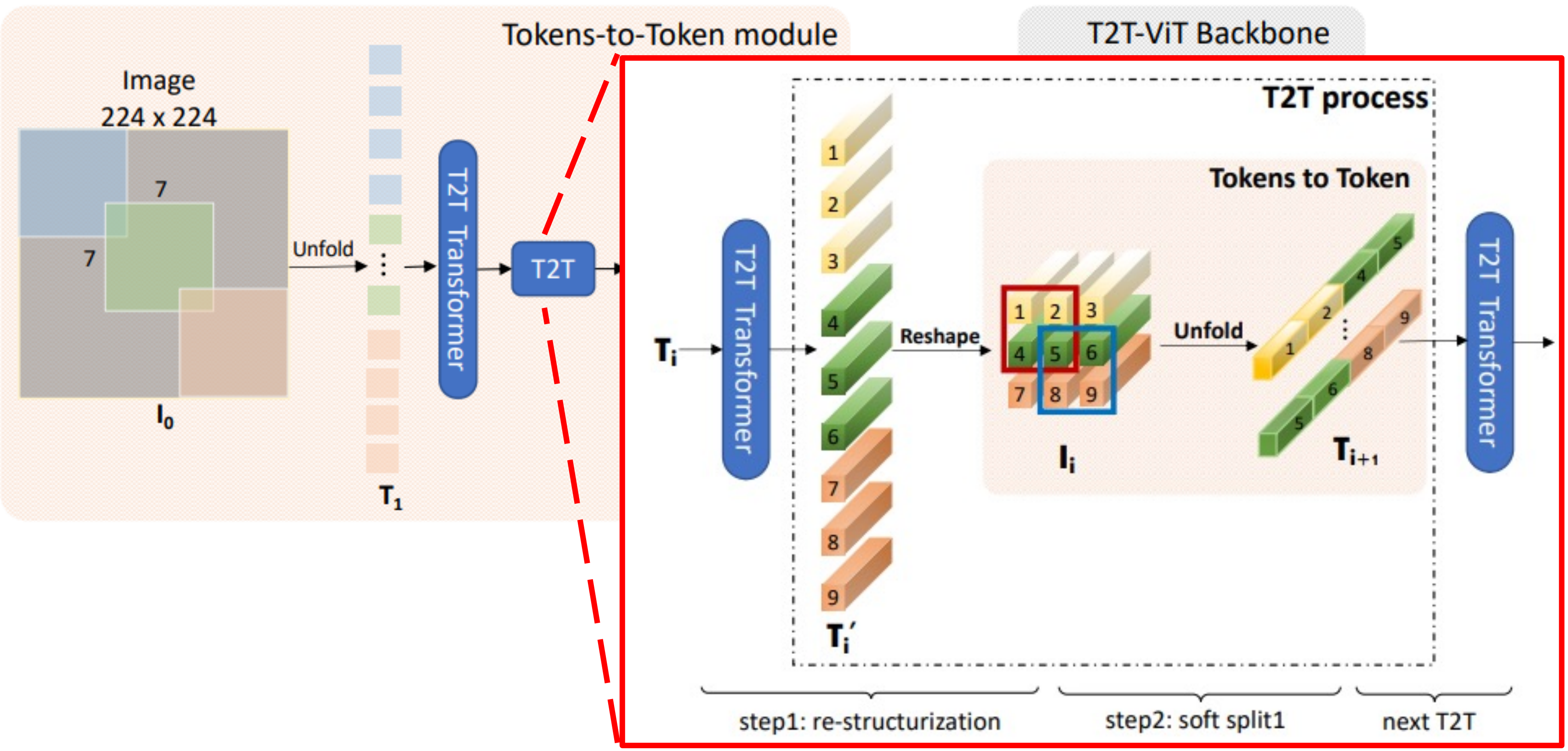
- **(Soft-split)** Splitting an image into overlapping patches
- **(Re-structurization)** Rearranging patch sequences into 2D image shape
- Iterating re-structurization and soft-split before Transformer backbone



*source: [Li et al. 2021] Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet, ICCV 2021

Token-to-Token Vision Transformer [Li et al., 2021]

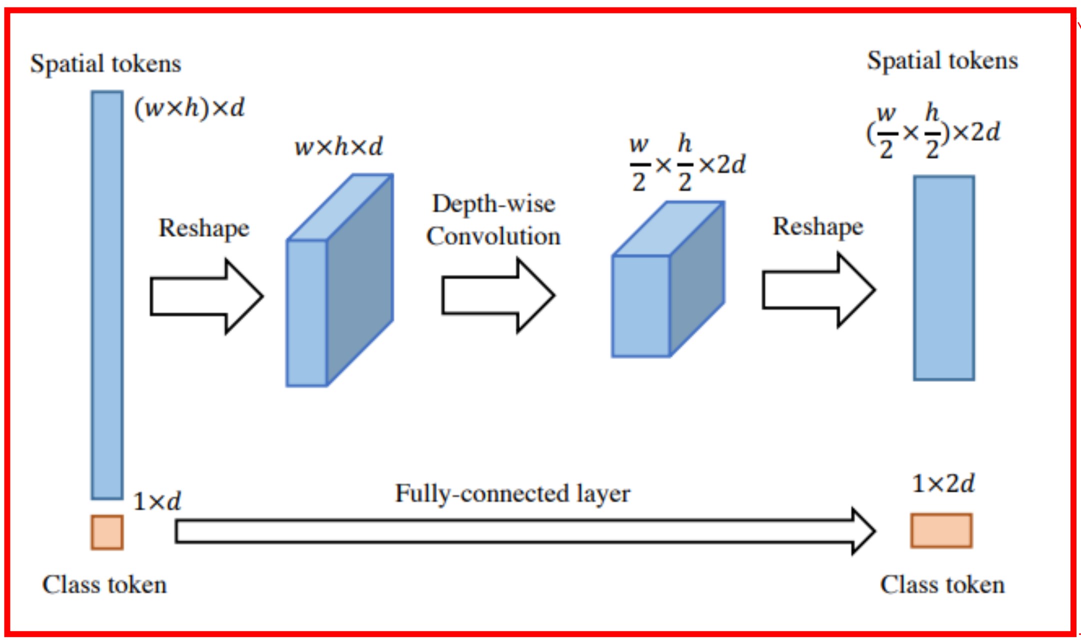
- **(Soft-split)** Splitting an image into overlapping patches
- **(Re-structurization)** Rearranging patch sequences into 2D image shape
- Iterating re-structurization and soft-split before Transformer backbone

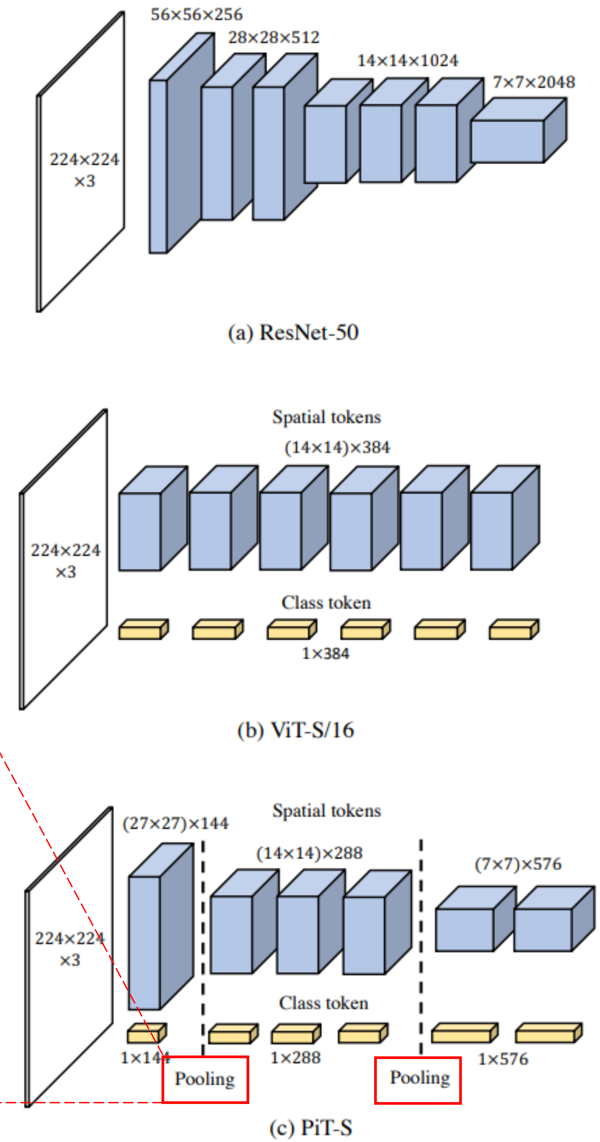


*source: [Li et al. 2021] Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet, ICCV 2021

Pooling-based Vision Transformer [Heo et al., 2021]

- Design of a hierarchical structure
    - **Motivation:** ResNet gradually downsamples the features from the input to the output
- Downsampling via the pooling layer based on depth-wise convolution
- Spatial reduction with small parameters



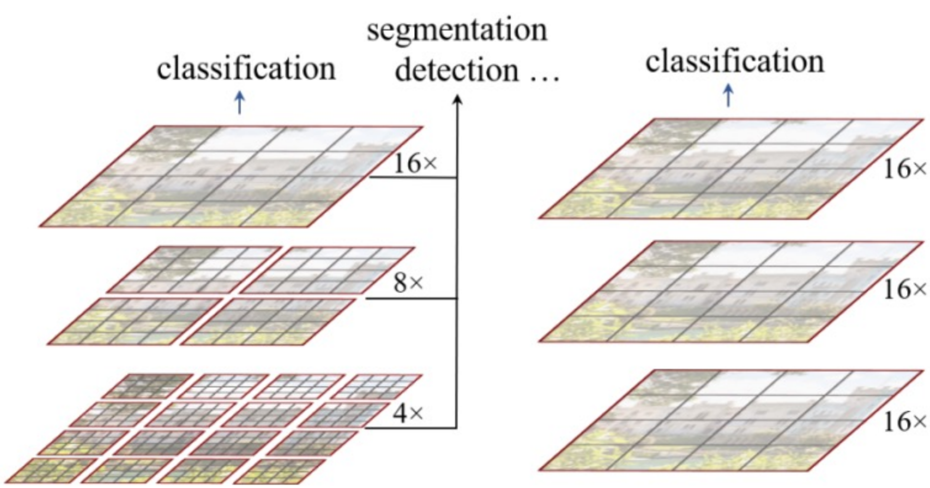


*source: [Heo et al. 2021] Rethinking Spatial Dimensions of Vision Transformers , ICCV 2021

Swin Transformer [Liu et al., 2021]

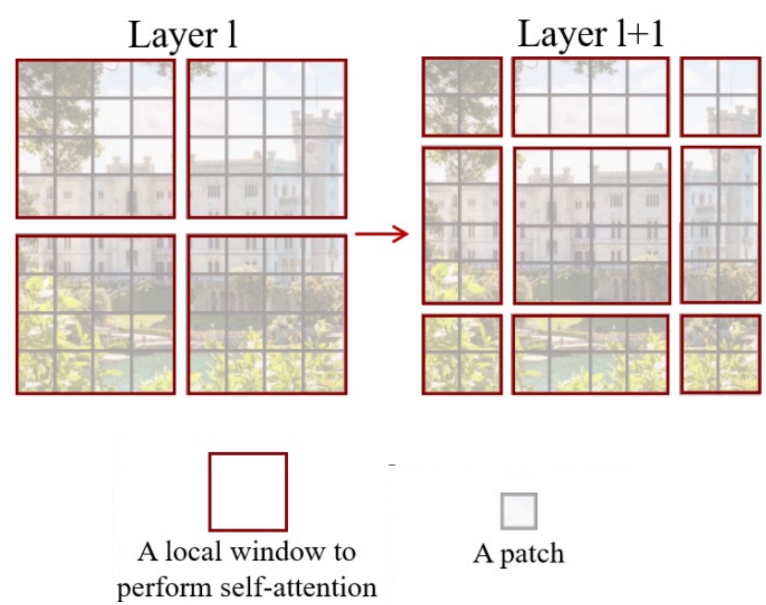
- Design of a hierarchical structure
- Various spatial resolutions (e.g., patch-shape) can be handled via shifted windows
- Efficient self-attention computation by using shifted windows scheme
- Concatenating 2 × 2 neighboring patches for downsampling operation
- Powerful performances in dense prediction tasks

  e.g., object detection and semantic segmentation



(a) Swin Transformer (ours)   (b) ViT

Shifted window scheme



*source: [Liu et al. 2021] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021

DeiT III [Touvron et al., 2022]

Question: Do vision transformers need some inductive bias under small data?

- Vision transformers achieved state-of-the-art performances but…
  - Required gigantic-scale training with JFT-300M data
  - Sub-optimal performance under the ImageNet-scale training
- Injecting some inductive bias (e.g., Swin, PiT) was needed for ImageNet-scale

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

*source: Dai et al., "Deformable Convolutional Networks", ICCV, 2017

Motivation: Do vision transformers need some inductive bias under small data?

- Vision transformers achieved state-of-the-art performances but…
  - Required gigantic-scale training with JFT-300M data
  - Sub-optimal performance under the ImageNet-scale training
- Injecting some inductive bias (e.g., Swin, PiT) was needed for ImageNet-scale

DeiT III [Touvron et al., 2022] finds that vanilla vision transformer can outperform CNNs in ImageNet-scale:

- The problem was in the **sub-optimal optimization designs**
  - LayerScale
  - Improved data augmentations

  could solve the optimization issues

  Check the paper for details!

	Previous approaches				Ours		
Procedure → Reference	ViT [13]	Steiner et al. [42]	DeiT [48]	Wightman et al. [57]	ImNet-1k	ImNet-21k Pretrain.	ImNet-21k Finetune.
Batch size	4096	4096	1024	2048	2048	2048	2048
Optimizer	AdamW	AdamW	AdamW	LAMB	LAMB	LAMB	LAMB
LR	3.10^{-3}	3.10^{-3}	1.10^{-3}	5.10^{-3}	3.10^{-3}	3.10^{-3}	3.10^{-4}
LR decay	cosine	cosine	cosine	cosine	cosine	cosine	cosine
Weight decay	0.1	0.3	0.05	0.02	0.02	0.02	0.02
Warmup epochs	3.4	3.4	5	5	5	5	5
Label smoothing ε	0.1	0.1	0.1	✗	✗	0.1	0.1
Dropout	✓	✓	✗	✗	✗	✗	✗
Stoch. Depth	✗	✓	✓	✓	✓	✓	✓
Repeated Aug	✗	✗	✓	✓	✓	✗	✗
Gradient Clip.	1.0	1.0	✗	1.0	1.0	1.0	1.0
H. flip	✓	✓	✓	✓	✓	✓	✓
RRC	✓	✓	✓	✓	✓	✗	✗
Rand Augment	✗	Adapt.	9/0.5	7/0.5	✗	✗	✗
3 Augment (ours)	✗	✗	✗	✗	✓	✓	✓
LayerScale	✗	✗	✗	✗	✓	✓	✓
Mixup alpha	✗	Adapt.	0.8	0.2	0.8	✗	✗
Cutmix alpha	✗	✗	1.0	1.0	1.0	1.0	1.0
Erasing prob.	✗	✗	0.25	✗	✗	✗	✗
ColorJitter	✗	✗	✗	✗	0.3	0.3	0.3
Test crop ratio	0.875	0.875	0.875	0.95	1.0	1.0	1.0
Loss	CE	CE	CE	BCE	BCE	CE	CE

*source: Dai et al., "Deformable Convolutional Networks", ICCV, 2017

Motivation: Do vision transformers need some inductive bias under small data?

- Vision transformers achieved state-of-the-art performances but…
    - Required gigantic-scale training with JFT-300M data
    - Sub-optimal performance under the ImageNet-scale training
- Injecting some inductive bias (e.g., Swin, PiT) was needed for ImageNet-scale

DeiT III [Touvron et al., 2022] finds that vanilla vision transformer can outperform CNNs in ImageNet-scale:

Architecture	nb params ($\times 10^6$)	throughput (im/s)	FLOPs ($\times 10^9$)	Peak Mem (MB)	Top-1 Acc.	V2 Acc.
"Traditional" ConvNets						
R-101x3↑384 [25]	388	_	204.6	_	84.4	_
R-152x4↑480 [25]	937	_	840.5	_	85.4	_
EfficientNetV2-S↑384 [45]	21.5	874	8.5	4515	84.9	74.5
EfficientNetV2-M↑480 [45]	54.1	312	25.0	7127	86.2	75.9
EfficientNetV2-L↑480 [45]	118.5	179	53.0	9540	86.8	76.9
EfficientNetV2-XL↑512 [45]	208.1	_	94.0	_	87.3	77.0
Patch-based ConvNets						
ConvNeXt-B [32]	88.6	563	15.4	3029	85.8	75.6
ConvNeXt-B↑384 [32]	88.6	190	45.1	7851	86.8	76.6
ConvNeXt-L [32]	197.8	344	34.4	4865	86.6	76.6
ConvNeXt-L↑384 [32]	197.8	115	101	11938	87.5	77.7
ConvNeXt-XL [32]	350.2	241	60.9	6951	87.0	77.0
ConvNeXt-XL↑384 [32]	350.2	80	179.0	16260	87.8	77.7
Vision Transformers derivative						
Swin-B [31]	87.8	532	15.4	4695	85.2	74.6
Swin-B↑384 [31]	87.9	160	47.0	19385	86.4	76.3
Swin-L [31]	196.5	337	34.5	7350	86.3	76.3
Swin-L↑384 [31]	196.7	100	103.9	33456	87.3	77.0
Vanilla Vision Transformers						
ViT-B/16 [42]	86.6	831	17.6	2078	84.0	_
ViT-B/16↑384 [42]	86.7	190	55.5	8956	85.5	_
ViT-L/16 [42]	304.4	277	61.6	3789	84.0	_
ViT-L/16↑384 [42]	304.8	67	191.1	12866	85.5	_
Our Vanilla Vision Transformers						
ViT-S	22.0	1891	4.6	987	83.1	73.8
ViT-B	86.6	831	17.6	2078	85.7	76.5
ViT-B↑384	86.9	190	55.5	8956	86.7	77.9
ViT-L	304.4	277	61.6	3789	87.0	78.6
ViT-L↑384	304.8	67	191.2	12866	87.7	79.1
ViT-H	632.1	112	167.4	6984	87.2	79.2

*source: Dai et al., "Deformable Convolutional Networks", ICCV, 2017

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

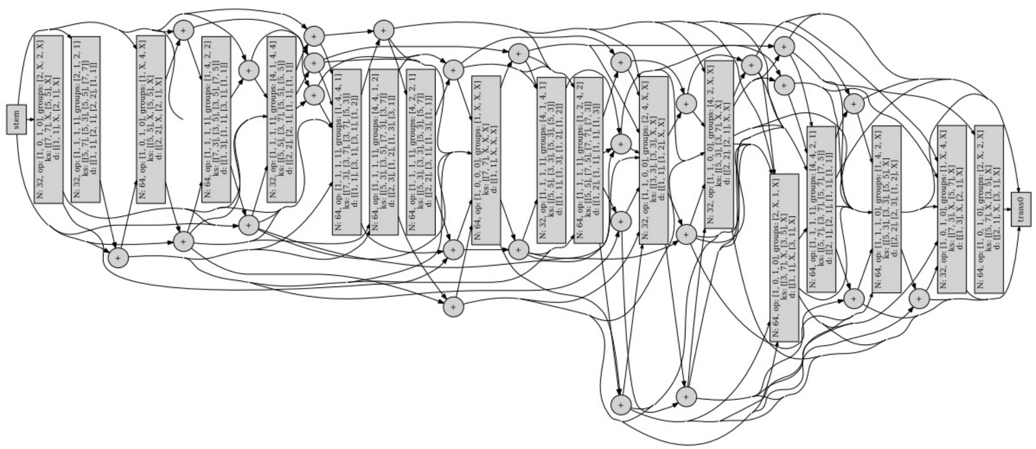
- Patch-based architectures for vision
- New design paradigms

Although the CNN architecture has evolved greatly, our **design principles are still relying on heuristics**

- Smaller kernel and smaller stride, increase cardinality instead of width ...

Recently, there have been works on automatically finding a structure which can outperform existing human-crafted architectures

1. **Search space**: Naïvely searching every model is nearly impossible
2. **Searching algorithm**: Evaluating each model is very costly, and black-boxed



A sample architecture found in [Brock et al., 2018]

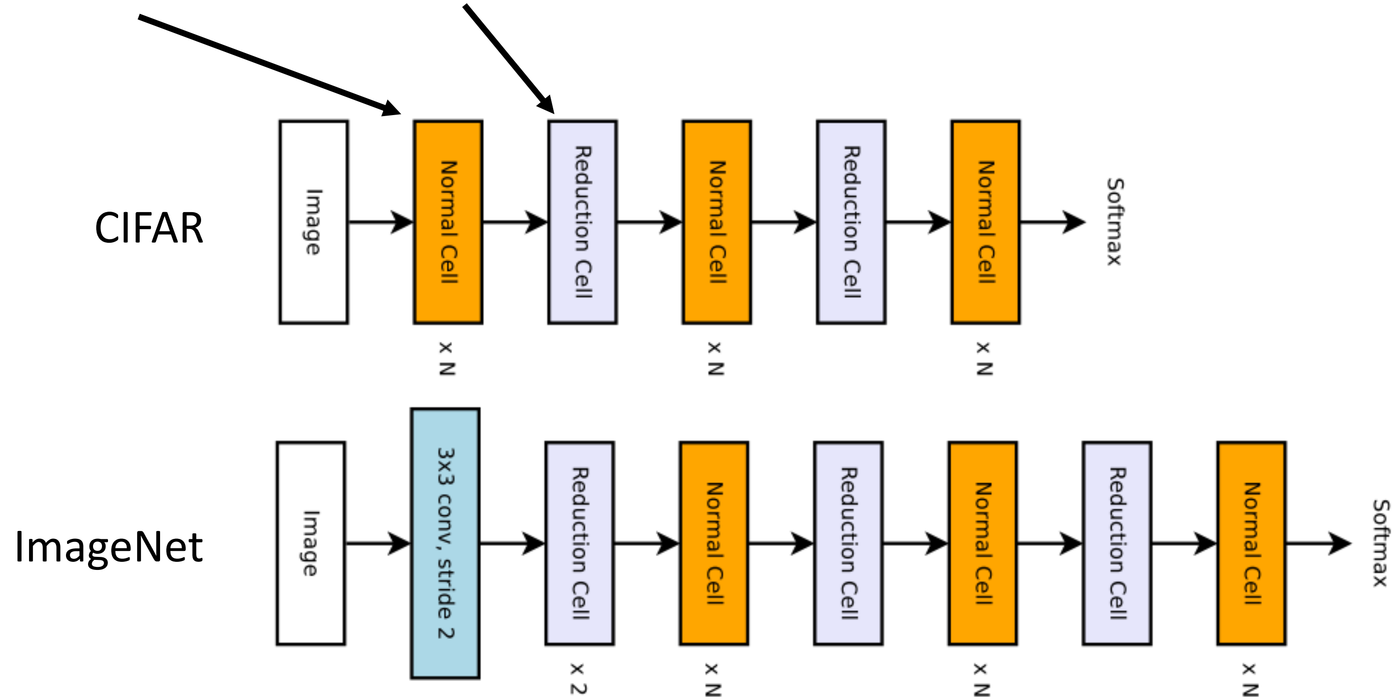
*source : Brock et al., “SMASH: One-Shot Model Architecture Search through HyperNetworks”, ICLR 2018

Designing a good search space is important in architecture searching

- **NASNet** reduces the search space by incorporating our design principles

Motivation: modern architectures are built simply: **a repeated modules**

- Try not to search the whole model, but only cells modules
- **Normal cell** and **Reduction cell** (cell w/ stride 2)

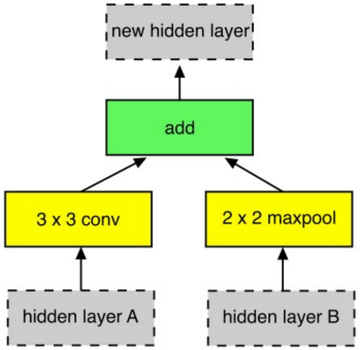


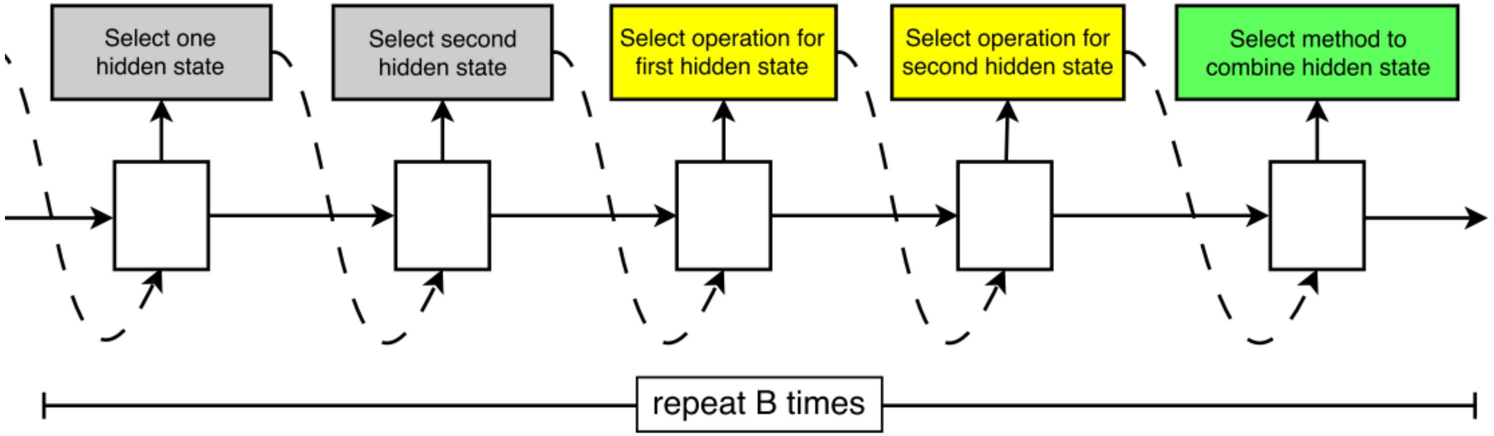
*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

Designing a good search space is important in architecture searching

- **NASNet** reduces the search space by incorporating our design principles

- Each cell consists of B **blocks**
- Each block is determined by **selecting methods**
    1. Select two hidden states from h_i, h_{i-1} or of existing block
    2. Select methods to process for each of the selected states
    3. Select a method to combine the two states
        - (1) **element-wise addition** or (2) **concatenation**



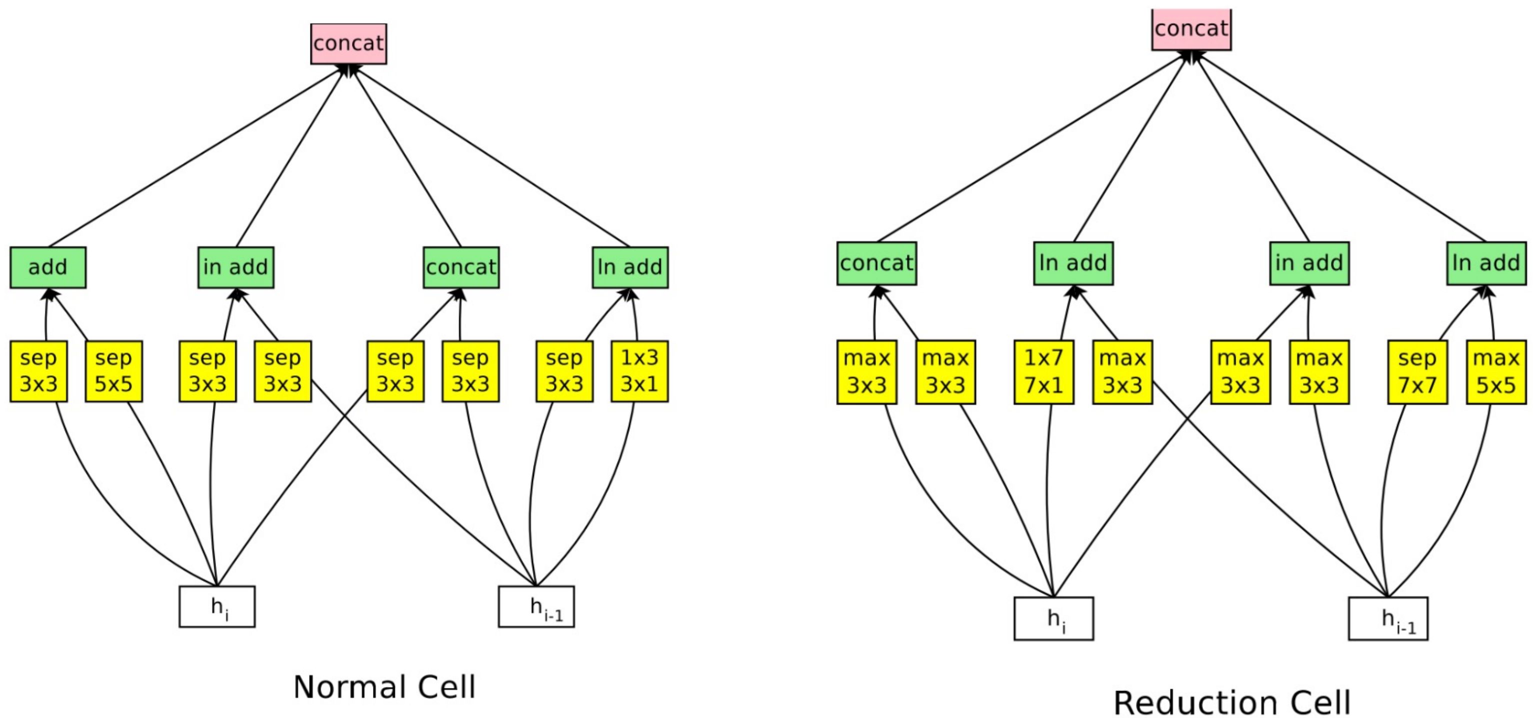


*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

Designing a good search space is important in architecture searching

- **NASNet** reduces the search space by incorporating our design principles

- Each cell consists of B **blocks**
    - **Example**: $B = 4$



*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

Designing a good search space is important in architecture searching

- **NASNet** reduces the search space by incorporating our design principles

- Set of methods to be selected based on their prevalence in the CNN literature

  - identity
  - 1x7 then 7x1 convolution
  - 3x3 average pooling
  - 5x5 max pooling
  - 1x1 convolution
  - 3x3 depthwise-separable conv
  - 7x7 depthwise-separable conv
  - 1x3 then 3x1 convolution
  - 3x3 dilated convolution
  - 3x3 max pooling
  - 7x7 max pooling
  - 3x3 convolution
  - 5x5 depthwise-seperable conv

Any searching methods can be used

- **Random search** [Bergstra et al., 2012] could also work
- **RL-based search** [Zoph et al., 2016] is mainly used in this paper

*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

Toward Automation of Network Design: NASNet [Zoph et al., 2018]

- The pool of workers consisted of **500 GPUs**, processing **over 4 days**

All architecture searches are performed on **CIFAR-10**

- NASNet-A: State-of-the-art error rates could be achieved
- NASNet-B/C: Extremely parameter-efficient models were also found

model	depth	# params	error rate (%)
DenseNet ($L = 40, k = 12$) [26]	40	1.0M	5.24
DenseNet($L = 100, k = 12$) [26]	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) [26]	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) [26]	190	25.6M	3.46
Shake-Shake 26 2x32d [18]	26	2.9M	3.55
Shake-Shake 26 2x96d [18]	26	26.2M	2.86
Shake-Shake 26 2x96d + cutout [12]	26	26.2M	2.56
NAS v3 [70]	39	7.1M	4.47
NAS v3 [70]	39	37.4M	3.65
NASNet-A (6 @ 768)	-	3.3M	3.41
NASNet-A (6 @ 768) + cutout	-	3.3M	2.65
NASNet-A (7 @ 2304)	-	27.6M	2.97
NASNet-A (7 @ 2304) + cutout	-	27.6M	2.40
NASNet-B (4 @ 1152)	-	2.6M	3.73
NASNet-C (4 @ 640)	-	3.1M	3.59

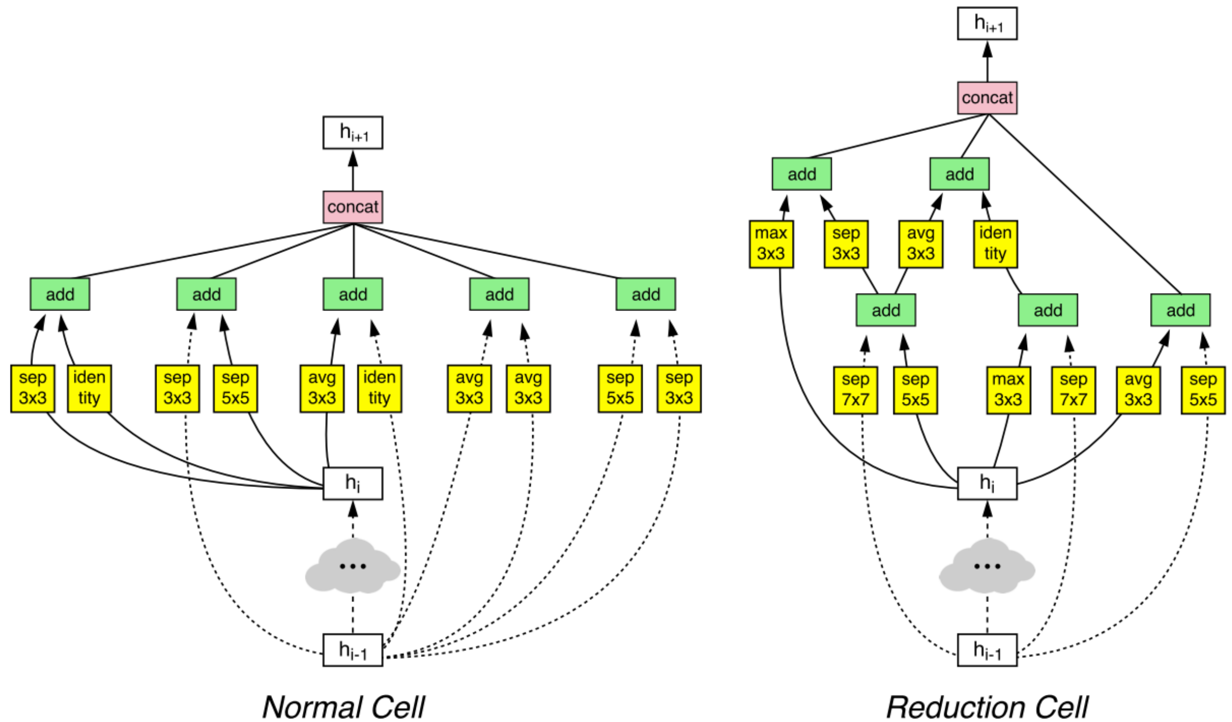
*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

Toward Automation of Network Design: NASNet [Zoph et al., 2018]

- The pool of workers consisted of **500 GPUs**, processing **over 4 days**

All architecture searches are performed on **CIFAR-10**

- NASNet-A: State-of-the-art error rates could be achieved
- NASNet-B/C: Extremely parameter-efficient models were also found



NASNet-A

*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

- The pool of workers consisted of **500 GPUs**, processing **over 4 days**

All architecture searches are performed on **CIFAR-10**

Cells found in CIFAR-10 could also transferred well into ImageNet

Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [59]	299×299	23.8 M	5.72 B	78.0	93.9
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [57]	299×299	55.8 M	13.2 B	80.4	95.3
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [67]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [68]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

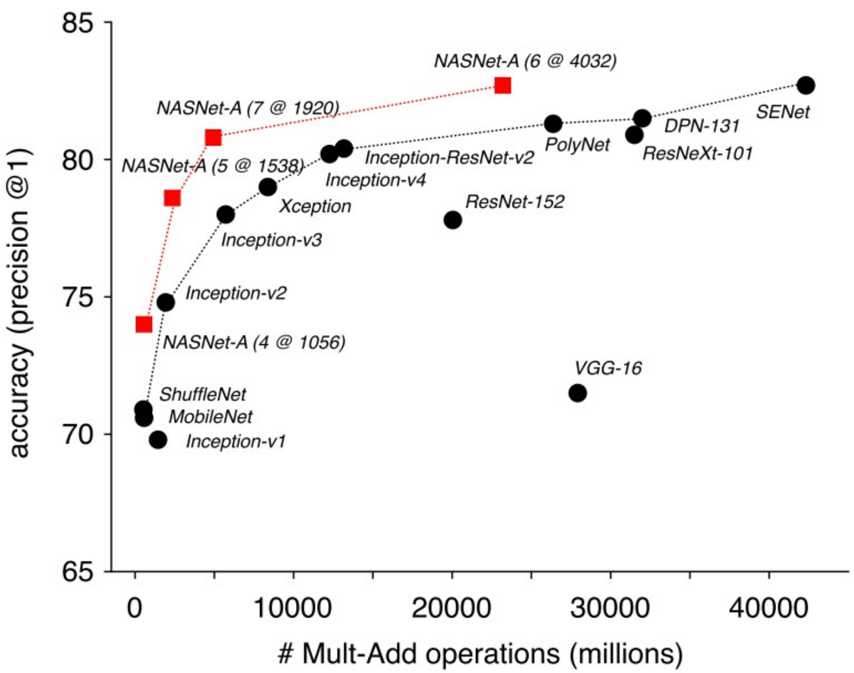
*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

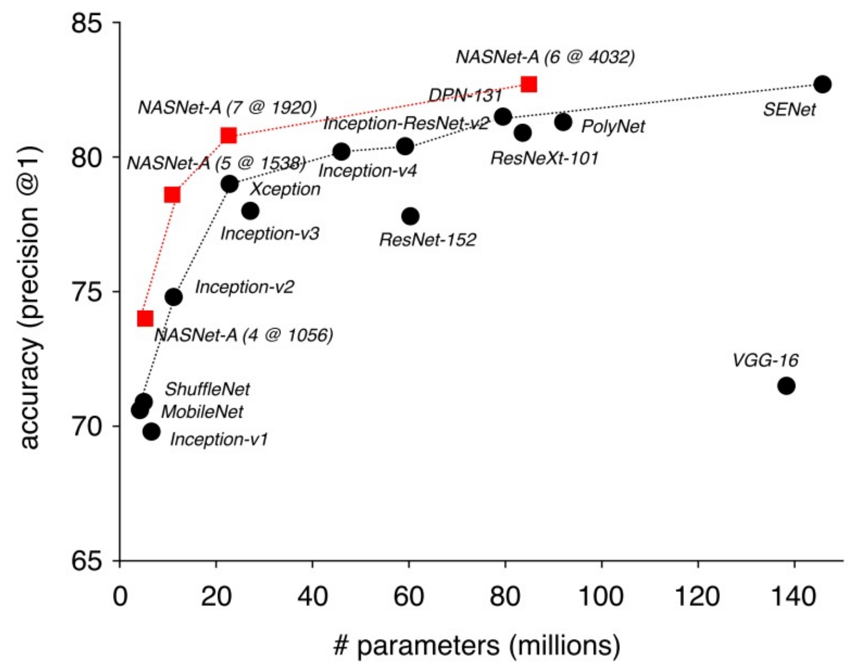
Toward Automation of Network Design: NASNet [Zoph et al., 2018]

- The pool of workers consisted of **500 GPUs**, processing **over 4 days**

All architecture searches are performed on **CIFAR-10**

Cells found in CIFAR-10 could also transferred well into ImageNet



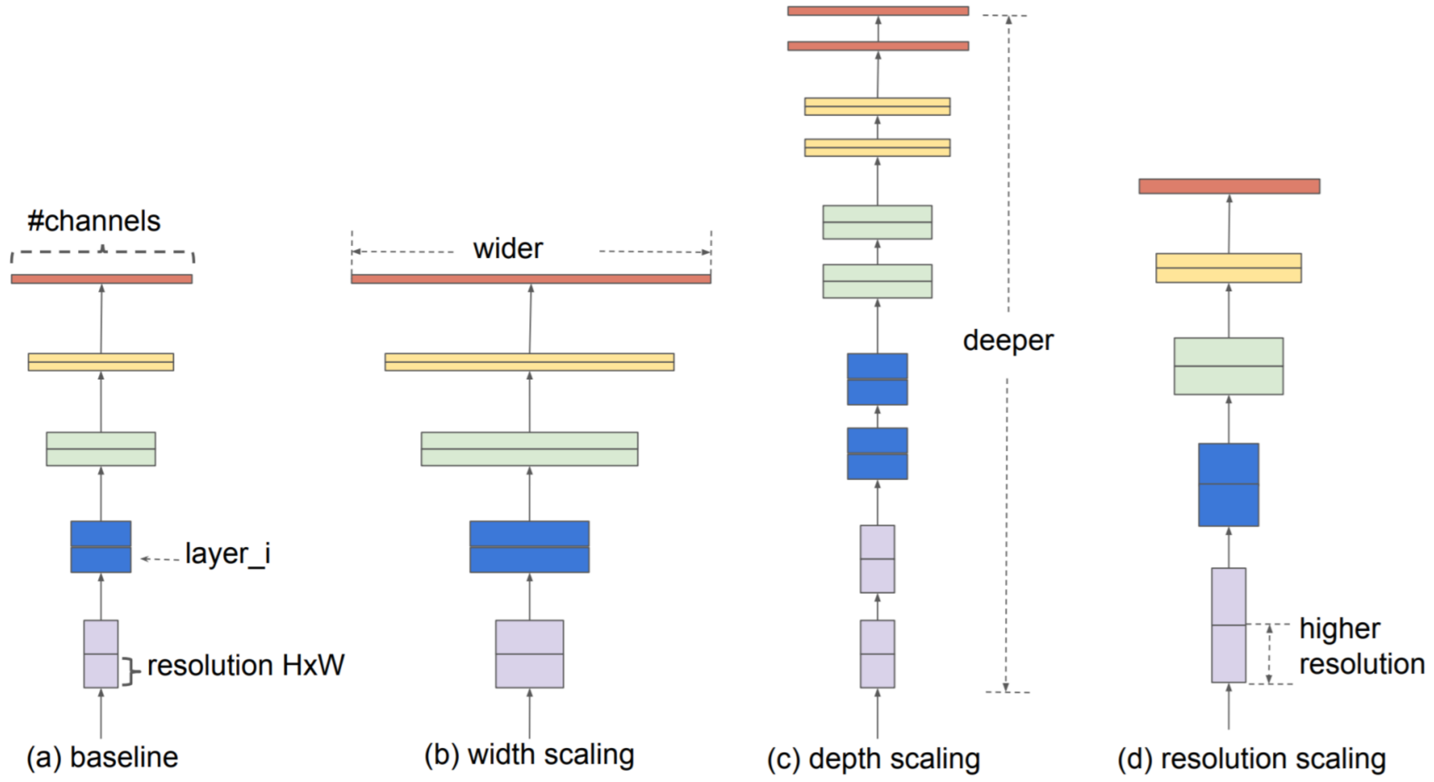


*source : Zoph et al., “Learning Transferable Architectures for Scalable Image Recognition”, CVPR 2018

Although **Scaling up** CNNs is widely used to achieve better generalization, the process of scaling has never been understood

- The common way is scaling model depth, width, and image resolution

Question: Is there a principled scaling method for better accuracy and efficiency?



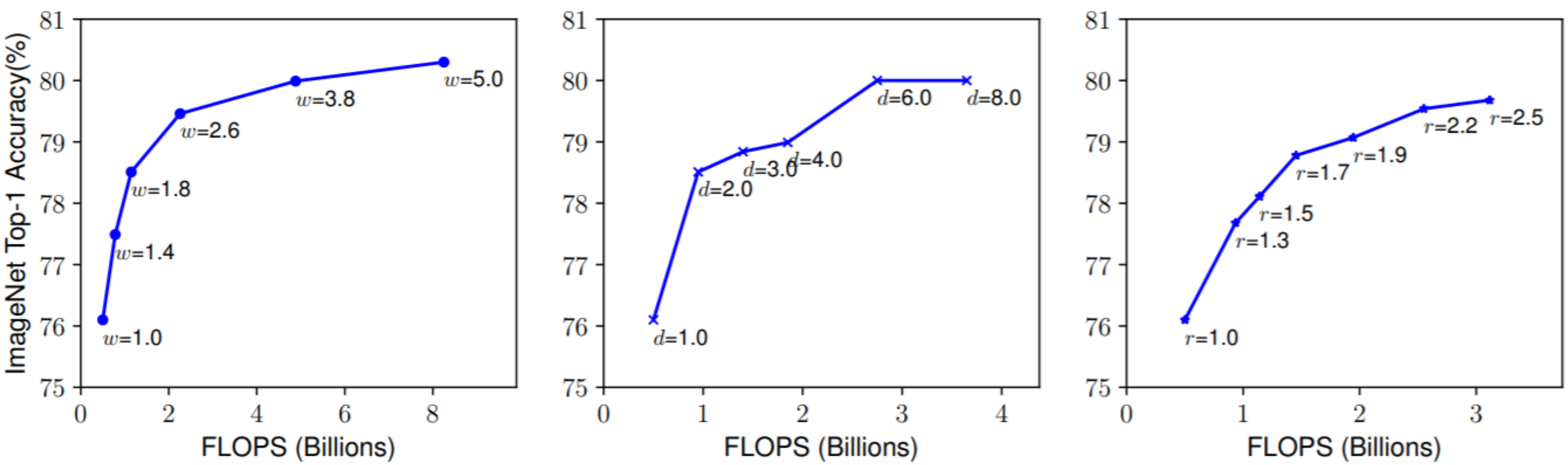
*source : Tan et al., “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, ICML 2019

The state-of-the-art ILSVRC classification in 2019 (top-5 error rate 2.9%)

- **EfficientNet** uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients (called ***"compound scaling"***)

Motivation: There exists certain relationship between network width, depth and image resolution

- Scaling single dimension has a limitation
  - Gain diminishes for bigger models.

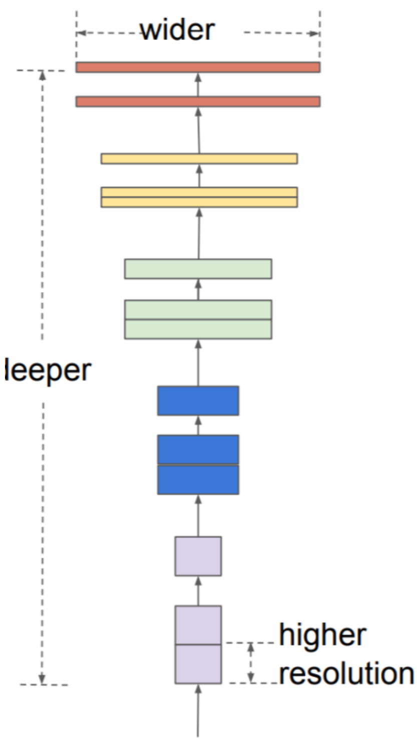


Depth w **Width d** **Resolution r**

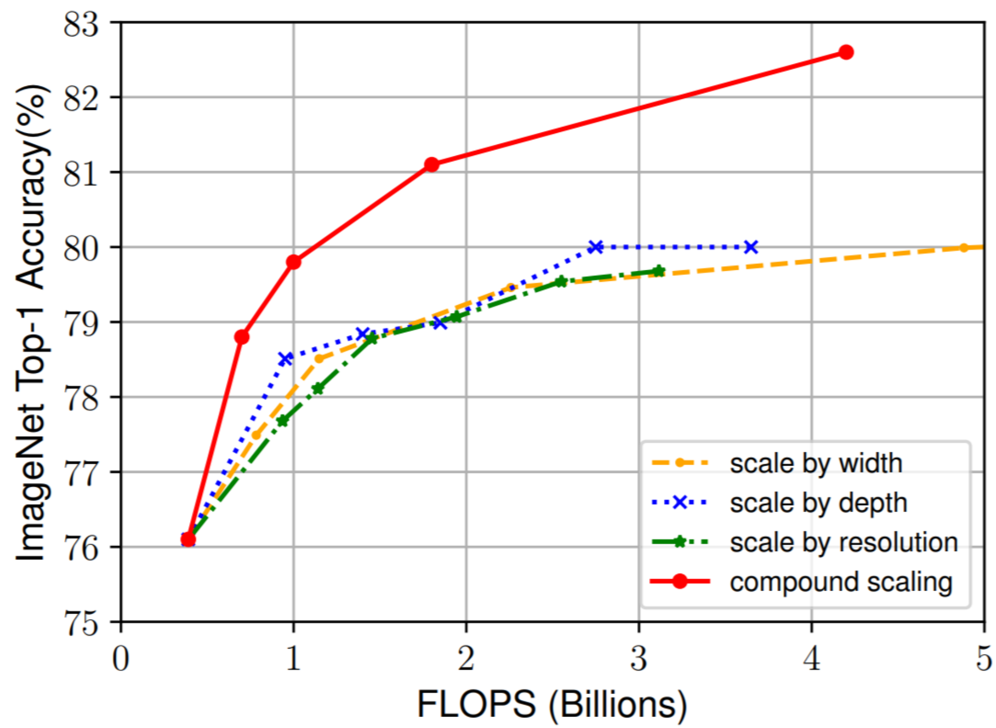
- Scaling all together with a fixed ratio

*source : Tan et al., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", ICML 2019

- **Compound scaling:** Scaling all together with a fixed ratio ϕ in a principled way
  - Depth $d = \alpha^{\phi}, \alpha \geq 1$
  - Width $w = \beta^{\phi}, \beta \geq 1$
  - Resolution $r = \gamma^{\phi}, \gamma \geq 1$
  - Finding α, β, γ under compound constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$
    - **Why?** Such scaling approximately increases total FLOPS by $(\alpha \cdot \beta^2 \cdot \gamma^2)^{\phi} \approx 2^{\phi}$



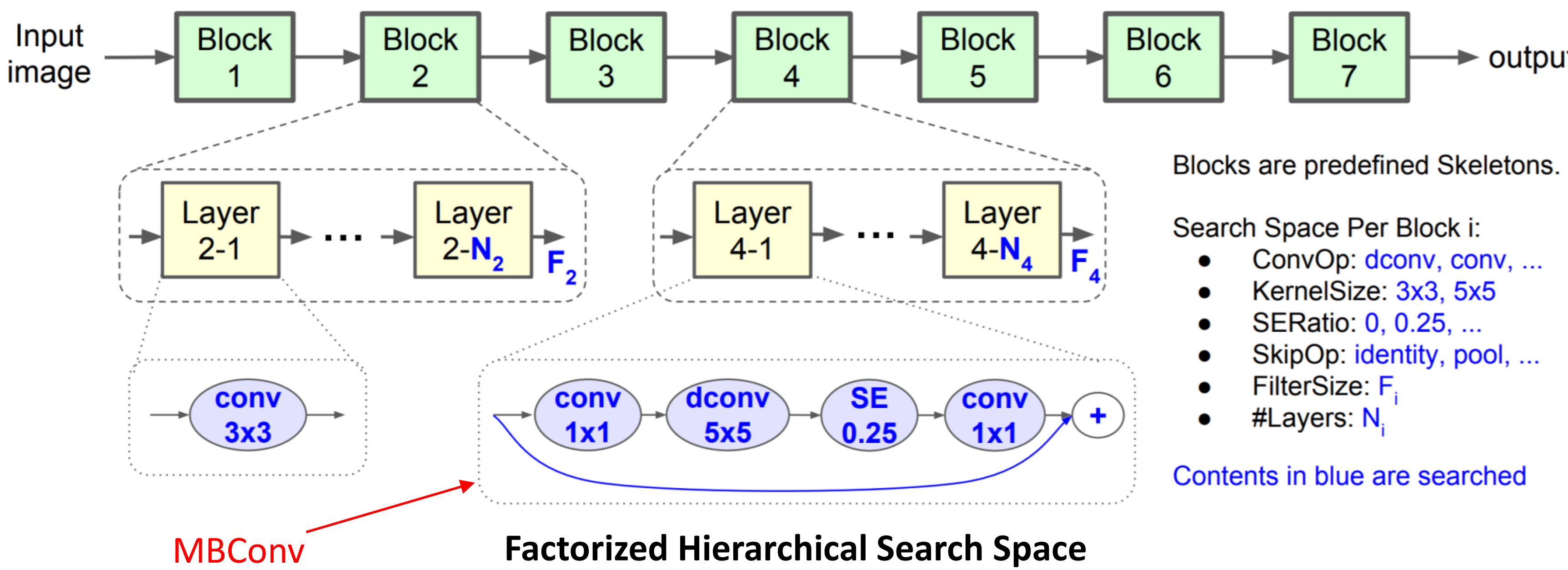
(e) compound scaling



*source : Tan et al., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", ICML 2019

Having a good baseline network is also critical!

- Multi-objective neural architecture search
  - Optimizing both **accuracy** and **FLOPS**
  - Search space is the same as MnasNet [Tan et al., 2019]
- Mobile-size baseline, called **EfficientNet-B0**
  - Main building block is mobile inverted bottleneck, **MBConv**
  - Adding squeeze-and-excitation (SE) optimization [Hu et al., 2018]



Factorized Hierarchical Search Space

*source : Tan et al., "Mnasnet: Platform-aware neural architecture search for mobile", CVPR 2019

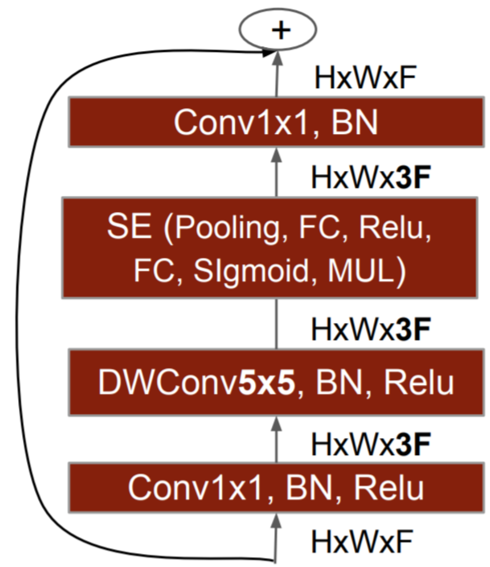
Principle of Network Scaling: EfficientNet [Tan et al., 2019]

Having a good baseline network is also critical!

- Multi-objective neural architecture search
  - Optimizing both **accuracy** and **FLOPS**
  - Search space is the same as MnasNet [Tan et al., 2019]
- Mobile-size baseline, called **EfficientNet-B0**
  - Main building block is mobile inverted bottleneck, **MBConv**
  - Adding squeeze-and-excitation (SE) optimization [Hu et al., 2018]
  - DWConv denotes depthwise convolution [Howard et al ., 2017]

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Architecture of EfficientNet-B0



MBConv

*source : Tan et al., “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, ICML 2019
Tan et al., “Mnasnet: Platform-aware neural architecture search for mobile”, CVPR 2019

From EfficientNet-B0 to B7

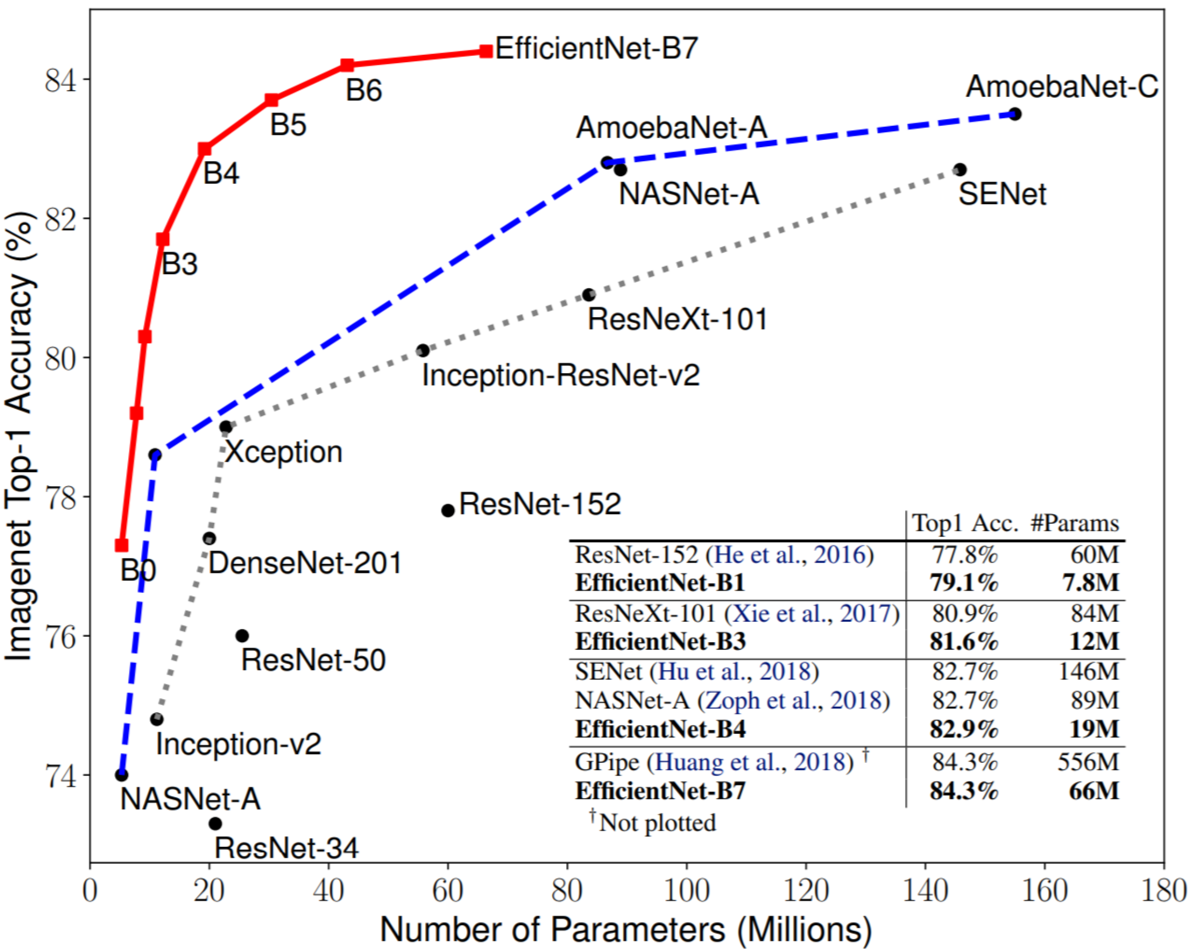
- **EfficientNet-B0**: Baseline model with $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$
- **EfficientNet-B1 to B7**: Scaling up EfficientNet-B0 with different ϕ

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

*source : Tan et al., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", ICML 2019

From EfficientNet-B0 to B7

- **EfficientNet-B0**: Baseline model with $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$
- **EfficientNet-B1 to B7**: Scaling up EfficientNet-B0 with different ϕ



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

† Not plotted

EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 1.3x smaller than NASNet-A.

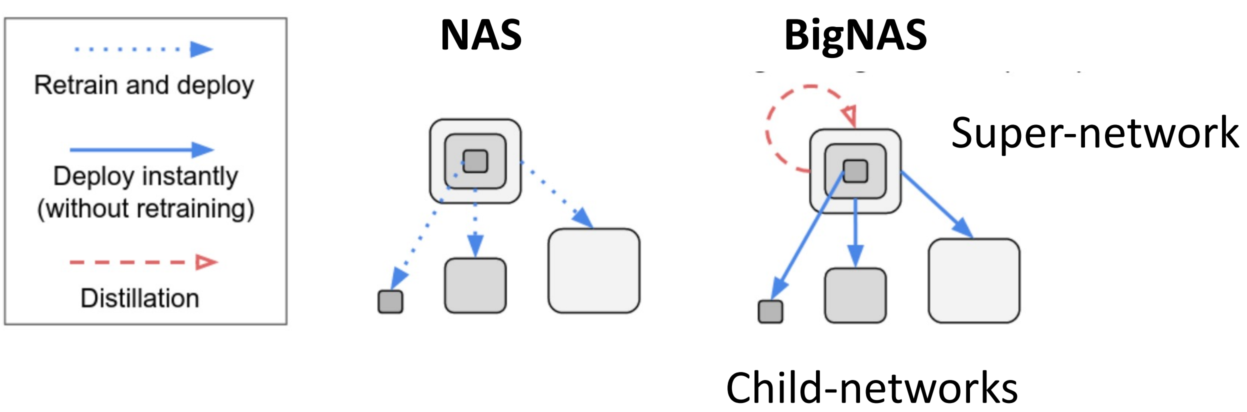
EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152

*source : Luo et al., “Neural Architecture Optimization”, Arxiv 2018

Automation of networks at different scales: BigNAS [Yu et al., 2020]

A **searched architecture** at each scale **requires re-training** from scratch

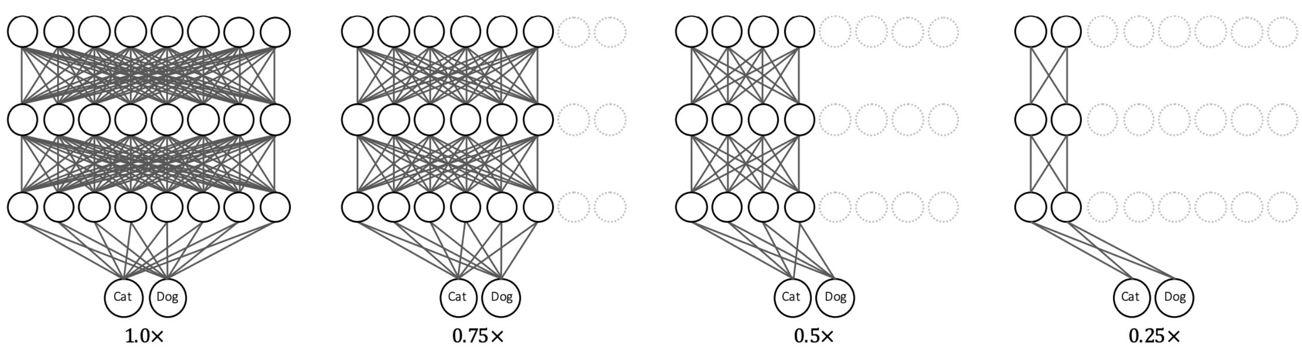
- Can we share weights between architecture instances?
- **BigNAS** trains a single set of parameters (super-network), then sample its subset (child-network)
  - A child-network can be evaluated and deployed without re-training!
  - How to train such a super-network?



*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

A **searched architecture** at each scale **requires re-training** from scratch

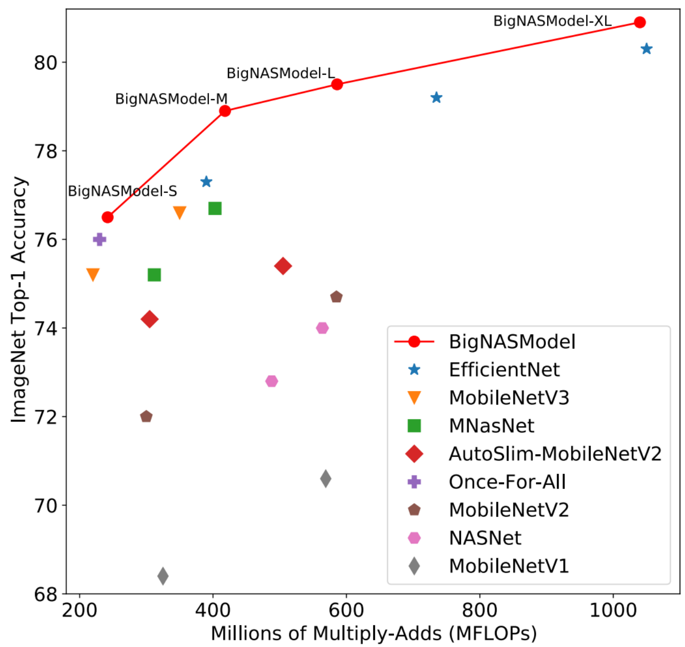
- Can we share weights between architecture instances?
- **BigNAS** trains a single set of parameters (super-network), then sample its subset (child-network)
    - **Sandwich Training Rule** (each iteration)
        - Sample the **biggest, smallest, and N random-sized children**
        - **Gradients are averaged** between all children
    - **Inplace Distillation**
        - Soft labels predicted by the biggest child model supervises all other child models



*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

A **searched architecture** at each scale **requires re-training** from scratch

- Can we share weights between architecture instances?
- **BigNAS** trains a single set of parameters (super-network), then sample its subset (child-network)
    - **BigNAS** sampled at different scale **outperforms existing models** without re-training
    - Training & evaluating BigNAS takes only 1300 TPU-hours (c.f., 60000 GPU-hours in original NAS)



Group	Model Family	Params	FLOPs	Top-1
200M FLOPs	MobileNetV1 $_{0.5\times}$	1.3M	150M	63.3
	MobileNetV2 $_{0.75\times}$	2.6M	209M	69.8
	AutoSlim-MobileNetV2	4.1M	207M	73.0
	MobileNetV3 $_{1.0\times}$	5.4M	219M	75.2
	MNasNet $_{A1}$	3.9M	315M	75.2
	Once-For-All	4.4M	230M	76.0
	Once-For-All $_{\text{finetuned}}$	4.4M	230M	76.4
	BigNASModel-S	**4.5M**	**242M**	**76.5**
400M FLOPs	NASNet $_{B}$	5.3M	488M	72.8
	MobileNetV2 $_{1.3\times}$	5.3M	509M	74.4
	MobileNetV3 $_{1.25\times}$	8.1M	350M	76.6
	MNasNet $_{A3}$	5.2M	403M	76.7
	EfficientNet $_{B0}$	5.3M	390M	77.3
	BigNASModel-M	**5.5M**	**418M**	**78.9**
600M FLOPs	MobileNetV1 $_{1.0\times}$	4.2M	569M	70.9
	NASNet $_{A}$	5.3M	564M	64.0
	DARTS	4.9M	595M	73.1
	EfficientNet $_{B1}$	7.8M	734M	79.2
	BigNASModel-L	**6.4M**	**586M**	**79.5**
1000M FLOPs	EfficientNet $_{B2}$	9.2M	1050M	80.3
	BigNASModel-XL	**9.5M**	**1040M**	**80.9**

*source : Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR 2018

Architecture searching is still an active research area

- AmoebaNet [Real et al., 2018]
- NAONet [Luo et al., 2018]
- BigNAS [Yu et al., 2020]
- NASViT [Gong et al., 2022]

- Specifically, NAS for vision transformers is emerging
  - Careful NAS design is required due to architectural differences
  - e.g., **Vision transformers are instable** during the early training stage due to the **lack** of **inductive bias for images**

Group	Method	M FLOPs	Top-1 accuracy (%)
200-300 (M)	AlphaNet-A0	203	77.9
	NASViT-A0 (ours)	**208**	**78.2**
300-400 (M)	LeViT (Graham et al., 2021)	300	76.6
	NASViT-A1 (ours)	**309**	**79.7**
	AlphaNet-A2	317	79.4
	FBNetV3 (Dai et al., 2020)	357	79.6
400-500 (M)	LeViT	406	78.6
	NASViT-A2 (ours)	**421**	**80.5**
	AlphaNet-A4	444	80.4
500-600 (M)	**NASViT-A3 (ours)**	**528**	**81.0**
	FBNetV3	557	80.8
	NASViT-A4 (ours)	**591**	**81.4**
	AlphaNet	596	81.1
600 - 1000 (M)	LeViT	658	80.0
	NASViT-A5 (ours)	**757**	**81.8**
	FBNetV3	762	81.5
> 1000 (M)	AutoFormer* (Chen et al., 2021a)	1,300	74.7
	PiT-XS (Heo et al., 2021)	1,400	79.1
	ViTAS-D* (Su et al., 2021)	1,600	76.2
	NASViT (supernet) (ours)	**1,881**	**82.9**
	CVT-13-NAS* (Wu et al., 2021)	4,100	82.2
	Swin-Tiny* (Liu et al., 2021)	4,500	81.3
	CVT-13* (Wu et al., 2021)	4,500	81.6
	T2T-ViT-14* (Yuan et al., 2021a)	5,200	81.5
	DeepViT (Zhou et al., 2021)	6,200	82.3

*source : Luo et al., “Neural Architecture Optimization”, Arxiv 2018

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Objects in real-world often contain sophisticated spatial information

- Multiple scales
- Irregular shapes

Drawbacks: geometric transformations are assumed fixed and known

- Different size and shape of kernels may be required
- But, regular kernels have fixed-size and shape

Scale:

Deformation:

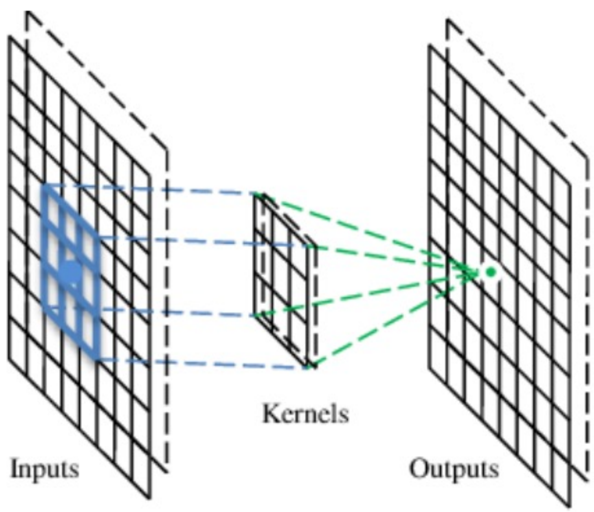
*source : https://jifengdai.org/slides/Deformable_Convolutional_Networks_Oral.pdf

Objects in real-world often contain sophisticated spatial information

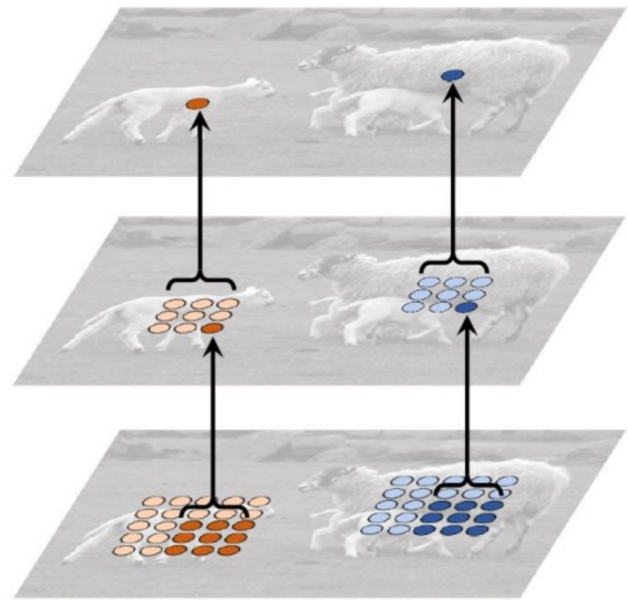
- Multiple scales
- Irregular shapes

Drawbacks: geometric transformations are assumed fixed and known

- Different size and shape of kernels may be required
- But, regular kernels have fixed-size and shape



regular convolution



2 layers of regular convolution

*source : https://jifengdai.org/slides/Deformable_Convolutional_Networks_Oral.pdf

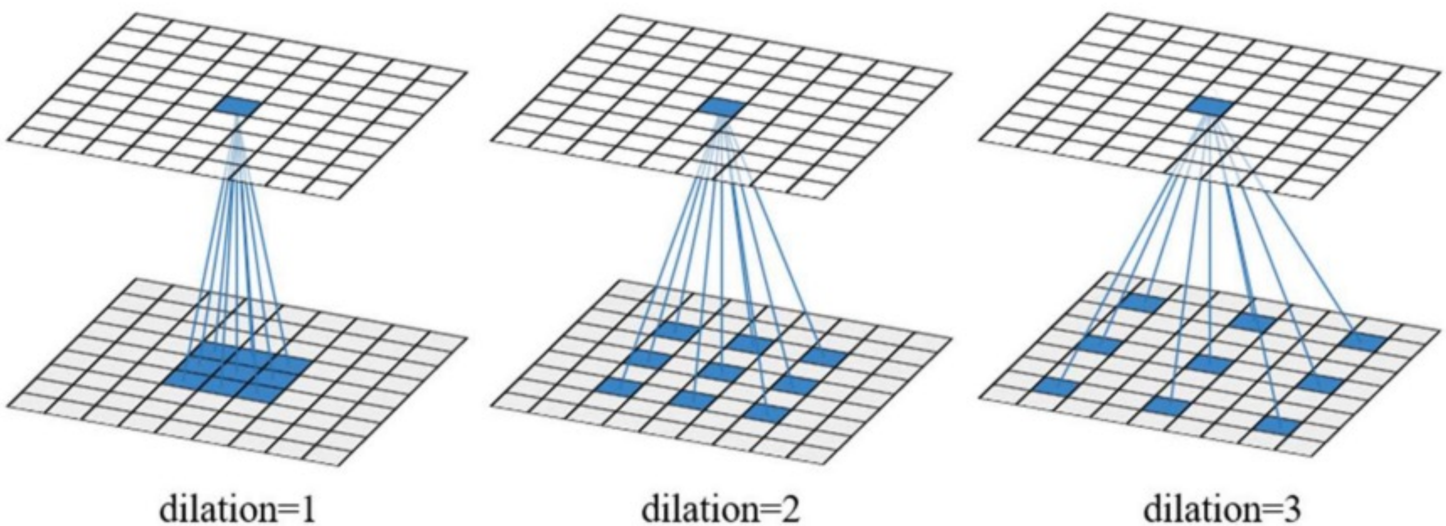
Dilated Convolution [Chen et al., 2017]

Motivation: Images in real-world usually contain **multi-scale objects**

- Regular convolution has a fixed-size of field of view
- Different size of kernels are required for multi-scale objects
- But, large-size of kernels may increase computational costs

Dilated convolution: Filling with **zero values** inside of large-size of kernels for efficient computation

- It can enlarge field-of-view to incorporate multi-scale context



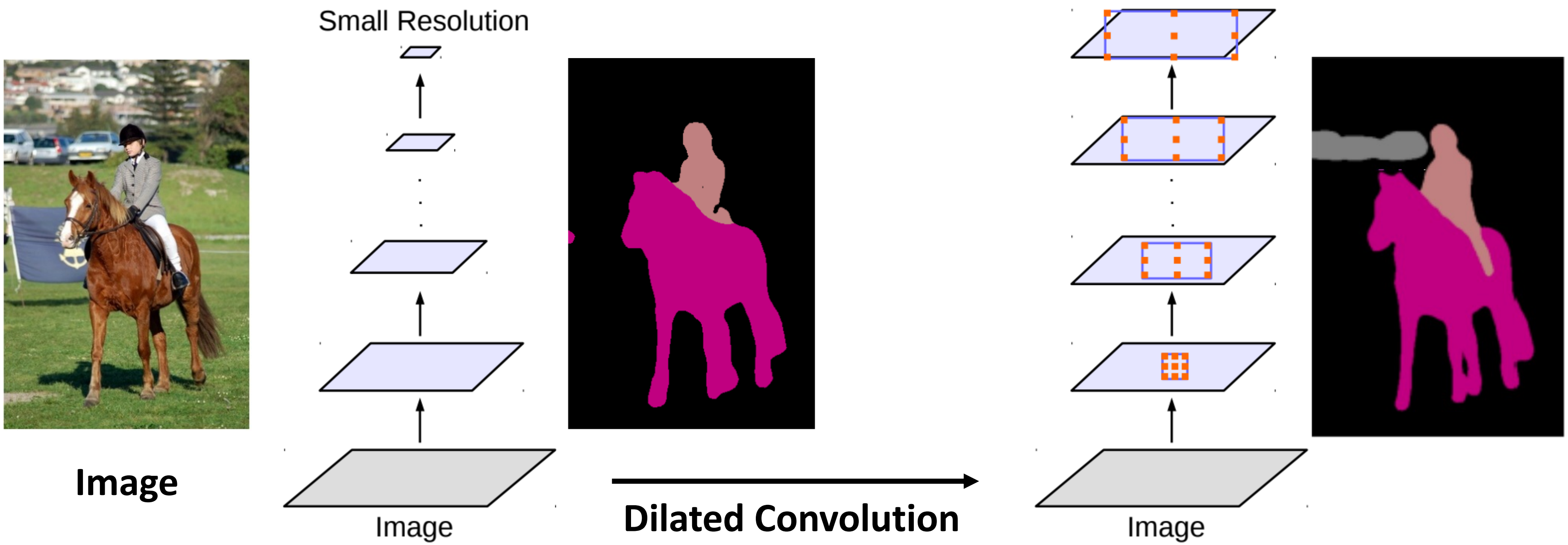
*source: Chen et al., "Rethinking atrous convolution for semantic image segmentation", ArXiv, 2019

Dilated Convolution [Chen et al., 2017]

Motivation: Images in real-world usually contain **multi-scale objects**

- Regular convolution has a fixed-size of field of view
- Different size of kernels are required for multi-scale objects
- But, large-size of kernels may increase computational costs

- **Example: Dilated convolution** in semantic segmentation



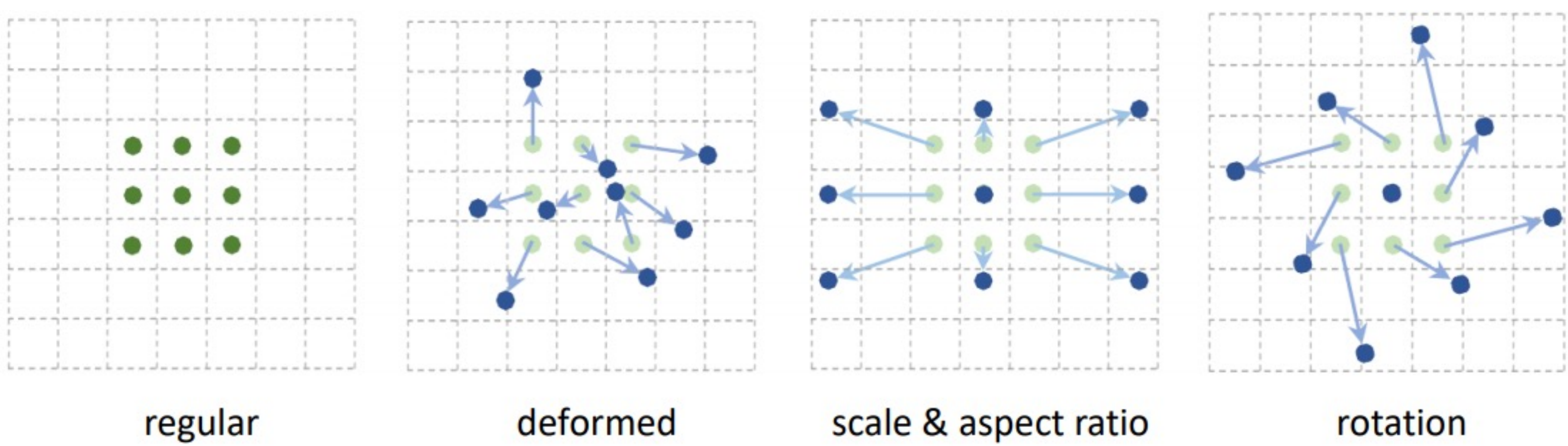
*source: Chen et al., "Rethinking atrous convolution for semantic image segmentation", ArXiv, 2019

Motivation: Shape of objects in the real world are usually irregular

- Different shape of kernels are required for irregular objects
- Regular convolution has a fixed-shape of kernel

Deformable convolution: Learning sampling location of kernels to capture irregular shape of objects

- Adding **offset field** to generate irregular sampling locations



Different types of sampling locations

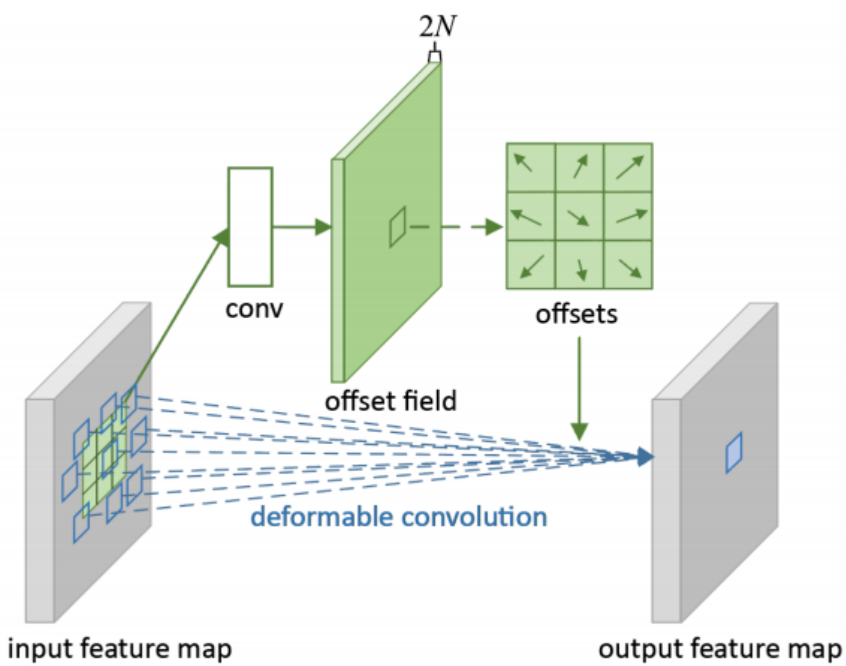
*source : https://jifengdai.org/slides/Deformable_Convolutional_Networks_Oral.pdf

Motivation: Shape of objects in the real world are usually irregular

- Different shape of kernels are required for irregular objects
- Regular convolution has a fixed-shape of kernel

Deformable convolution: Learning sampling location of kernels to capture irregular shape of objects

- Adding **offset field** to generate irregular sampling locations



Regular convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n)$$

Deformable convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n)$$

where $\Delta\mathbf{p}_n$ is generated by a sibling branch of regular convolution (offset field)

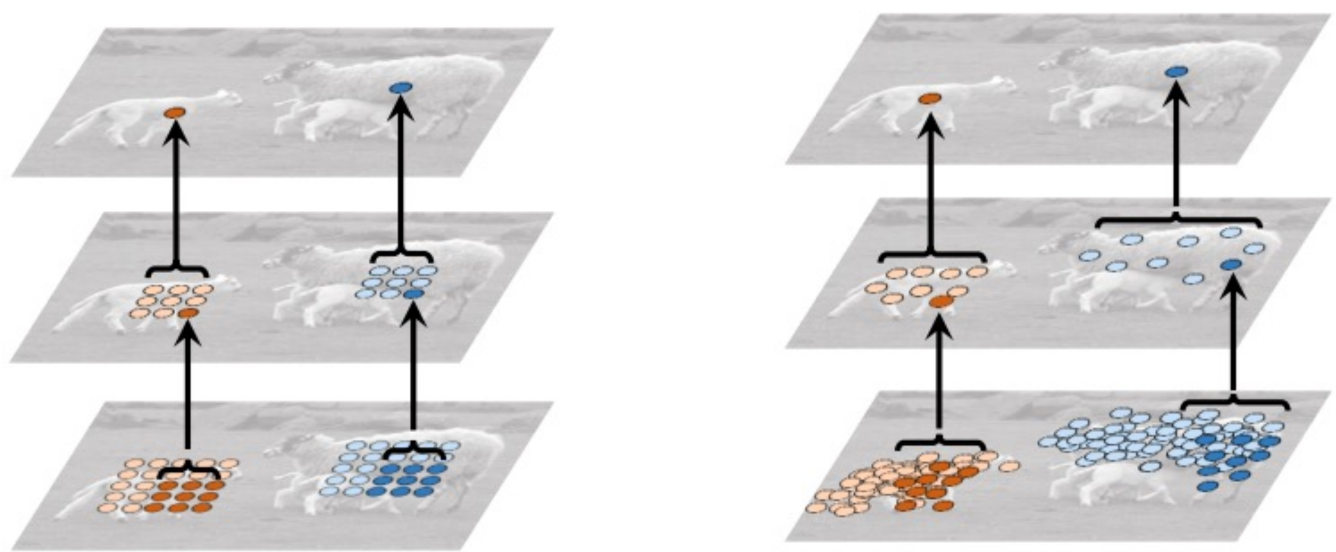
*source : https://jifengdai.org/slides/Deformable_Convolutional_Networks_Oral.pdf

Motivation: Shape of objects in the real world are usually irregular

- Different shape of kernels are required for irregular objects
- Regular convolution has a fixed-shape of kernel

Deformable convolution: Learning sampling location of kernels to capture irregular shape of objects

- Adding **offset field** to generate irregular sampling locations



(a) standard convolution (b) deformable convolution

*source : https://jifengdai.org/slides/Deformable_Convolutional_Networks_Oral.pdf

Deformable Convolution [Dai et al., 2017]

Motivation: Shape of objects in the real world are usually irregular

- Different shape of kernels are required for irregular objects
- Regular convolution has a fixed-shape of kernel

Learned offsets in the **deformable convolution** layers are highly adaptive to the image content

- Different size and shape of kernels for multiple objects

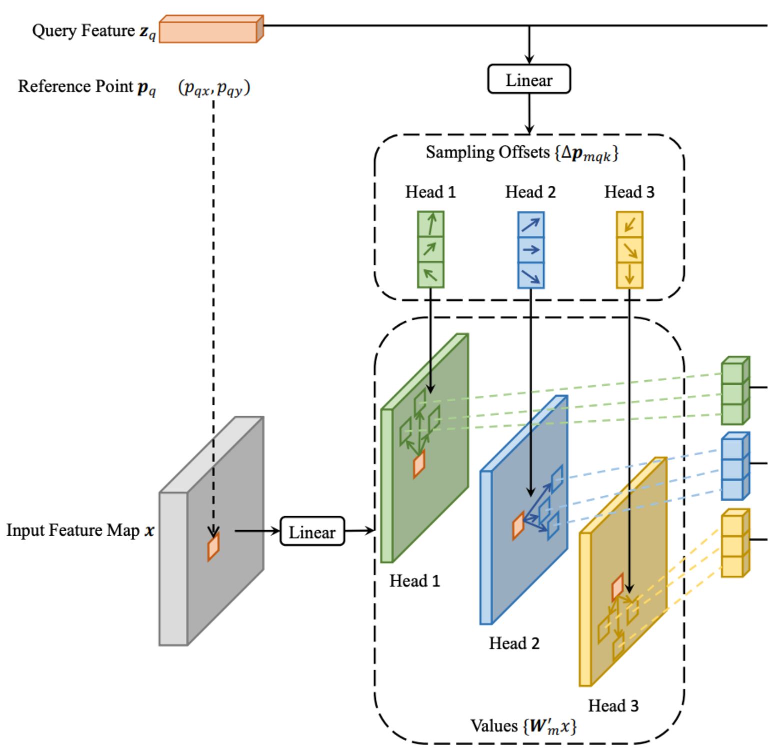
Visualizations of sampling locations

*source: Dai et al., “Deformable Convolutional Networks”, ICCV, 2017

Motivation: Make image patches in vision transformers deformable!

Square patches in the **vision transformers** could be too restrictive for localization (e.g., object detection, segmentation)

- Deformable DETR [Zhu et al., 2020] additionally learns the offset of pixels in a patch



Regular attention

$$\sum_{m=1}^{M} \boldsymbol{W}_m \big[\sum_{k \in \Omega_k} A_{mqk} \cdot \boldsymbol{W}'_m \boldsymbol{x}_k \big],$$

Deformable attention

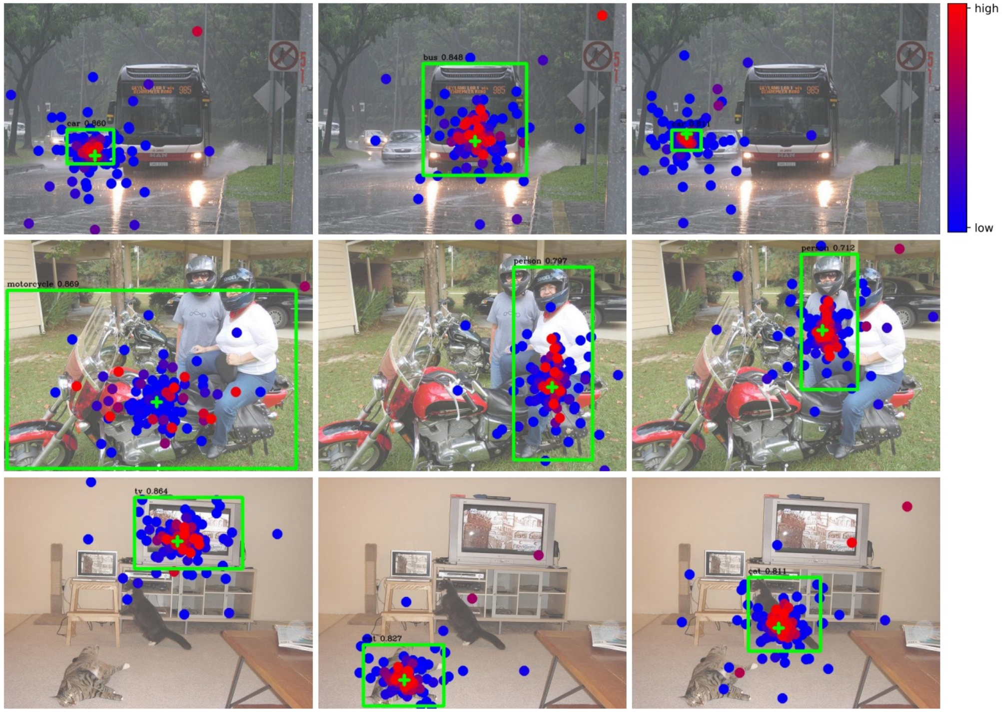
$$\sum_{m=1}^{M} \boldsymbol{W}_m \big[\sum_{k=1}^{K} A_{mqk} \cdot \boldsymbol{W}'_m \boldsymbol{x}(\boldsymbol{p}_q + \Delta \boldsymbol{p}_{mqk}) \big]$$

*source: Dai et al., “Deformable Convolutional Networks”, ICCV, 2017

Motivation: Make image patches in vision transformers deformable!

Square patches in the **vision transformers** could be too restrictive for localization (e.g., object detection, segmentation)

- Deformable DETR [Zhu et al., 2021] additionally learns the offset of pixels in a patch
- **Self-attention** is regularized around the **localization of objects**

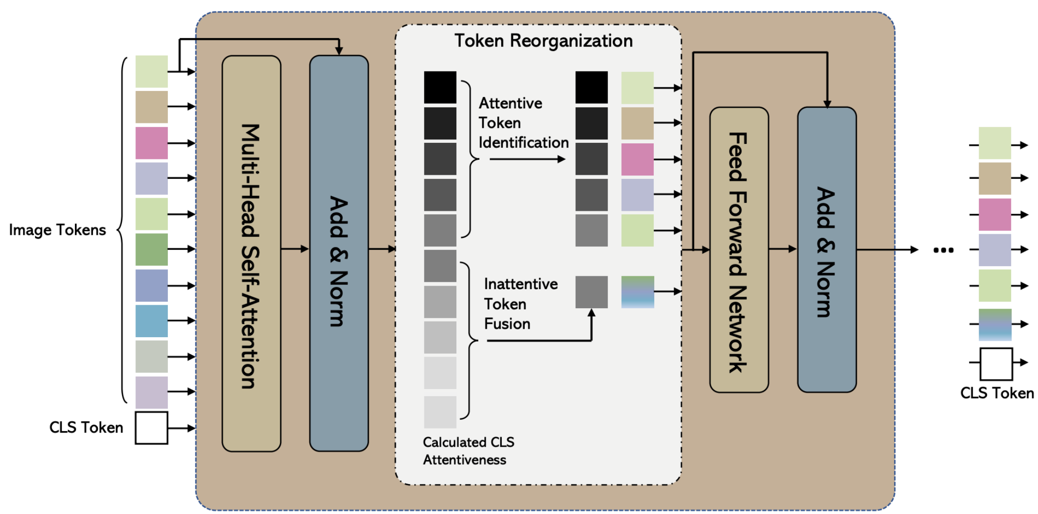


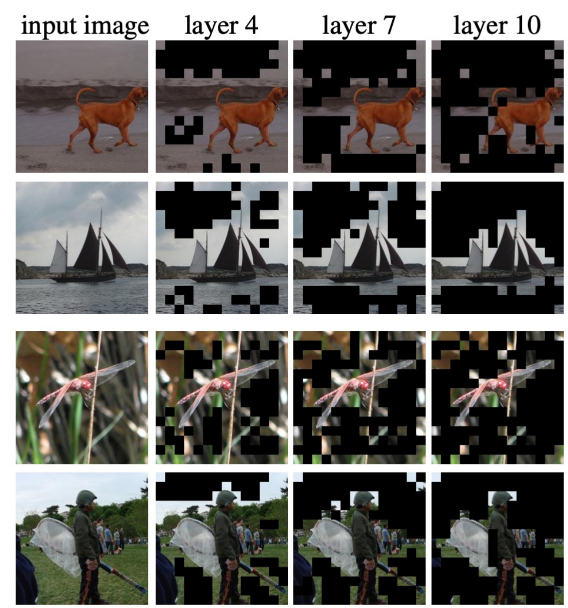
*source: Dai et al., "Deformable Convolutional Networks", ICCV, 2017

Motivation: Not all patches are equivalently important

Some **image patches** could contain **redundant** and less important information
- EViT [Liang et al., 2022], ATS [Fayyaz et al., 2022] merges these patches
- Less important patches (e.g., background) are identified at each attention layer
  - Attention & value-norms are used as importance scores

$$\mathcal{S}_j = \frac{\mathcal{A}_{1,j} \times ||\mathcal{V}_j||}{\sum_{i=2} \mathcal{A}_{1,i} \times ||\mathcal{V}_i||}$$



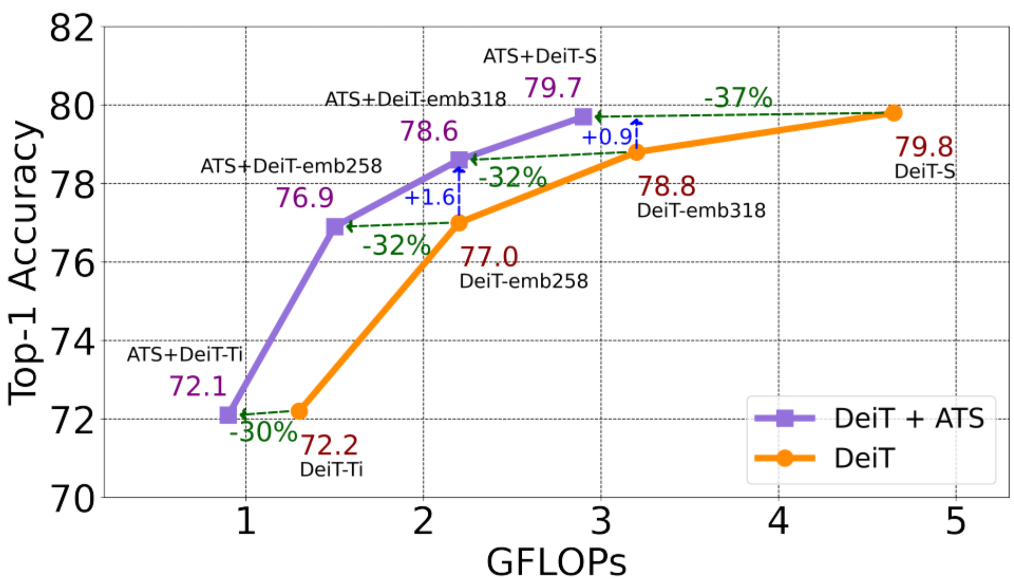


*source: Dai et al., "Deformable Convolutional Networks", ICCV, 2017

Motivation: Not all patches are equivalently important

Some image patches could contain redundant and less important information

- EViT [Liang et al., 2022], ATS [Fayyaz et al., 2022] merges these patches
- ATS [Fayyaz et al., 2022] achieves the comparable accuracy at 37% reduced computations (GFLOPs) than DeiT



ImageNet classification accuracy per GFLOP

*source: Dai et al., “Deformable Convolutional Networks”, ICCV, 2017

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

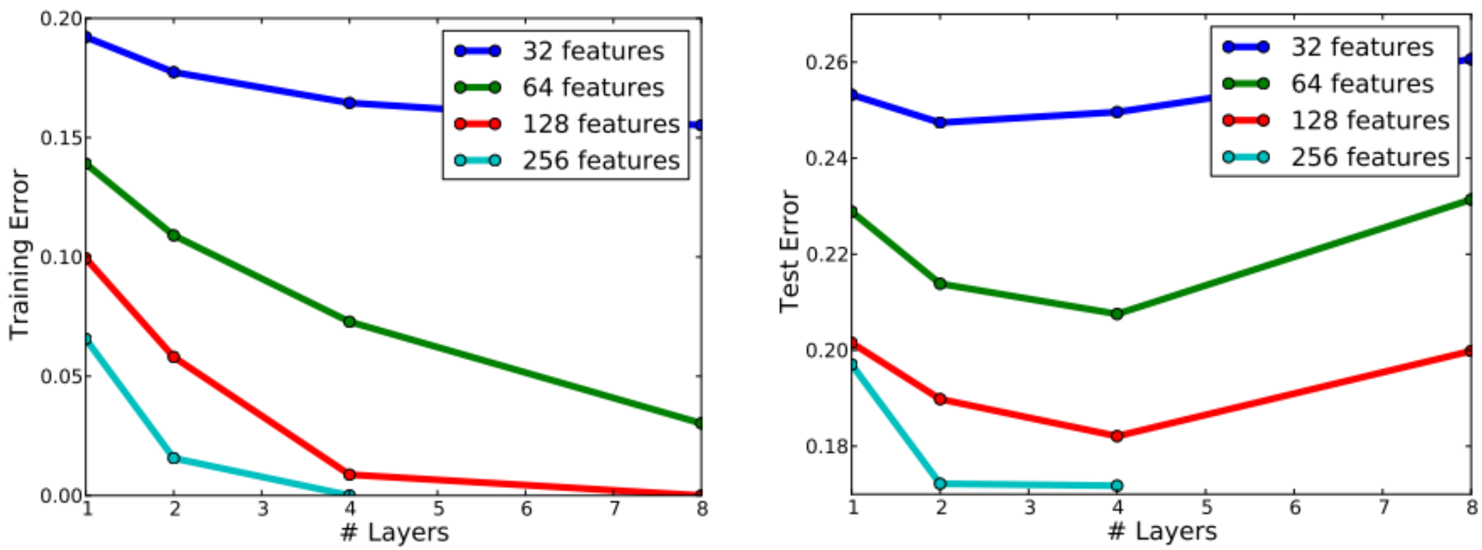
- Patch-based architectures for vision
- New design paradigms

ResNet improved generalization by revolution of depth

Quiz: But, does it fully explain why deep ResNets generalize well?

Increasing depth **does not always mean** better generalization

- Naïve CNNs are very easy to overfit on deeper networks [Eigen et al., 2014]

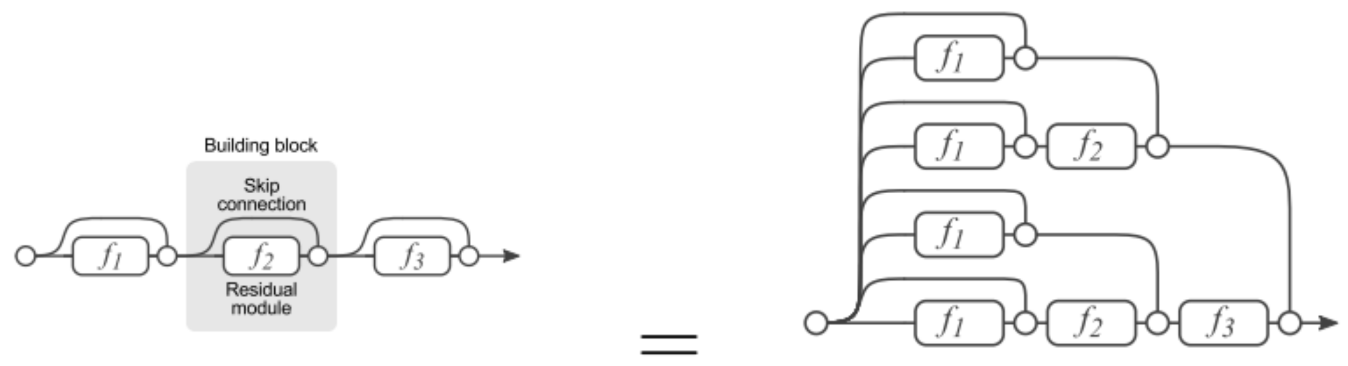


*source : Eigen et al., "Understanding Deep Architectures using a Recursive Convolutional Network", Arxiv 2014

Veit et al. (2016): ResNet can be viewed as a collection of many paths, instead of a single ultra-deep network

- Each module in a ResNet receives a **mixture of 2^{n-1} different distributions**

$$
\begin{aligned}
y_3 &= y_2 + f_3(y_2) \\
&= \left[\boxed{y_1 + f_2(y_1)}\right] + f_3\left(\boxed{y_1 + f_2(y_1)}\right) \\
&= \left[\boxed{y_0 + f_1(y_0)} + f_2\left(\boxed{y_0 + f_1(y_0)}\right)\right] + f_3\left(\boxed{y_0 + f_1(y_0)} + f_2\left(\boxed{y_0 + f_1(y_0)}\right)\right)
\end{aligned}
$$



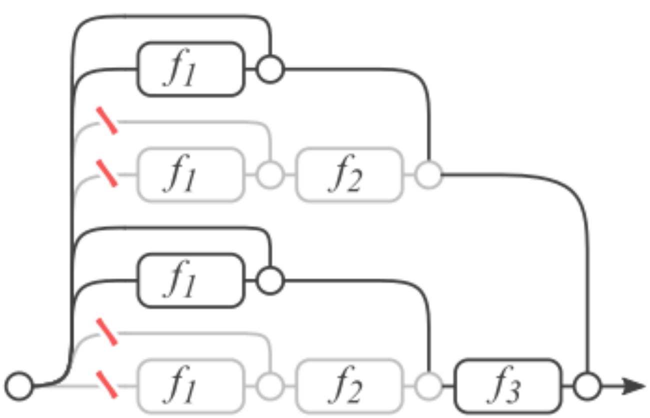
(a) Conventional 3-block residual network

(b) Unraveled view of (a)

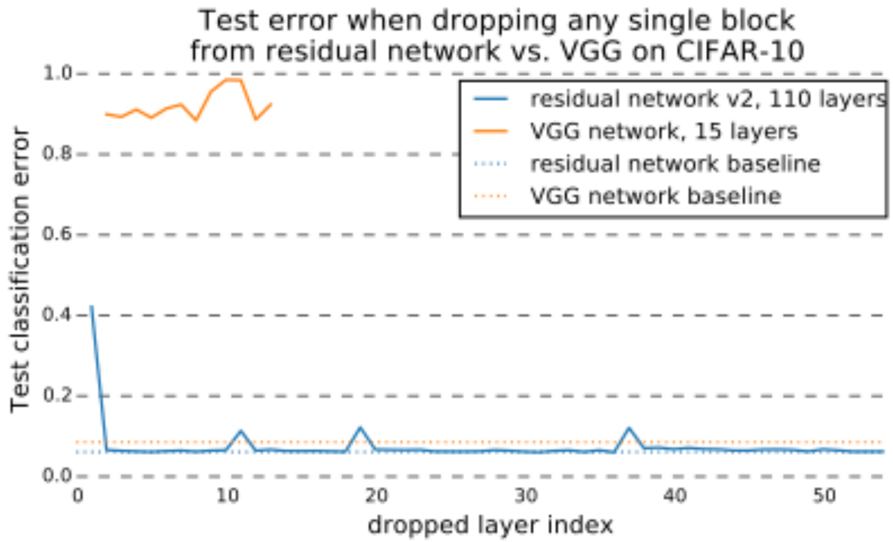
*source : Veit et al., “ResNets behave like ensembles of relatively shallow nets”, NIPS 2016

Veit et al. (2016): ResNet can be viewed as a collection of many paths, instead of a single ultra-deep network

- Deleting a module in ResNet has a minimal effect on performance
- Similar effect as removing 2^{n-1} paths out of 2^n: still 2^{n-1} paths alive!



(a) Deleting f_2 from unraveled view



Next, visualizing loss functions in CNN

*source : Veit et al., “ResNets behave like ensembles of relatively shallow nets”, NIPS 2016

Trainability of neural nets is highly dependent on network architecture

- However, the effect of each choice on the underlying loss surface is unclear
  - Why are we able to minimize highly non-convex neural loss?
  - Why do the resulting minima generalize?

Li et al. (2018) analyzes **random-direction 2D plot of loss** around local minima

$$f(\alpha, \beta) = L(\theta^* + \alpha\delta + \beta\eta)$$

Local minima **Random directions**

- δ and η are sampled from a random Gaussian distribution
- To remove some scaling effect, δ and η are normalized filter-wise

$$\delta_{i,j} \leftarrow \frac{\delta_{i,j}}{||\delta_{i,j}||}||\theta_{i,j}||$$

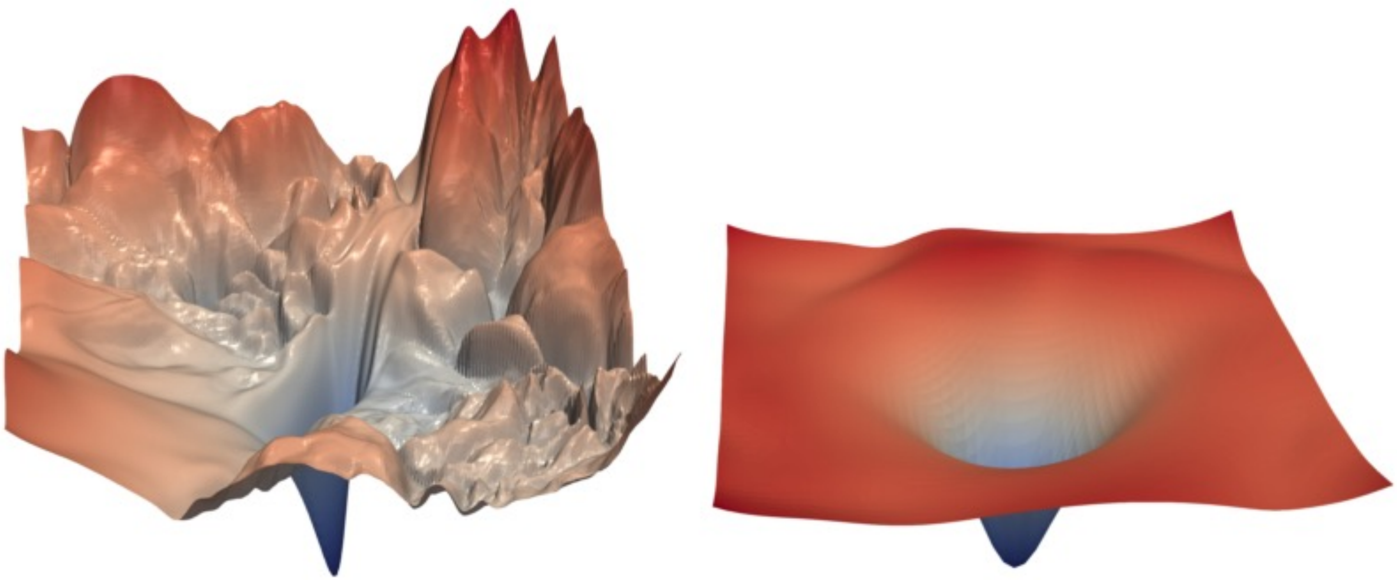
i^{th} **layer,** j^{th} **filter**

*source : Li et al., "Visualizing the loss landscape of neural nets", ICLR Workshop 2018

Visualizing the loss landscape of neural nets [Li et al., 2018]

Li et al. (2018) analyzes **random-direction 2D plot of loss** around local minima

Modern architectures prevent the loss to be chaotic as depth increases



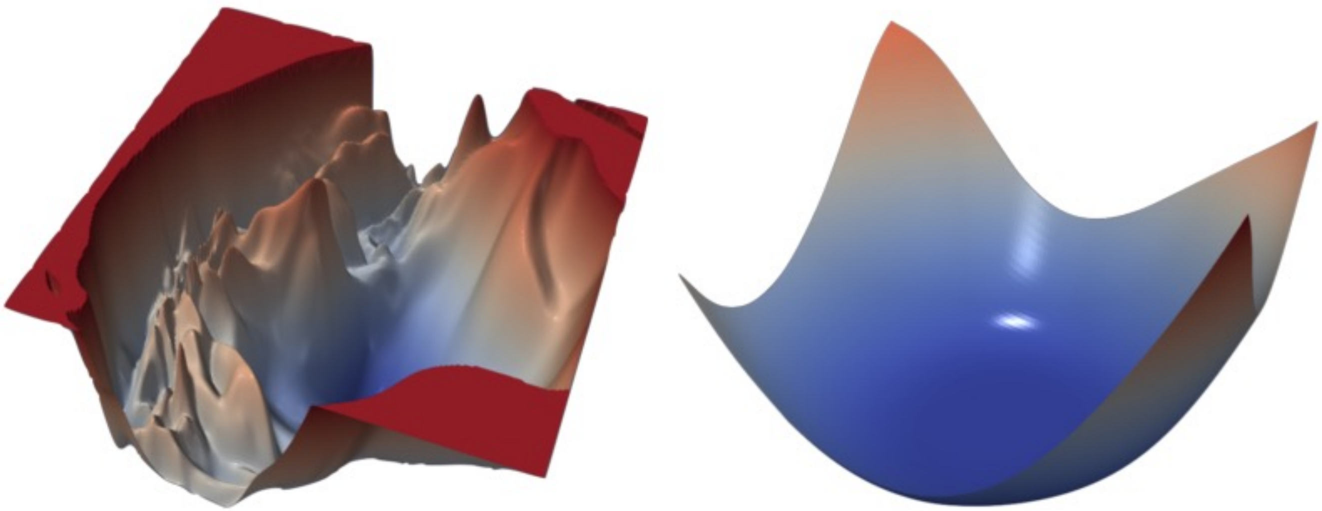
(a) without skip connections (b) with skip connections

ResNet-56

*source : Li et al., "Visualizing the loss landscape of neural nets", ICLR Workshop 2018

Li et al. (2018) analyzes **random-direction 2D plot of loss** around local minima

Modern architectures prevent the loss to be chaotic as depth increases



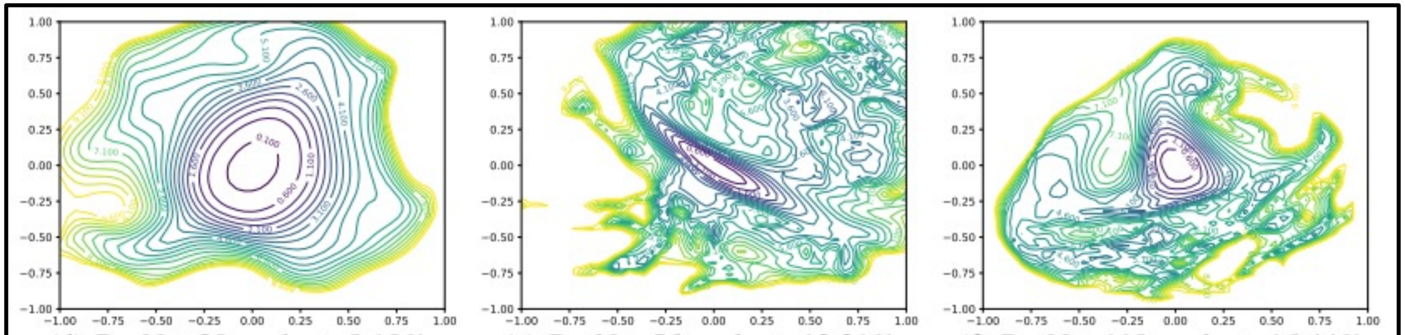
(a) 110 layers, no skip connections (b) DenseNet, 121 layers

*source : Li et al., "Visualizing the loss landscape of neural nets", ICLR Workshop 2018

Li et al. (2018) analyzes **random-direction 2D plot of loss** around local minima

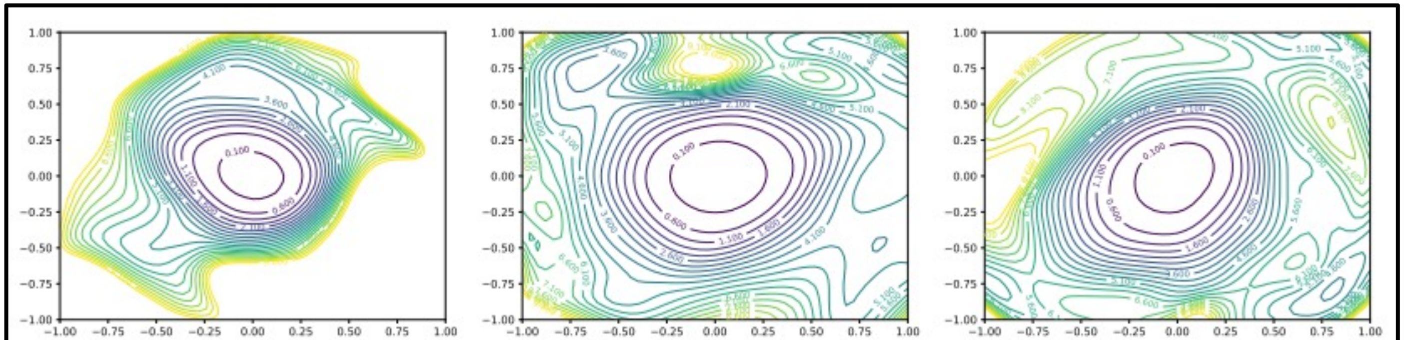
Modern architectures prevent the loss to be chaotic as depth increases

ResNet, **no shortcuts** ⇒ sharp minima



(d) ResNet-20-noshort, 8.18% (e) ResNet-56-noshort, 13.31% (f) ResNet-110-noshort, 16.44%

ResNet ⇒ flat minima



(a) ResNet-20, 7.37% (b) ResNet-56, 5.89% (c) ResNet-110, 5.79%

*source : Li et al., "Visualizing the loss landscape of neural nets", ICLR Workshop 2018

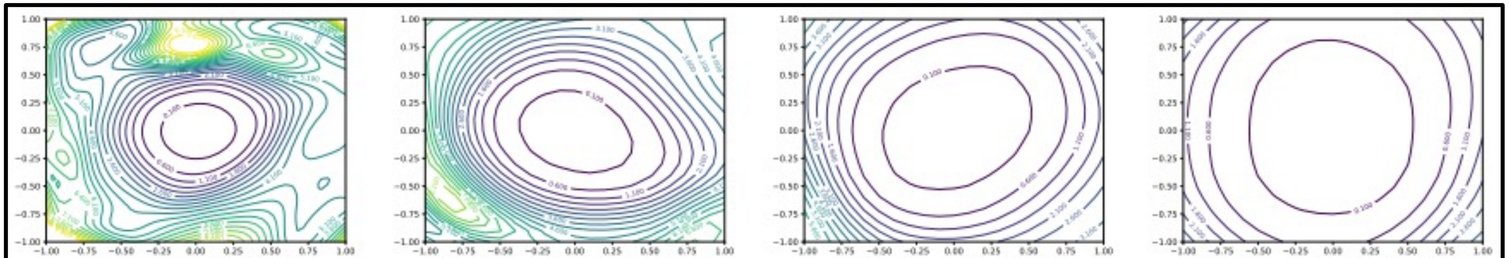
Visualizing the loss landscape of neural nets [Li et al., 2018]

Li et al. (2018) analyzes **random-direction 2D plot of loss** around local minima

Wide-ResNet lead the network toward more flat minimizer

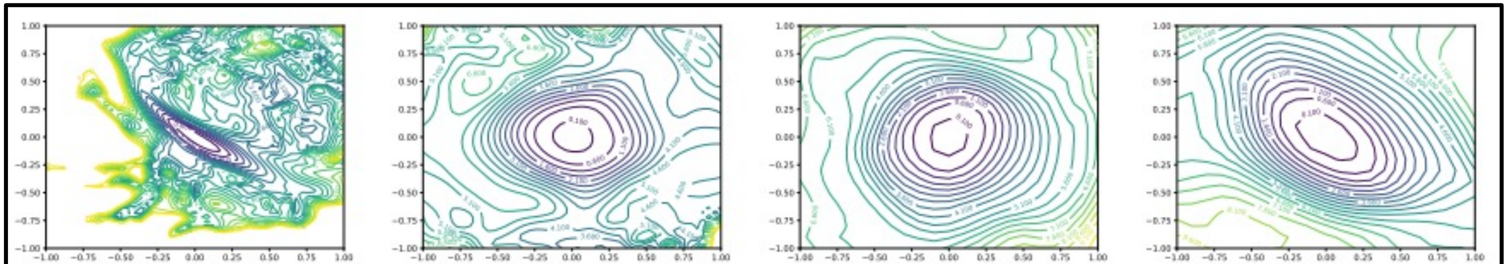
- WideResNet-56 with **width-multiplier** $k = 1, 2, 4, 8$
- Increased width flatten the minimizer in ResNet

WRN-56



(a) $k = 1$, 5.89%　(b) $k = 2$, 5.07%　(c) $k = 4$, 4.34%　(d) $k = 8$, 3.93%

WRN-56, **no shortcuts**



(e) $k = 1$, 13.31%　(f) $k = 2$, 10.26%　(g) $k = 4$, 9.69%　(h) $k = 8$, 8.70%

Next, minimum energy paths in CNNs

*source : Li et al., "Visualizing the loss landscape of neural nets", ICLR Workshop 2018

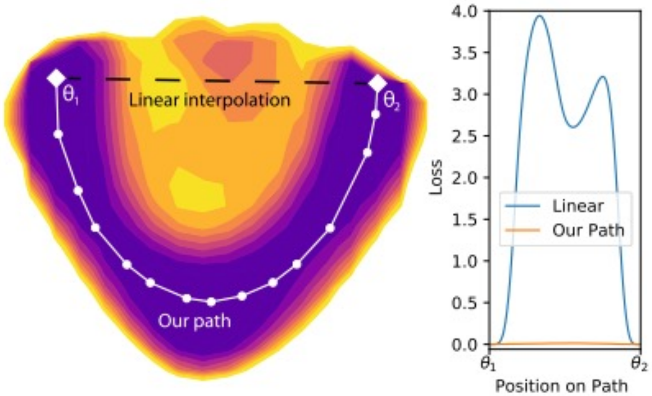
Essentially no barriers in neural network energy landscape [Draxler et al., 2018]

Draxler et al. (2018) analyzes **minimum energy paths** [Jónsson et al., 1998] between two local minima θ_1 and θ_2 of a given model:

$$p(\theta_i, \theta_2)^* = \underset{\text{path } p:\ \theta_1 \to \theta_2}{\operatorname{argmin}} \left(\max_{\theta \in p} L(\theta) \right)$$

- They found a path $\theta_1 \to \theta_2$ with almost zero barrier
  - A path that **keeps low loss constantly** both in training and test
- The gap vanishes as the model grows, especially on modern architectures
  - e.g. ResNet, DenseNet

- Minima of a loss of deep neural networks are perhaps on a single connected manifold



DenseNet-40-12

*source : Draxler et al., "Essentially no barriers in neural network energy landscape", ICML 2018

Essentially no barriers in neural network energy landscape [Draxler et al., 2018]

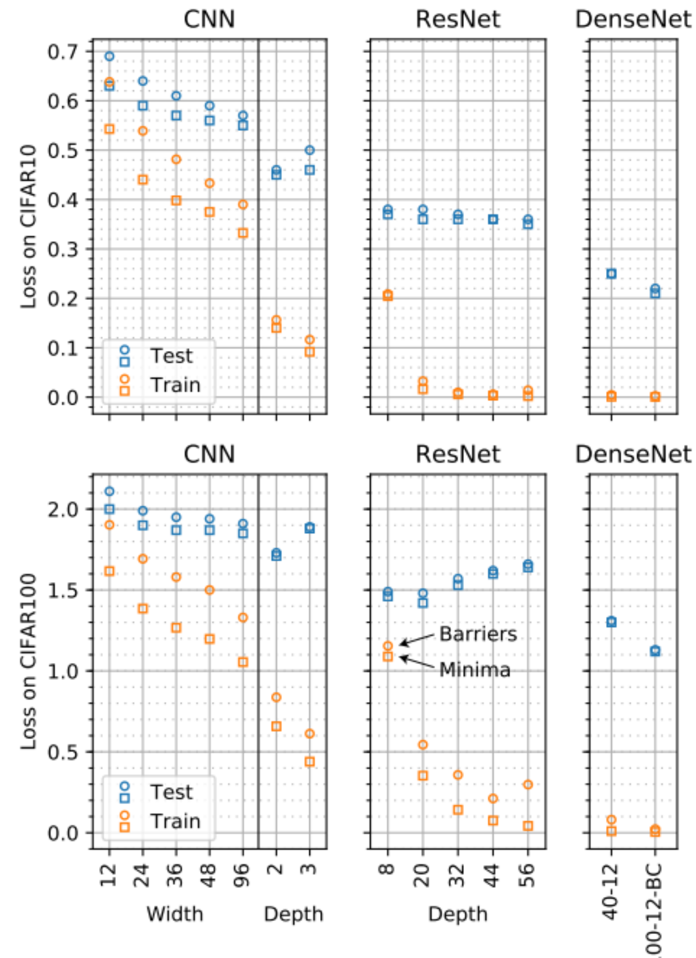
For a given model with two local minima θ_1 and θ_2, they applied **AutoNEB** [Kolsbjerg et al., 2016] to find a minimum energy path

- A state-of-the-art for connecting minima from molecular statistical mechanics

- The deeper and wider an architecture, the lower are the saddles between minima

- They essentially vanish for current-day deep architectures

- The **test accuracy** is also preserved
  - **CIFAR-10**: < +0.5%
  - **CIFAR-100**: < +2.2%

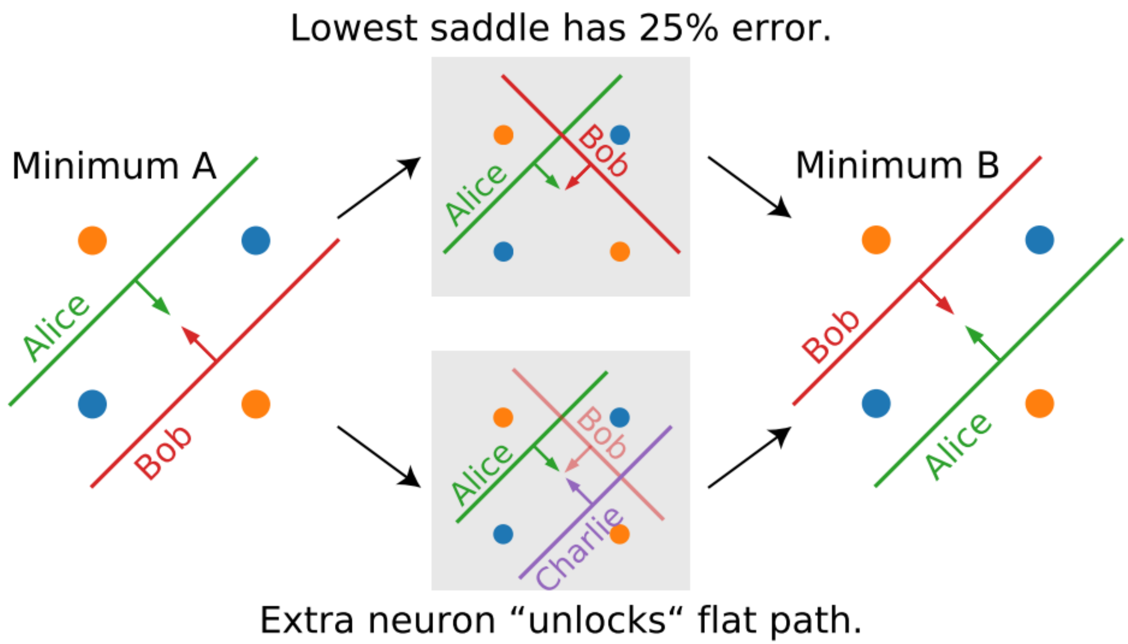


*source : Draxler et al., “Essentially no barriers in neural network energy landscape”, ICML 2018

- The deeper and wider an architecture, the lower are the barriers
- They essentially vanish for current-day deep architectures

Why do this phenomenon happen?

- **Parameter redundancy** may help to flatten the neural loss

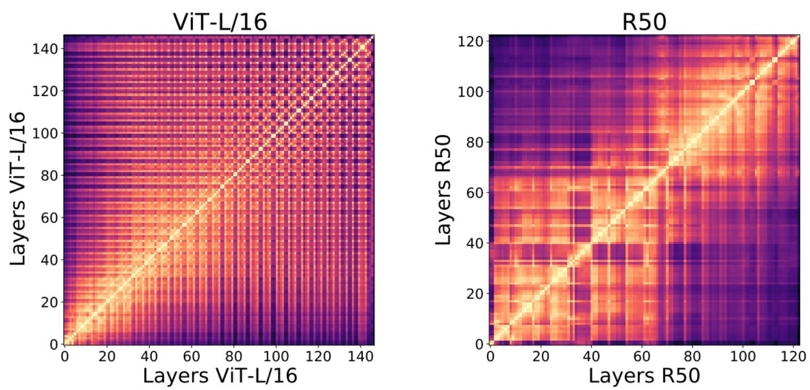


*source : Draxler et al., “Essentially no barriers in neural network energy landscape”, ICML 2018

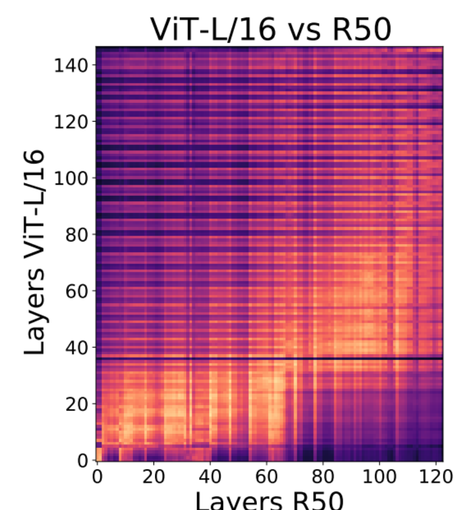
Raghu et al. (2021) analyzes **representation similarity** in transformer layers:

- **ViT** tends to have **uniform representation** over different layers
  - All layers in ViT show much greater similarity than ResNet
  - In ResNet, similarity is divided into different (lower/higher) stages

- ViT and ResNet features are similar in lower stages, but significantly different in higher stages



Cosine similarity of representations in layers within ViT and ResNet



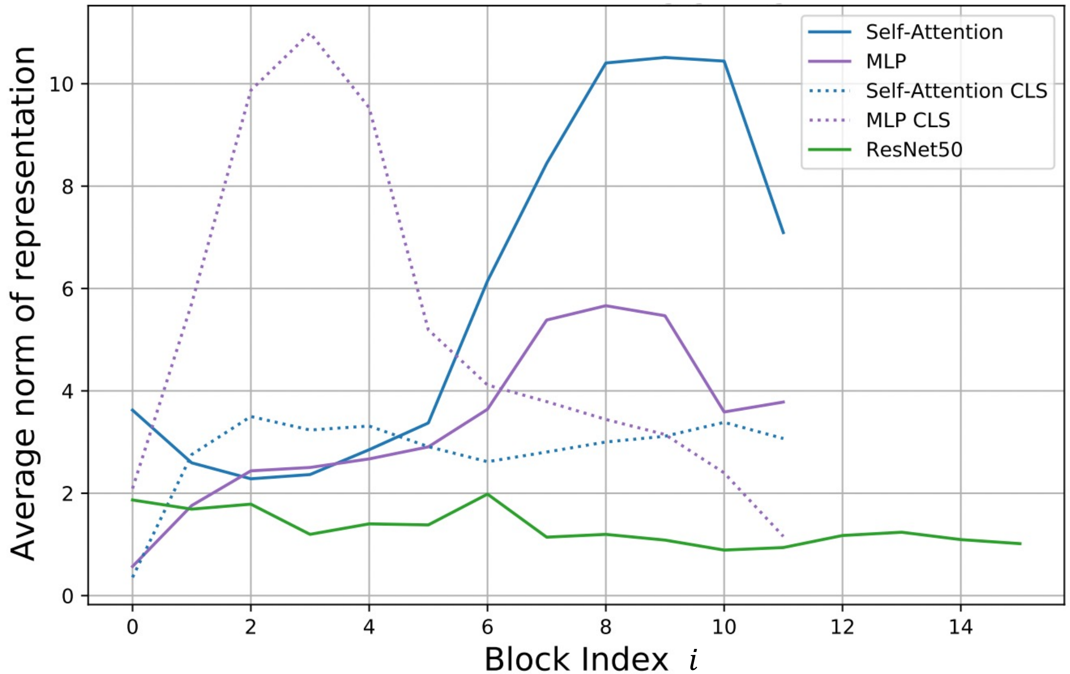
Cosine similarity of representations in layers between ViT vs. ResNet

*source : Draxler et al., “Essentially no barriers in neural network energy landscape”, ICML 2018

Raghu et al. (2021) analyzes **representation similarity** in transformer layers:

- **ViT** tends to have **uniform representation** over different layers
  - All layers in ViT show much greater similarity than ResNet
  - In ResNet, similarity is divided into different (lower/higher) stages
- This is mainly due to **stronger skip-connection in ViT**
  - $\frac{\|\boldsymbol{z_i}\|}{\|f_i(\boldsymbol{z}_i)\|}$: norm ratio of $\boldsymbol{z_i}$ (**skip-connection**) and $\boldsymbol{f_i}$ **(MLP or Self-Attention)**
  - The **skip-connection** in ViT is even stronger in **deeper layers**

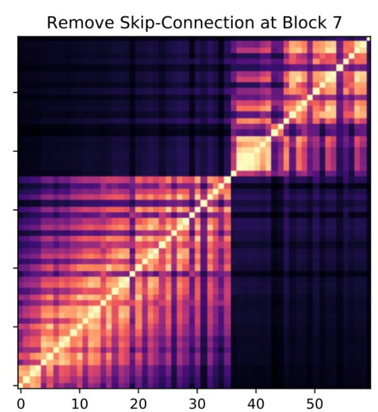
$$\frac{\|\boldsymbol{z_i}\|}{\|f_i(\boldsymbol{z}_i)\|}$$



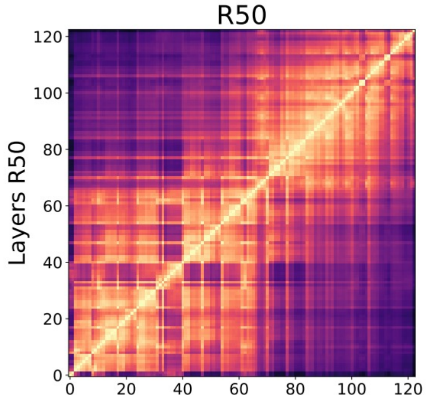
Raghu et al. (2021) analyzes **representation similarity** in transformer layers:

- **ViT** tends to have **uniform representation** over different layers
  - All layers in ViT show much greater similarity than ResNet
  - In ResNet, similarity is divided into different (lower/higher) stages
- This is mainly due to **stronger skip-connection in ViT**
  - $\frac{\|\boldsymbol{z_i}\|}{\|\boldsymbol{f}_i(\boldsymbol{z}_i)\|}$: norm ratio of $\boldsymbol{z_i}$ (**skip-connection**) and $\boldsymbol{f}_i$ **(MLP or Self-Attention)**
  - The **skip-connection** in ViT is even stronger in **deeper layers**

- When **skip-connection removed** at a middle-block (e.g., $i = 7$) the **cosine similarity of ViT becomes similar to that of ResNets**



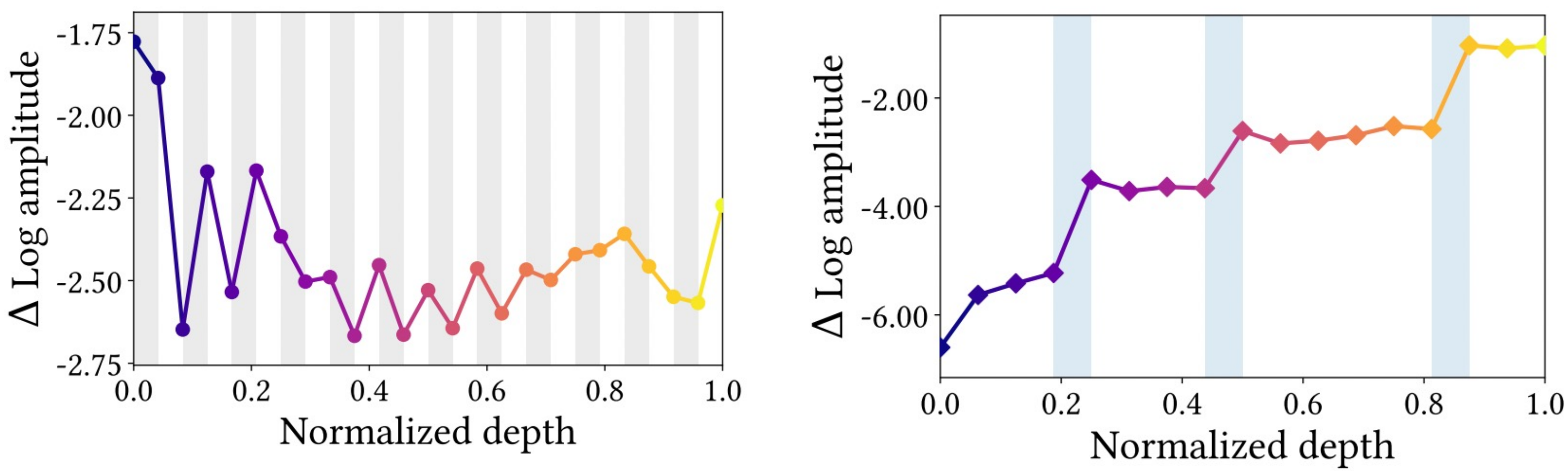
Cosine similarity over the layers
(**ViT** with skip-connection removed at $i = 7$)



Cosine similarity over the layers
(ResNet)

Park et al. (2022) analyzes **frequency domain** of vision transformer layers:

- Self-attention layer keeps high-frequency information
    - **MLPs** variants (e.g., CNNs, MLP in transformers) act as **high-pass filters**
    - However, **self-attention tend to act as low-pass filters**
        - ViT deals with both high- and low-frequency information
          (while CNNs simply pass high-frequency information)



Amplitude of high-frequency signals in Fourier space of feature maps

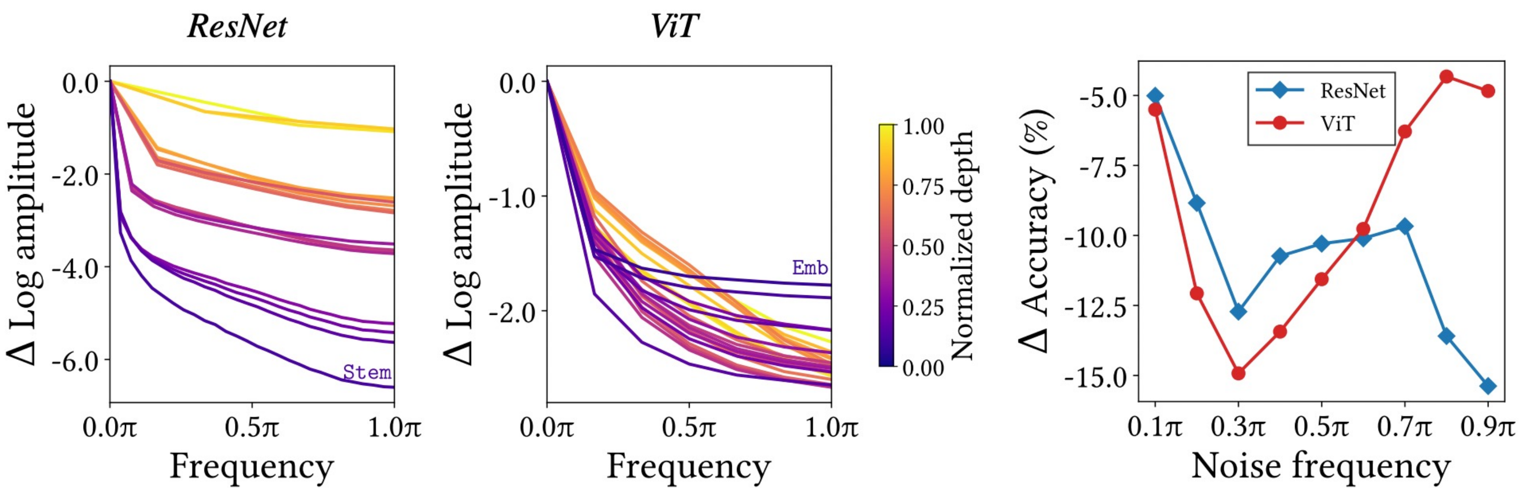
*source : Draxler et al., “Essentially no barriers in neural network energy landscape”, ICML 2018

Park et al. (2022) analyzes **frequency domain** of vision transformer layers:

- Self-attention layer keeps high-frequency information
  - MLPs variants (e.g., CNNs, MLP in transformers) act as high-pass filters
  - However, self-attention tend to act as low-pass filters

- Processing both low- and high-frequency information contributes to **robustness against high-frequency noises in ViT** vs. ResNet
  - Frequency-specific noise with Gaussian noise $\boldsymbol{\delta}$ and Fourier transform $\boldsymbol{\mathcal{F}}$

$$\boldsymbol{x}_{\text{noise}} = \boldsymbol{x}_0 + \mathcal{F}^{-1}\left(\mathcal{F}(\delta) \odot \boxed{\mathbf{M}_f}\right)$$ frequency mask



(a) Relative log amplitudes of Fourier transformed feature maps. (b) Robustness for noise frequency

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

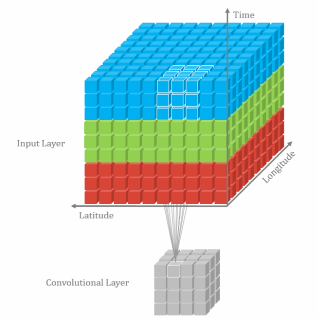
Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

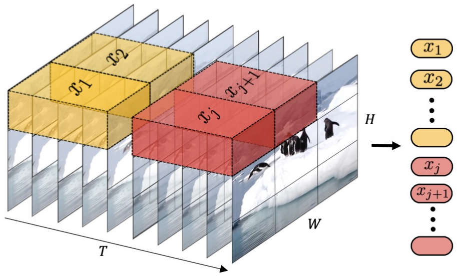
Overview: Deep Spatial-Temporal Models

Deep **spatial-temporal model** as an extension of spatial models

- **3D convolutional neural networks** and **video vision transformers**

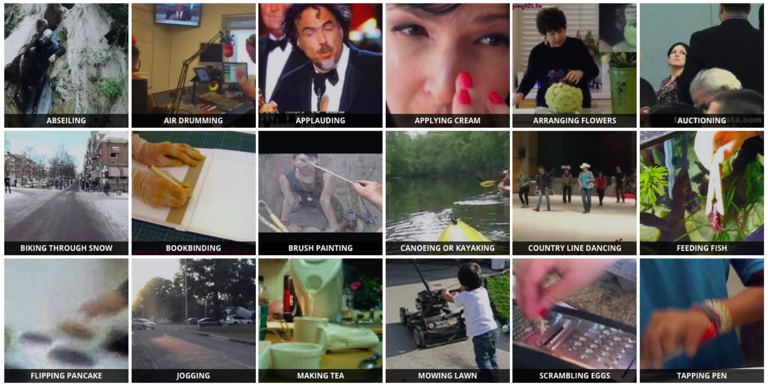


3D Convolutional Neural Networks



Video Vision Transformers

Video Action Recognition [Karpathy et al., 2014]



*source: https://towardsdatascience.com/downloading-the-kinetics-dataset-for-human-action-recognition-in-deep-learning-500c3d50f776

Deep Object Tracking [Wang et al., 2020]

*source: https://github.com/Zhongdao/Towards-Realtime-MOT

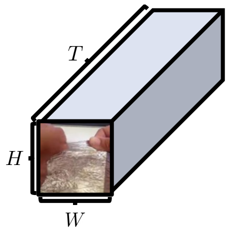
Problem: The curse of dimensionality

- Spatial-temporal data is high-dimensional (e.g., channels × height × width × time)
- Brute-force extension of spatial models is often intractable
- Data sub-sampling & approximated network architectures are typically employed:
  - How to fuse information from spatial cue (appearance) and temporal cue (motion)
  - Long-range modeling

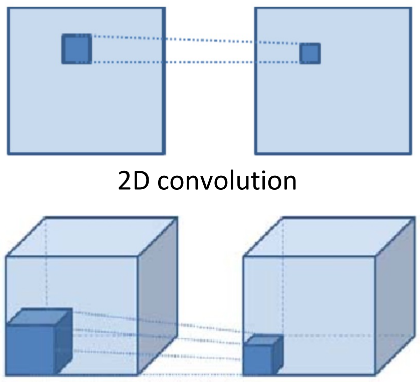
Good models should be **computationally scalable** (e.g., linear complexity to temporal dimension) and should deal with **information fusion** & **long-range modeling** problems

Evolution of CNN Architectures for Video: 3D CNNs

- Raw video data structure
  - Video is a **3D tensor** with 2 spatial and 1 time axes
  - How to learn **good representation** for video?
    - **3D CNN** directly extends convolution with cuboid (3D) kernel

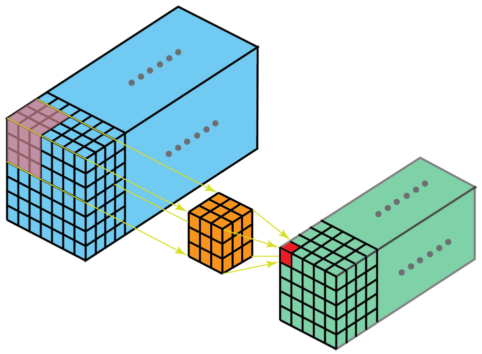


Video Tensor



2D convolution

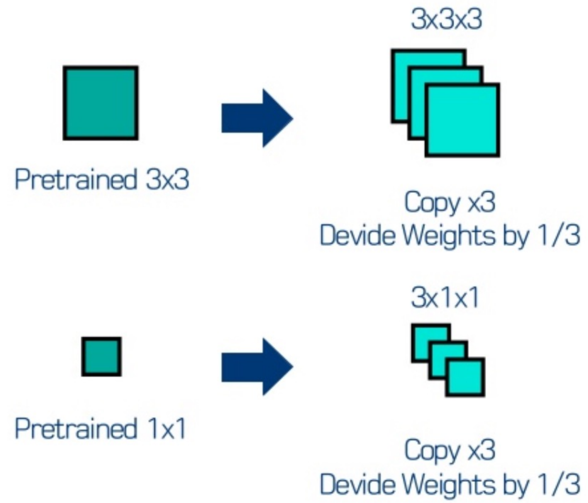
3D convolution



3D convolution in video data

- Some early works employed 3D CNNs for video, however:
  - **3D-Conv** [Ji et al., 2012] and **C3D** [Tran et al., 2015]
  - Their performances were **unsatisfactory** due to **optimization difficulty of 3D CNNs**
    - Can we leverage **pre-trained representation** for images? i.e., **transfer learning**

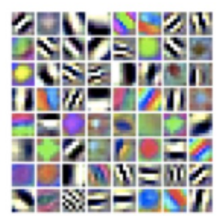
- **Inflated 3D (I3D)** [Carreira and Zisserman, 2017]
  - Adapting a pre-trained 2D CNN model for 3D CNN
    - I3D utilizes the Inception architecture
    - Instead of training from scratch, I3D leverages ImageNet-pretraining
  - **Weight inflating** technique for initializing 3D kernels with 2D kernels
    1. Extend a dimension by stacking existing 2D kernel
    2. Divide weights by the stack length to ensure the same output scale



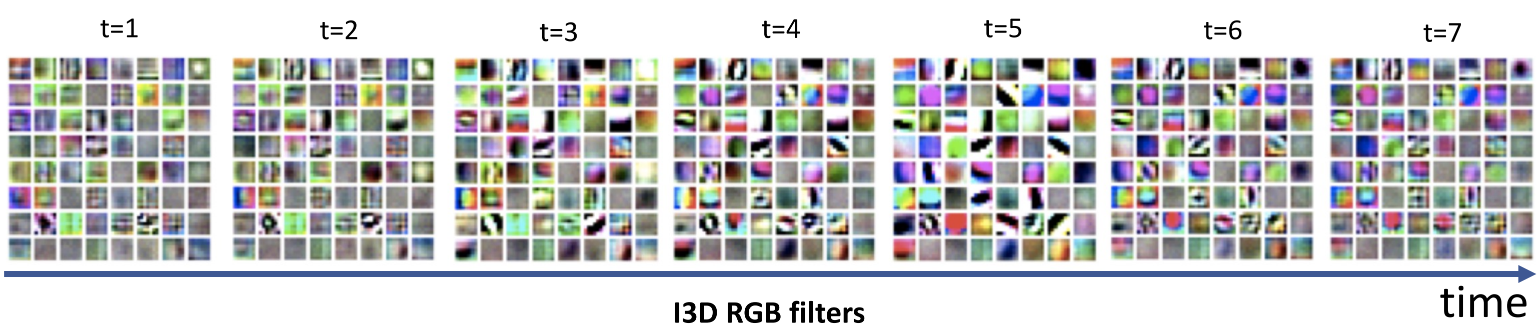
*source : https://chacha95.github.io/2019-07-04-VideoUnderstanding3/

Evolution of CNN Architectures for Video: 3D CNNs

- **Inflated 3D (I3D)** [Carreira and Zisserman, 2017]
  - 3D Convolutional feature map learned by I3D
    - Top row: the 3D filter trained with I3D networks
    - Bottom row: the original 2D filter from Inception
  - 3D kernel sliced at each time resembles geometric patterns of the 2D filter
    - Representation of 2D CNN is **effectively transferred to 3D**



Original 2D filters from Inception



I3D RGB filters

*source : [Carreira and Zisserman, 2017] Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset

- **Inflated 3D (I3D)** [Carreira and Zisserman, 2017]
  - I3D beats hand-craft video representations (e.g., optical flow) by a large margin
  - Transferring the architecture of 2D CNN models is the key idea

  - **ResNet3D** [Hara et al., 2018]
    - Residual connections for 3D CNN
    - Transfers ResNet [He et al., 2016] architecture to 3D CNN

  - **ResNeXt** for 3D [Chen et al., 2018]
    - Multi-Fiber Networks for Video Recognition
    - Translates the multiple parallel path to 3D CNN

  - **STCNet** [Diba et al., 2018]
    - Spatio-Temporal Channel correlation networks
    - Translates the Sequeeze-and-Excitation mechanism to 3D CNN

Evolution of CNN Architectures for Video: Efficient 3D CNNs

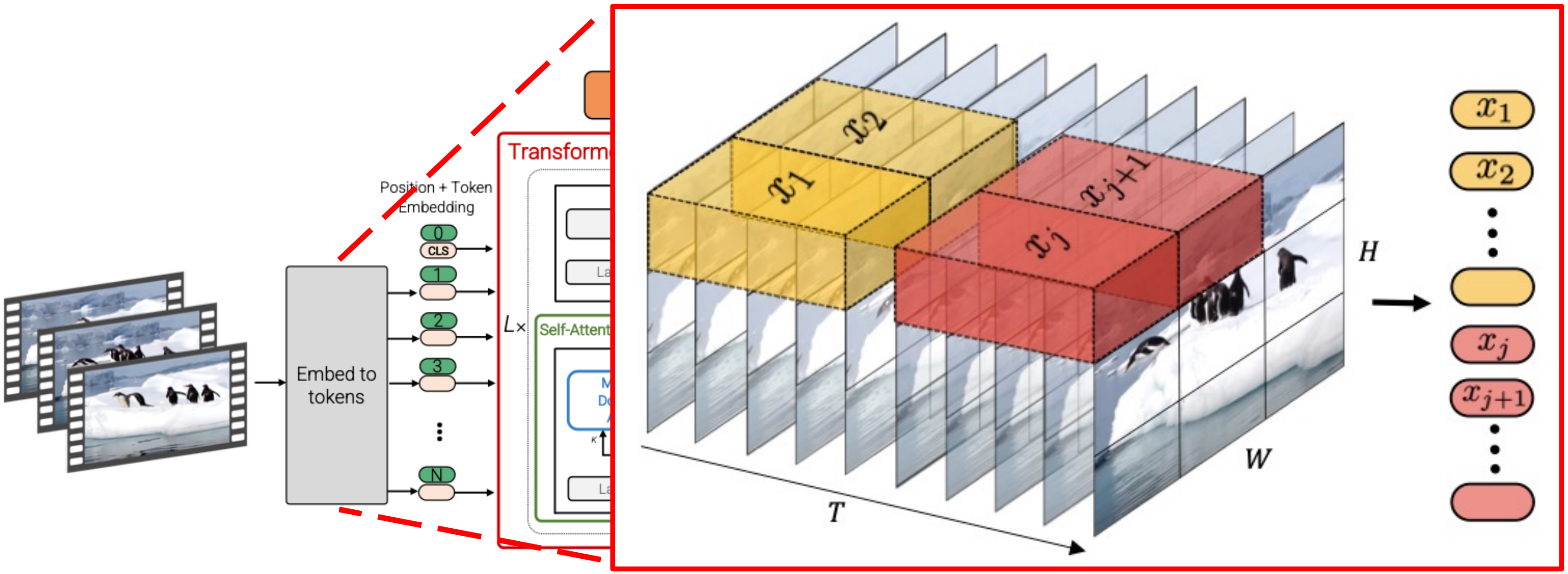
- Executing **3D CNNs** is computationally expensive
  - I3D [Carreira and Zisserman, 2017] demands computation burden comparable to the state-of-the-art transformer models (100+ GFLOPs)
  - A line of research pursuing efficient 3D CNN architectures

- **Factorization** of 3D kernel
  - A **3D CNN kernel of size** $(\boldsymbol{P}\times\boldsymbol{M}\times\boldsymbol{N})$ **can be factorized** to two convolutions;
    - A spatial 2D kernel $(1\times M\times N)$ and a temporal 1D kernel $(P\times 1\times 1)$
  - **R2+1D** [Tran et al., 2018] and **P3D** [Qiu et al., 2017] directly adopts this idea to largely save FLOPs

- Application of **channel-wise separated convolutions**
  - **CSN** [Tran et al., 2019] shows the efficacy of separating channel interactions and spatiotemporal interactions
    - State-of-the-art performance is achieved with $\times 3$ less computations than I3D [Carreira and Zisserman, 2017]

Video Vision Transformer (ViViT) [Arnab & Dehghani et al., 2021]

- ViViT is a pure **transformer** framework for video classification
- **Tubelet embedding** (3D extension of ViT)
  - Extract non-overlapping, spatial-temporal tubes from input volume
  - Linearly project them into $\mathbb{R}^d$



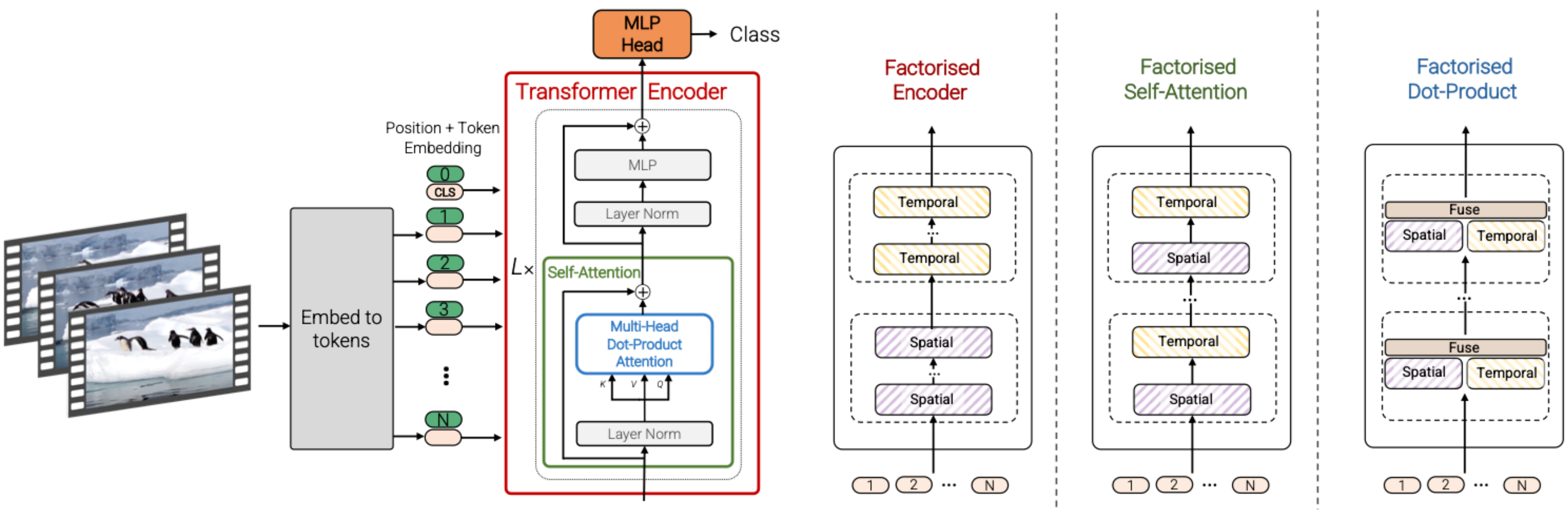
*source: [Arnab & Dehghani et al., 2021] A Video Vision Transformer, ICCV 2021

Video Vision Transformer (ViViT) [Arnab & Dehghani et al., 2021]

- Suggests different designs of **spatial & temporal attention**

  1. Joint Spatio-temporal attention

  - Simply forwards all pairwise interactions between all spatio-temporal tokens through transformer encoder
  - Unlike CNN, it can model long-range interactions across the video from the 1st layer
  - Requires quadratic complexity, $\mathcal{O}((n_h \cdot n_w \cdot n_t)^2)$, with number of tokens



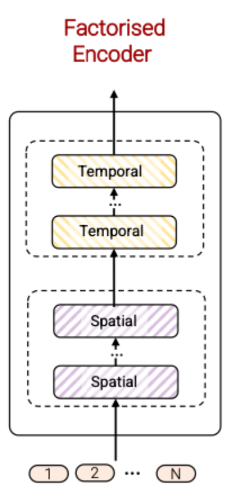
*source: [Arnab & Dehghani et al., 2021] A Video Vision Transformer, ICCV 2021

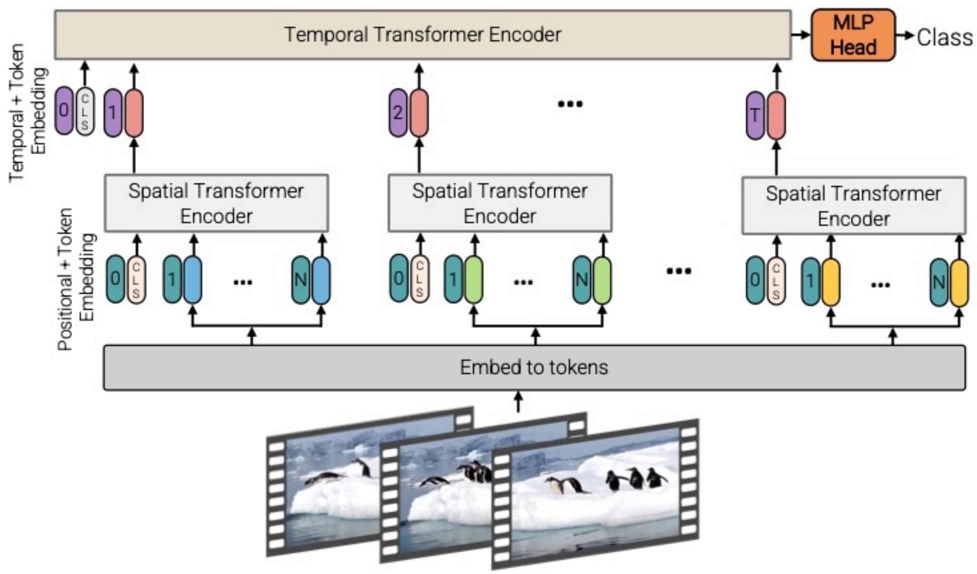
Video Vision Transformer (ViViT) [Arnab & Dehghani et al., 2021]

- Suggests different designs of **spatial & temporal attention**

2. Factorized encoder

- **Spatial encoder** models interactions between tokens from the same temporal index
- **Temporal encoder** models interactions between tokens from different temporal indices
- Requires more transformer layers (i.e., more parameters) than the joint design
- But less complexity, $\mathcal{O}((n_h \cdot n_w)^2 + n_t^2)$





*source: [Arnab & Dehghani et al., 2021] A Video Vision Transformer, ICCV 2021

Video Vision Transformer (ViViT) [Arnab & Dehghani et al., 2021]

- The factorized encoder design shows the best accuracy-to-FLOPs ratio
- However, the joint-design performs better and requires smaller number of parameters.

- Instead of factorizing the model, can we design approximate attention for both performance and FLOPs efficiency?

	K400	EK	FLOPs ($\times 10^9$)	Params ($\times 10^6$)	Runtime (ms)
Model 1: Spatio-temporal	80.0	43.1	455.2	88.9	58.9
Model 2: Fact. encoder	78.8	43.7	284.4	115.1	17.4
Model 3: Fact. self-attention	77.4	39.1	372.3	117.3	31.7
Model 4: Fact. dot product	76.3	39.5	277.1	88.9	22.9
Model 2: Ave. pool baseline	75.8	38.8	283.9	86.7	17.3

Comparison between model variants

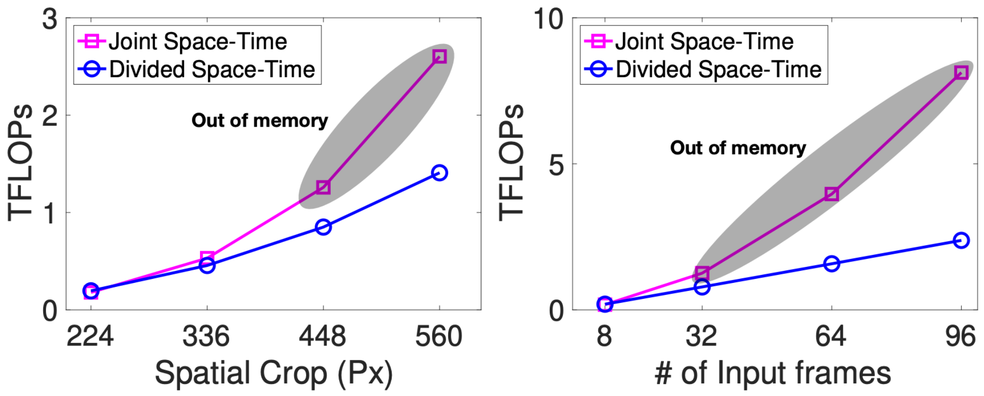
Method	Top 1	Top 5	Views	TFLOPs
bIVNet [19]	73.5	91.2	–	–
STM [33]	73.7	91.6	–	–
TEA [42]	76.1	92.5	10×3	2.10
TSM-ResNeXt-101 [43]	76.3	–	–	–
I3D NL [75]	77.7	93.3	10×3	10.77
CorrNet-101 [70]	79.2	–	10×3	6.72
ip-CSN-152 [66]	79.2	93.8	10×3	3.27
LGD-3D R101 [51]	79.4	94.4	–	–
SlowFast R101-NL [21]	79.8	93.9	10×3	7.02
X3D-XXL [20]	80.4	94.6	10×3	5.82
TimeSformer-L [4]	80.7	**94.7**	1×3	7.14
ViViT-L/16x2 FE	80.6	92.7	1×1	3.98
ViViT-L/16x2 FE	**81.7**	93.8	1×3	11.94

Kinetics-400 dataset benchmark

*source: [Arnab & Dehghani et al., 2021] A Video Vision Transformer, ICCV 2021

Brute-force joint spatial-temporal attention is intractable for transformers

- Due to the **quadratic complexity** with respect to inputs
- This motivates the development of more efficient attention scheme
  - Time-Space Transformer (TimeSformer) [Bertasius et al., 2021]
  - Video Swin Transformer [Liu et al., 2021]

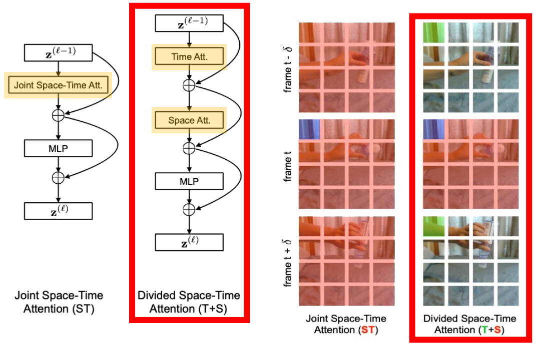


Video classification cost in TFLOPs

Time-Space Transformer (TimeSformer) [Bertasius et al., 2021]

- Proposes **divided space-time attention**
  - Instead of exhaustively comparing all pairs of patches (i.e., joint space-time attention), it separately applies temporal attention and spatial attention one after the other
- **Temporal attention**
  - Each patch (blue) is compared only with the patches at the same spatial location in other frames (green)
  - Initialized to zero (so that function as identity mapping in early training stages)
- **Spatial attention**
  - Each patch (blue) is compared only with the patches within the same frame (red)

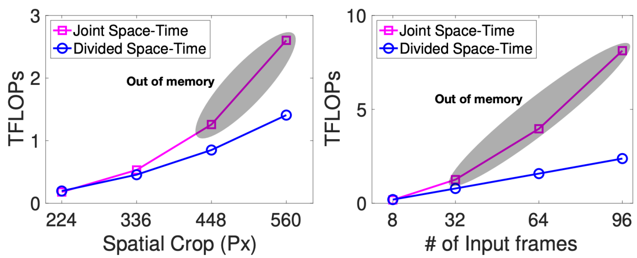
- Designs may look similar to ViViT (model 3) in a big picture, however, implementation details differ including 1) **time– then–space** att., 2) **zero initializations** for temporal layers



*source: [Bertasius et al. 2021] Is Space-Time Attention All You Need for Video Understanding?, ICML 2021

Time-Space Transformer (TimeSformer) [Bertasius et al., 2021]

- Divided space-time attention leads to dramatic computational savings with respect to spatial resolution/video length
- Outperforms SOTA models while requiring less computational complexity
  - $O(S^2T) + O(ST^2)$ instead of $O(S^2T^2)$



Method	Top-1	Top-5	TFLOPs
R(2+1)D (Tran et al., 2018)	72.0	90.0	17.5
bLVNet (Fan et al., 2019)	73.5	91.2	0.84
TSM (Lin et al., 2019)	74.7	N/A	N/A
S3D-G (Xie et al., 2018)	74.7	93.4	N/A
Oct-I3D+NL (Chen et al., 2019)	75.7	N/A	0.84
D3D (Stroud et al., 2020)	75.9	N/A	N/A
I3D+NL (Wang et al., 2018b)	77.7	93.3	10.8
ip-CSN-152 (Tran et al., 2019)	77.8	92.8	3.2
CorrNet (Wang et al., 2020a)	79.2	N/A	6.7
LGD-3D-101 (Qiu et al., 2019)	79.4	94.4	N/A
SlowFast (Feichtenhofer et al., 2019b)	79.8	93.9	7.0
X3D-XXL (Feichtenhofer, 2020)	80.4	94.6	5.8
TimeSformer	78.0	93.7	**0.59**
TimeSformer-HR	79.7	94.4	5.11
TimeSformer-L	**80.7**	**94.7**	7.14

3D CNNs

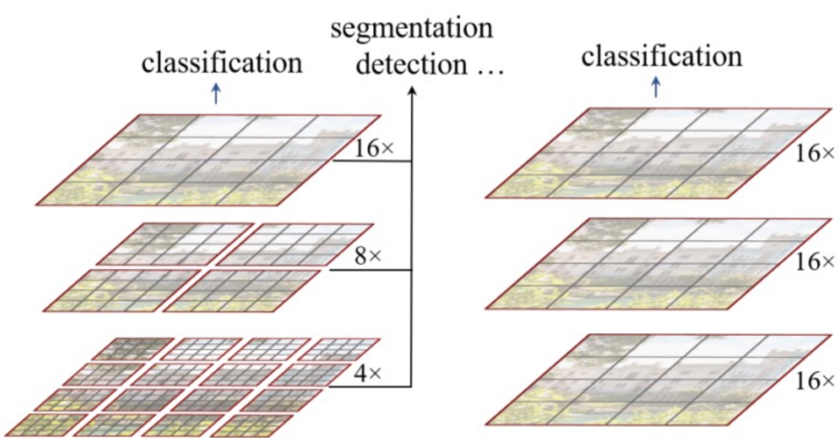
TimeSformer

Kinetics-400 dataset benchmark

*source: [Bertasius et al. 2021] Is Space-Time Attention All You Need for Video Understanding?, ICML 2021

Video Swin Transformer [Liu et al., 2021]

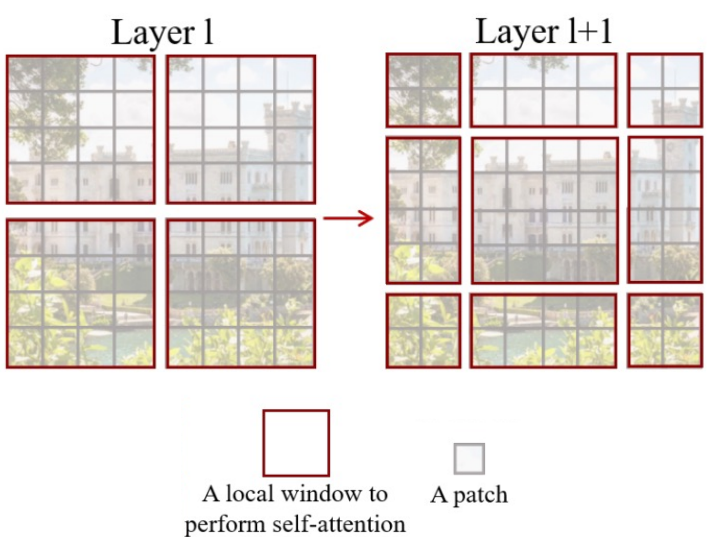
- Recall: **Swin Transformer** [Liu et al., 2021]
  - Design of a hierarchical structure
  - Various spatial resolutions (e.g., patch-shape) can be handled via shifted windows
  - Efficient self-attention computation by using shifted windows scheme
  - Concatenating 2 × 2 neighboring patches for downsampling operation
  - Powerful performances in dense prediction tasks
    e.g., object detection and semantic segmentation



(a) Swin Transformer (ours)

(b) ViT

Shifted window scheme

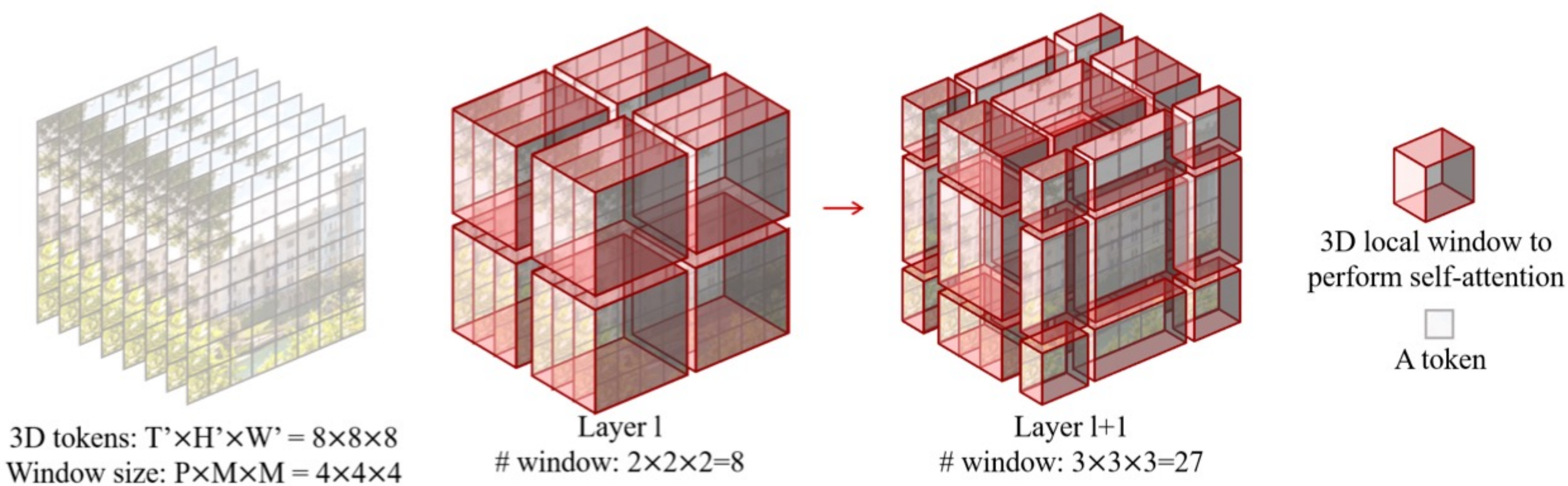


*source: [Liu et al. 2021] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021

Video Swin Transformer [Liu et al., 2021]

- In videos, pixels that are closer to each other in spatiotemporal distance are more likely to be correlated (i.e., spatiotemporal locality)
- Thus, **local** attention computation well approximates spatiotemporal self-attention
- Video Swin Transformer is a spatial-**temporal** adaptation of Swin Transformer

  i.e., extension from spatial locality to **spatial-temporal locality**



*source: [Liu et al. 2021] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021

Video Swin Transformer [Liu et al., 2021]

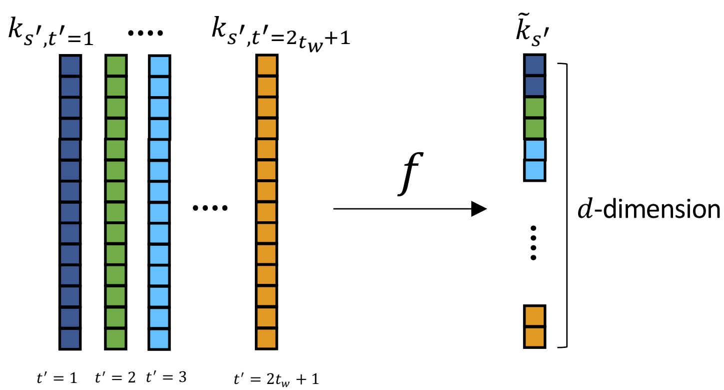
- Outperforms SOTA 3D CNN models while requiring smaller computation costs for inference
- Also outperforms SOTA transformer-based models while requiring half less computational costs

Method	Pretrain	Top-1	Top-5	Views	FLOPs	Param	
R(2+1)D [37]	-	72.0	90.0	10 × 1	75	61.8	3D CNNs
I3D [6]	ImageNet-1K	72.1	90.3	-	108	25.0	
NL I3D-101 [40]	ImageNet-1K	77.7	93.3	10 × 3	359	61.8	
ip-CSN-152 [36]	-	77.8	92.8	10 × 3	109	32.8	
CorrNet-101 [39]	-	79.2	-	10 × 3	224	-	
SlowFast R101+NL [13]	-	79.8	93.9	10 × 3	234	59.9	
X3D-XXL [12]	-	80.4	94.6	10 × 3	144	20.3	
MViT-B, 32×3 [10]	-	80.2	94.4	1 × 5	170	36.6	Transformer-based models
MViT-B, 64×3 [10]	-	81.2	95.1	3 × 3	455	36.6	
TimeSformer-L [3]	ImageNet-21K	80.7	94.7	1 × 3	2380	121.4	
ViT-B-VTN [29]	ImageNet-21K	78.6	93.7	1 × 1	4218	11.04	
ViViT-L/16x2 [1]	ImageNet-21K	80.6	94.7	4 × 3	1446	310.8	
ViViT-L/16x2 320 [1]	ImageNet-21K	81.3	94.7	4 × 3	3992	310.8	
ip-CSN-152 [36]	IG-65M	82.5	95.3	10 × 3	109	32.8	
ViViT-L/16x2 [1]	JFT-300M	82.8	95.5	4 × 3	1446	310.8	
ViViT-L/16x2 320 [1]	JFT-300M	83.5	95.5	4 × 3	3992	310.8	
ViViT-H/16x2 [1]	JFT-300M	84.8	95.8	4 × 3	8316	647.5	
Swin-T	ImageNet-1K	78.8	93.6	4 × 3	88	28.2	Ours
Swin-S	ImageNet-1K	80.6	94.5	4 × 3	166	49.8	
Swin-B	ImageNet-1K	80.6	94.6	4 × 3	282	88.1	
Swin-B	ImageNet-21K	82.7	95.5	4 × 3	282	88.1	
Swin-L	ImageNet-21K	83.1	95.9	4 × 3	604	197.0	
Swin-L (384↑)	ImageNet-21K	84.6	96.5	4 × 3	2107	200.0	
Swin-L (384↑)	ImageNet-21K	**84.9**	**96.7**	10 × 5	2107	200.0	

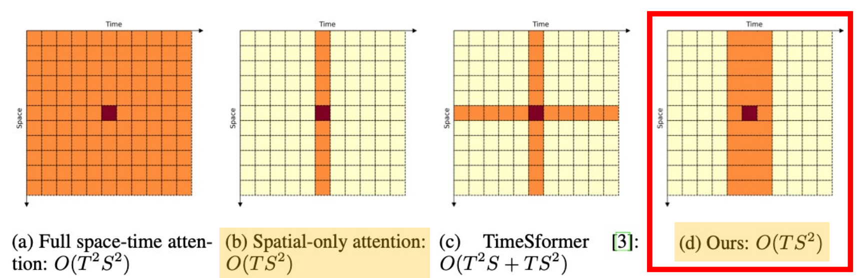
*source: [Liu et al. 2021] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021

X-ViT [Bulat et al., 2021]

- **Space-time mixing attention—$\boldsymbol{O(TS^2)}$ complexity**
  - The following architectural changes in X-ViT **reduce the full quadratic complexity** $\boldsymbol{O(T^2S^2)}$ to the proposed $\boldsymbol{O(TS^2)}$
    1. Restricting **attentions within a temporal window** of $[t - t_w, t + t_w]$ for each $q_{s,t}$
       → The complexity becomes $O(T(2t_w + 1)^2S^2)$
    2. Instead of individual space-time keys, the **time compression** $\boldsymbol{f}$ is applied such that a single attention is considered over time with $\tilde{k}_{s'} \triangleq f([k_{s',t-t_w}; \ldots; k_{s',t+t_w}])$
    3. Instead of general affine transforms, **"shift trick"** is employed as the implementatio n of f to further save computations:
       - Given a key $k_{s',t'} \in \mathbb{R}^d$, split its channels into $(2t_w + 1)$ segments, then pick t he $t' \in [1, 2t_w + 1]$th index to form the final $\tilde{k}_{s'}$ → The complexity becomes $O(T\cancel{(2t_w + 1)}S^2)$
         Can be disregarded as $\boldsymbol{2t_w + 1}$ is a small constant



The shift trick in X-ViT



X-ViT

*Red is the query vector
*Orange is the key vector that the query vector attends to

*source: [Bulat et al. 2021] Space-time Mixing Attention for Video Transformer, NeurIPS 2021

X-ViT [Bulat et al., 2021]

- Achieves comparable performance to SOTA models while requiring significantly lower computational complexity
  - **X-ViT (16-frames, 850 GFLOPs)** achieves performance comparable to heavy-weight variants of TimeSformer (96-frames, 7140 GFLOPs) and ViViT (32 frames, 4340 GFLOPs)

- Allows for an efficient approximation of local space-time attention at no extra cost

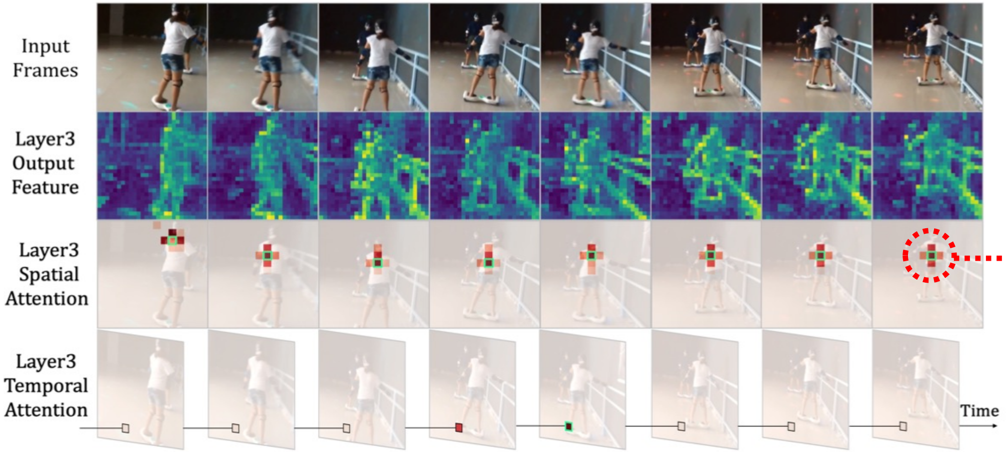
Method	Top-1	Top-5	# Frames	Views	Params	FLOPs ($\times 10^9$)
bLVNet [14]	73.5	91.2	24×2	3×3	25M	840
STM [19]	73.7	91.6	16	-	24M	-
TEA [25]	76.1	92.5	16	10×3	25.6M	2,100
TSM R50 [26]	74.7	-	16	10×3	25.6M	650
I3D NL [44]	77.7	93.3	128	10×3	-	10,800
CorrNet-101 [40]	79.2	-	32	10×3	-	6,700
ip-CSN-152 [38]	79.2	93.8	8	10×3	-	3,270
LGD-3D R101 [31]	79.4	94.4	16	-	-	-
SlowFast 8×8 R101+NL [16]	78.7	93.5	8	10×3	-	3,480
SlowFast 16×8 R101+NL [16]	79.8	93.9	16	10×3	-	7,020
X3D-XXL [15]	80.4	94.6	-	10×3	20.3M	5,823
TimeSformer-L [3]	**80.7**	94.7	96	1×3	121M	7,140
ViViT-L/16x2 [1]	80.6	94.7	32	4×3	312M	17,352
X-ViT (Ours)	78.5	93.7	8	1×3	92M	425
X-ViT (Ours)	79.4	93.9	8	2×3	92M	850
X-ViT (Ours)	80.2	94.7	16	1×3	92M	850
X-ViT (Ours)	**80.7**	**94.7**	16	2×3	92M	1700

*source: [Bulat et al. 2021] Space-time Mixing Attention for Video Transformer, NeurIPS 2021

3D convolutions vs. Vision Transformers

- **3D convolutions**
  - Pro: Can capture detailed local spatiotemporal features to suppress local redundancy
  - Con: Inefficient to capture global (long-range) dependency due to limited receptive field
- **Vision Transformers**
  - Pro: Can capture global (long-range) dependency by self-attention mechanism
  - Con: Inefficient to encode spatiotemporal feature in shallow layers (local redundancy) and requires explicit position embedding (which could be sub-optimal for videos)

Integrating merits of both, a unified model has been proposed



Visualizations of TimeSformer [Bertasius et al., 2021]

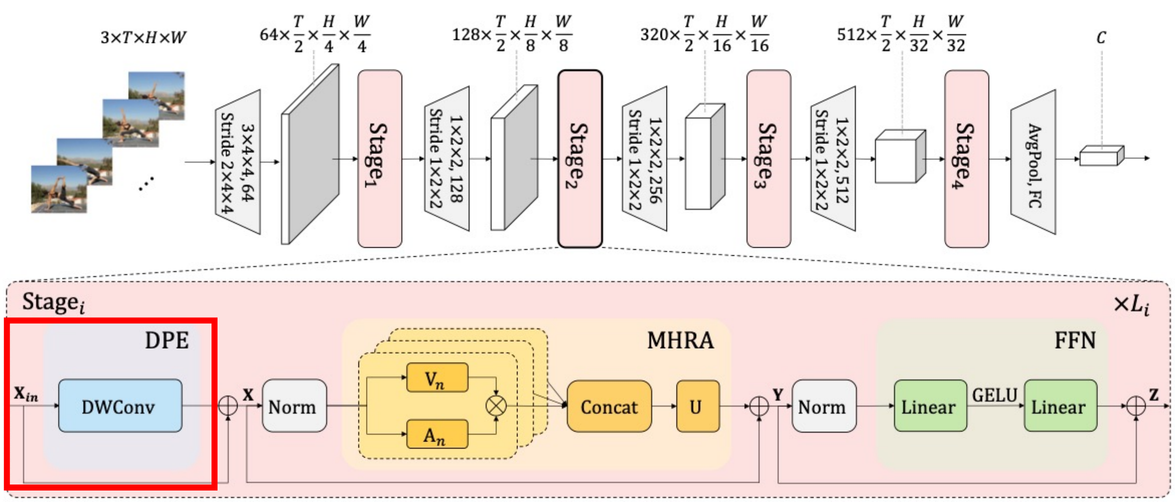
- Vision transformer learns local repre sentations **with redundant global** at tention
- This wastes **large computation** to en code only very local spatiotemporal representations

UniFormer [Li et al., 2022]

- **Dynamic Position Embedding (DPE)**
  - Instead of explicit position embedding, dynamic position embedding (DPE) is used:

$$\mathrm{DPE}(\mathbf{X}_{in}) = \mathrm{DWConv}(\mathbf{X}_{in})$$

  - DPE dynamically integrates 3D position information into all tokens
  - $\boldsymbol{DWConv}$ is a simple 3D depth-wise convolution with zero paddings
    - Shared parameters & locality of convolution tackles permutation-invariance
    - In CPE, zero paddings help tokens on the borders be aware of their absolute positions
    - That is, all tokens progressively encode their position information via querying their neighbor



*source: [Li et al. 2022] Uniformer: Unified Transformer for Efficient Spatiotemporal Representation Learning, ICLR 2022

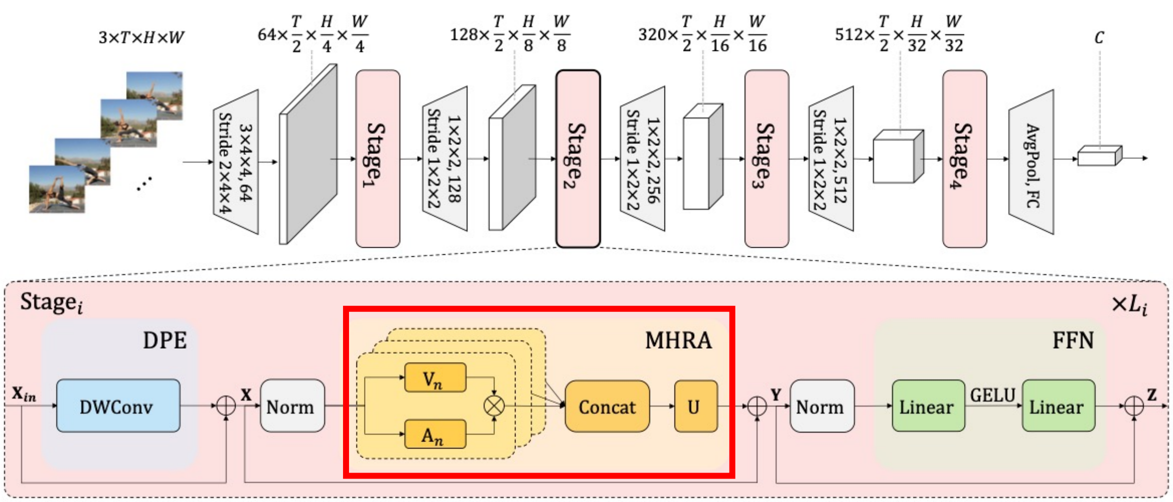
UniFormer [Li et al., 2022]

- **Multi-Head Relation Aggregator (MHRA)**

  1) Local MHRA (for shallow layers)

  - Aim for shallow layers is to learn detailed video representation from local spatiotemporal context to reduce redundancy
  - Design token affinity to be local learnable parameter matrix, which depends only on relative 3D position between tokens
  - RA learns local spatiotemporal affinity between one anchor token X_i and other tokens in the small tube $\Omega_i^{t\times h\times w}$

$$\mathrm{A}_n^{local}(\mathbf{X}_i, \mathbf{X}_j) = a_n^{i-j}, \quad where \ \ j \in \Omega_i^{t\times h\times w}$$



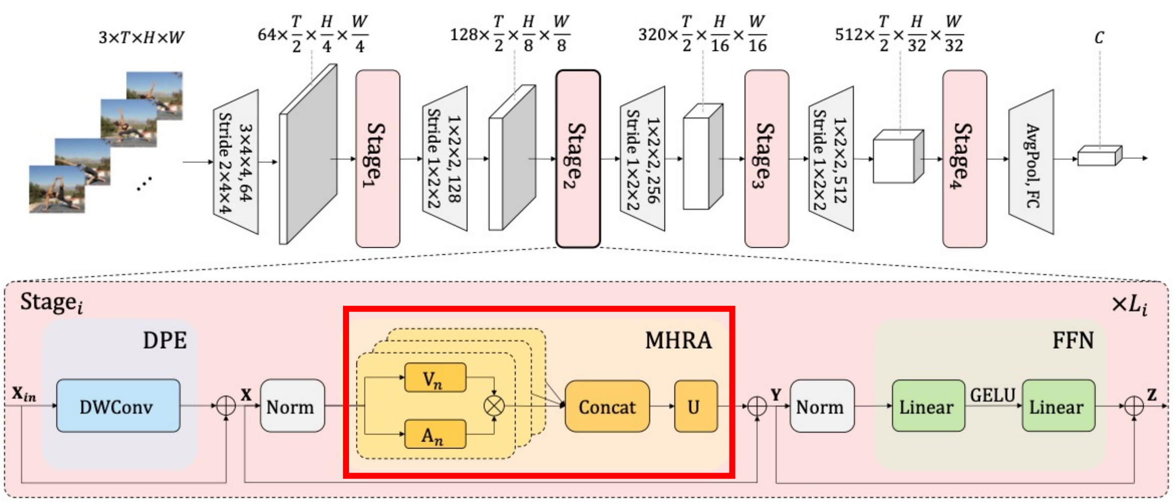
*source: [Li et al. 2022] Uniformer: Unified Transformer for Efficient Spatiotemporal Representation Learning, ICLR 2022

UniFormer [Li et al., 2022]

- **Multi-Head Relation Aggregator (MHRA)**
  2) **Global MHRA** (for deep layers)
  - Aim for deep layers is to capture long-term token dependency in global video clip
  - Design token affinity via comparing content similarity among all tokens in global view

$$\mathrm{A}_n^{global}(\mathbf{X}_i, \mathbf{X}_j) = \frac{e^{Q_n(\mathbf{X}_i)^T K_n(\mathbf{X}_j)}}{\sum_{j' \in \Omega_{T\times H\times W}} e^{Q_n(\mathbf{X}_i)^T K_n(\mathbf{X}_{j'})}}$$

  - X_j can be any token in global 3D tube $\Omega_{T\times H\times W}$
  - $Q_n(\cdot)$ and $K_n(\cdot)$ are two different linear transformations



*source: [Li et al. 2022] Uniformer: Unified Transformer for Efficient Spatiotemporal Representation Learning, ICLR 2022

UniFormer [Li et al., 2022]

- Uniformer outperforms existing models with much fewer computational cost
- Achieves a preferable balance between computation and accuracy

Method	Pretrain	#Frame	GFLOPs	SSV1		SSV2	
				Top-1	Top-5	Top-1	Top-5
TSN(Wang et al., 2016)	IN-1K	16×1×1	66	19.9	47.3	30.0	60.5
TSM(Lin et al., 2019)	IN-1K	16×1×1	66	47.2	77.1	-	-
GST(Luo & Yuille, 2019)	IN-1K	16×1×1	59	48.6	77.9	62.6	87.9
MSNet(Kwon et al., 2020)	IN-1K	16×1×1	101	52.1	82.3	64.7	89.4
CT-Net(Li et al., 2021a)	IN-1K	16×1×1	75	52.5	80.9	64.5	89.3
CT-Net$_{EN}$(Li et al., 2021a)	IN-1K	8+12+16+24	280	56.6	83.9	67.8	91.1
TDN(Wang et al., 2020b)	IN-1K	16×1×1	72	53.9	82.1	65.3	89.5
TDN$_{EN}$(Wang et al., 2020b)	IN-1K	8+16	198	56.8	84.1	68.2	91.6
TimeSformer-HR(Bertasius et al., 2021)	IN-21K	16×3×1	5109	-	-	62.5	-
X-ViT(Bulat et al., 2021)	IN-21K	32×3×1	1270	-	-	65.4	90.7
Mformer-L(Patrick et al., 2021)	K400	32×3×1	3555	-	-	68.1	91.2
ViViT-L(Arnab et al., 2021)	K400	16×3×4	11892	-	-	65.4	89.8
MViT-B,64×3(Fan et al., 2021)	K400	64×1×3	1365	-	-	67.7	90.9
MViT-B-24,32×3(Fan et al., 2021)	K600	32×1×3	708	-	-	68.7	91.5
Swin-B(Liu et al., 2021b)	K400	32×3×1	963	-	-	69.6	92.7
Our UniFormer-S	K400	16×1×1	42	53.8	81.9	63.5	88.5
Our UniFormer-S	K600	16×1×1	42	54.4	81.8	65.0	89.3
Our UniFormer-S	K400	16×3×1	125	57.2	84.9	67.7	91.4
Our UniFormer-S	K600	16×3×1	125	57.6	84.9	69.4	92.1
Our UniFormer-B	K400	16×3×1	290	59.1	86.2	70.4	92.8
Our UniFormer-B	K600	16×3×1	290	58.8	86.5	70.2	**93.0**
Our UniFormer-B	K400	32×3×1	777	60.9	87.3	**71.2**	92.8
Our UniFormer-B	K600	32×3×1	777	**61.0**	**87.6**	**71.2**	92.8

*source: [Li et al. 2022] Uniformer: Unified Transformer for Efficient Spatiotemporal Representation Learning, ICLR 2022

Table of Contents

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

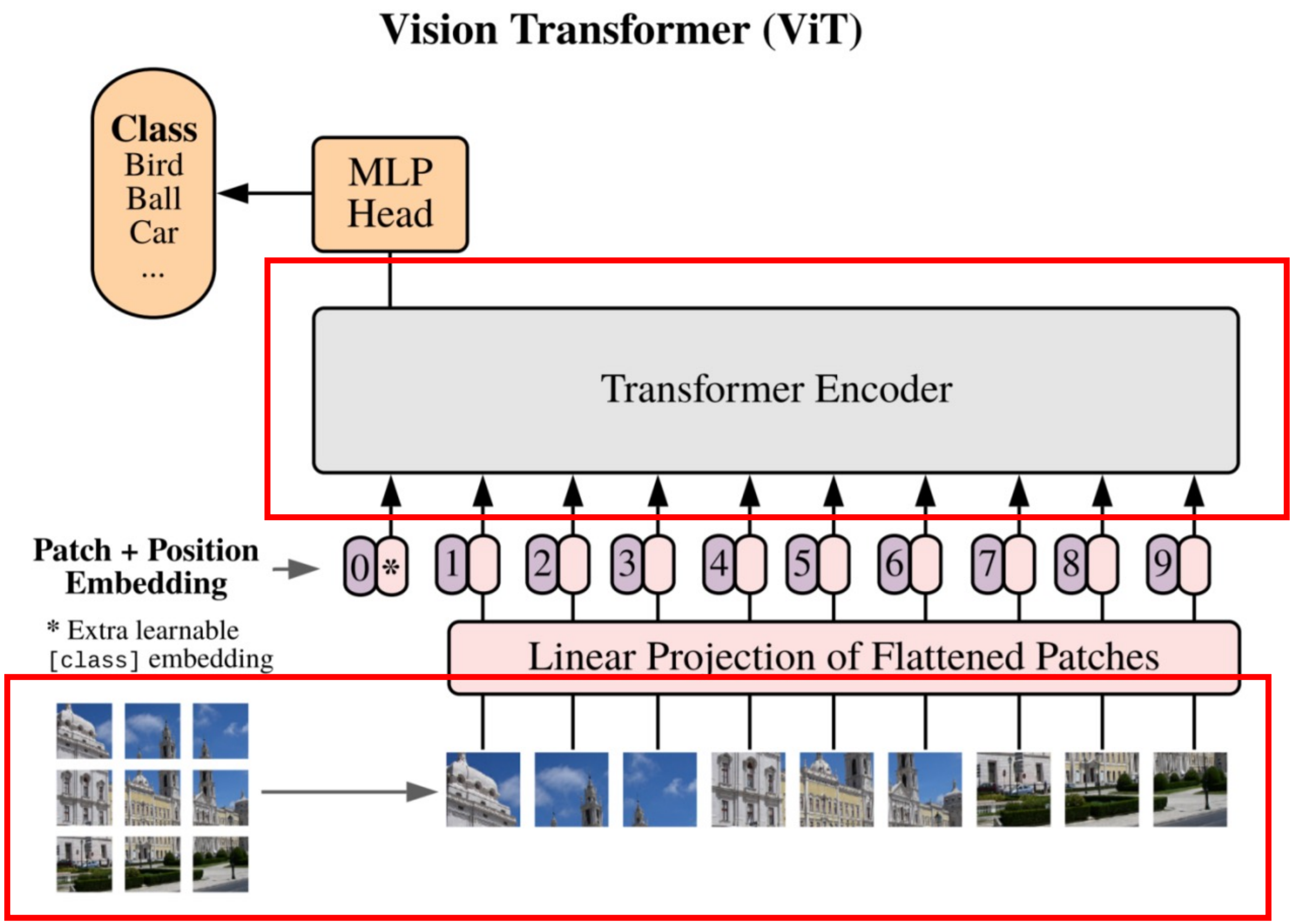
- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

Question: Is the success of **Vision Transformers** due to

1. the powerful Transformer architecture?
2. using patches as the input representation?



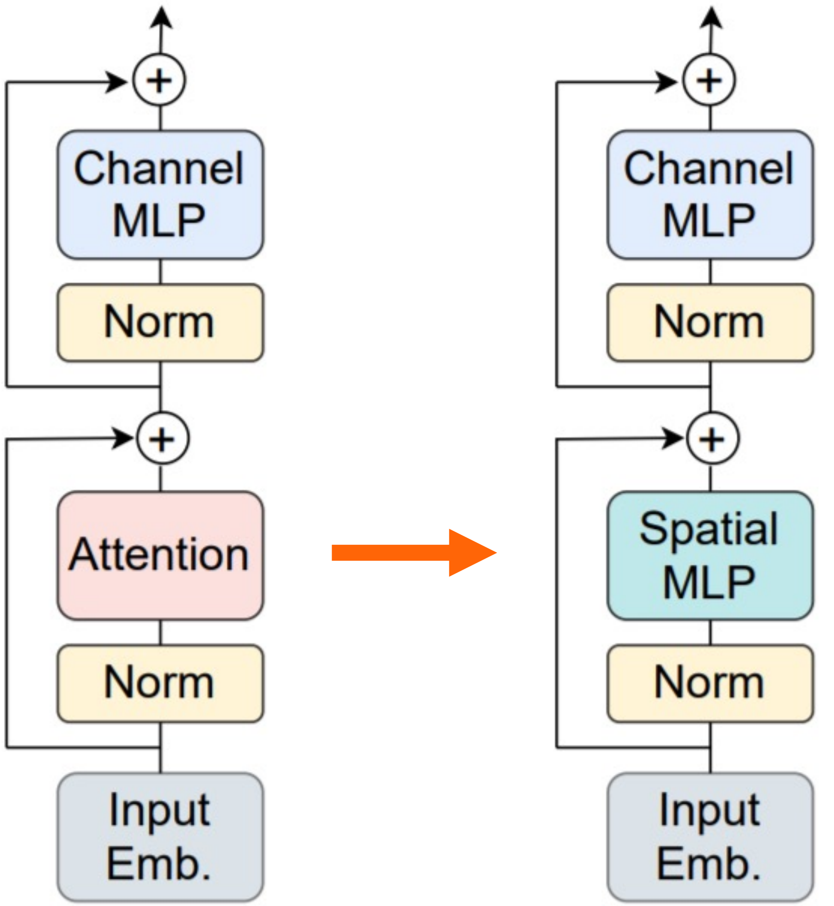
*source: [Dosovitskiy et al. 2021] An image is worth 16x16 words: Transformers for image recognition at scale, ICLR 2021

- **Tolstikhin et al.** (2021) suggests MLP module as an alternative of self-attention module
  - For a given Image I,

$$Z = \mathrm{MLP}\big(\mathrm{Norm}(Y)\big) + Y$$
$$= \sigma(\mathrm{Norm}(Y)W_1)W_2 + Y$$

$$Y = \mathrm{SelfAttn}\big(\mathrm{Norm}(X)\big) + X$$

$$X = \mathrm{InputEmbed}(I)$$



$$Z = \mathrm{MLP}\big(\mathrm{Norm}(Y)\big) + Y$$
$$= \sigma(\mathrm{Norm}(Y)W_1)W_2 + Y$$

$$Y = \mathrm{MLP}\big(\mathrm{Norm}(X)\big) + X$$

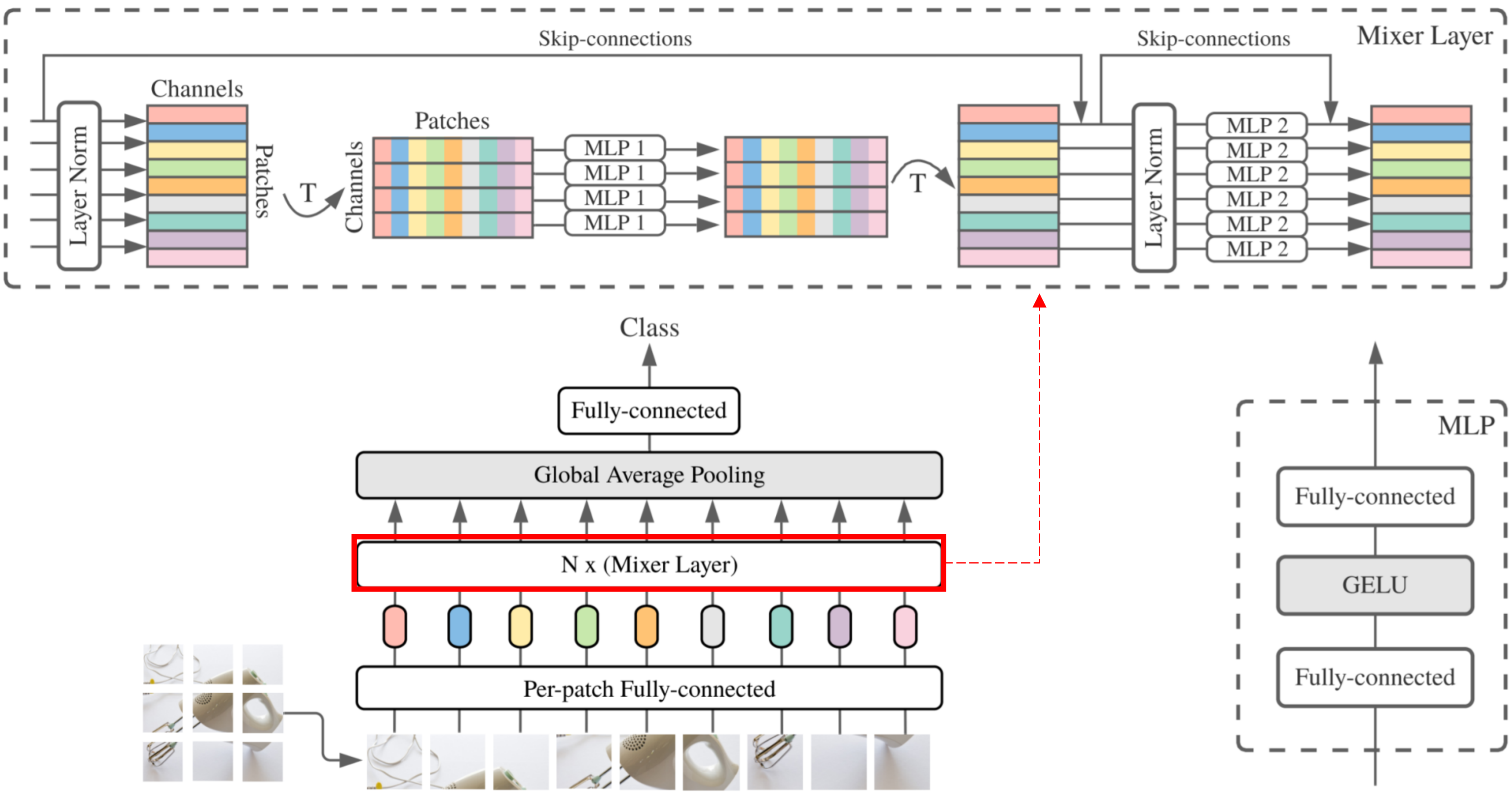
$$X = \mathrm{InputEmbed}(I)$$

Transformer architectures **MLP architectures**

*source : [Yu et al., 2021] MetaFormer is Actually What You Need for Vision, arxiv 2021

MLP-Mixer [Tolstikhin et al., 2021]

- Replacing the self-attention into MLP layers
- Removing position embedding & [class] token
- Mixing spatial & channel dimension separately



MLP-Mixer [Tolstikhin et al., 2021]

- Replacing the self-attention into MLP layers
- Removing position embedding & [class] token
- Mixing spatial & channel dimension separately

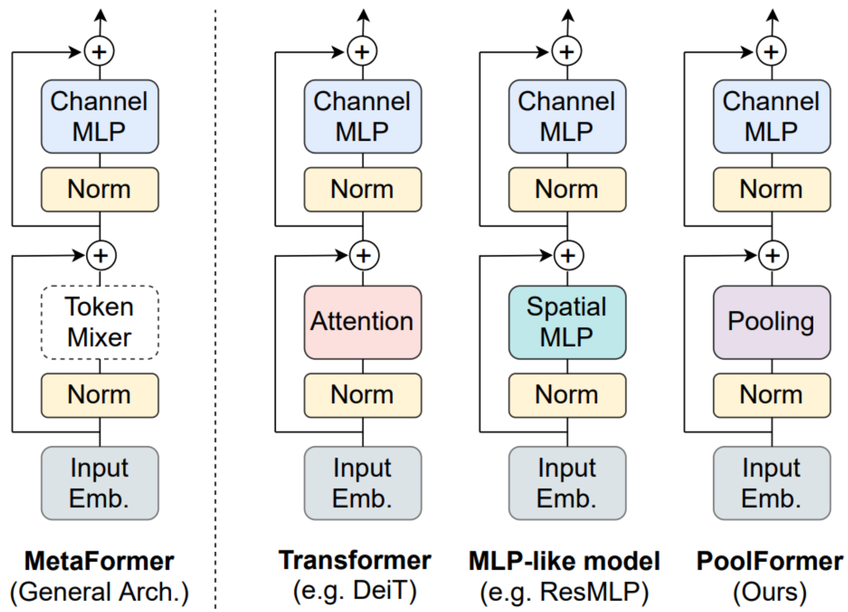
- **MLP-Mixer** shows competitive performances compared to Vision Transformers

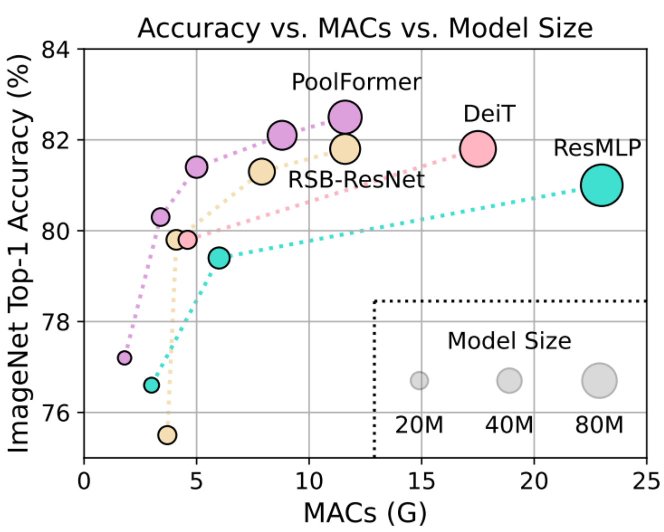
	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
			Pre-trained on ImageNet-21k (public)			
HaloNet [51]	85.8	—	—	—	120	0.10k
Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
BiT-R152x4 [22]	85.39	—	94.04	70.64	26	0.94k
			Pre-trained on JFT-300M (proprietary)			
NFNet-F4+ [7]	89.2	—	—	—	46	1.86k
Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
BiT-R152x4 [22]	87.54	90.54	95.33	76.29	26	9.90k
ViT-H/14 [14]	88.55	90.72	95.97	77.63	15	2.30k

General Patch-based Architectures: MetaFormers

- **MetaFormers [Yu et al, 2022]** reveals that patch-based architecture with any token-mixing method can work well

- For example, replacing self-attention with sophisticated average pooling (**PoolFormer**) allows light-weight model in terms of both computations and # parameters





*source : [Yu et al., 2022] MetaFormer is Actually What You Need for Vision, arxiv 2021

- **MetaFormers [Yu et al, 2022]** reveals that patch-based architecture with any token-mixing method can work well

- For example, replacing self-attention with sophisticated average pooling (**PoolFormer**) allows light-weight model in terms of both computations and # parameters
  - Sophisticated design of **token-mixing is important** such as pooling sizes
  - **Mixing different strategies** (e.g., pooling + attention) is also effective

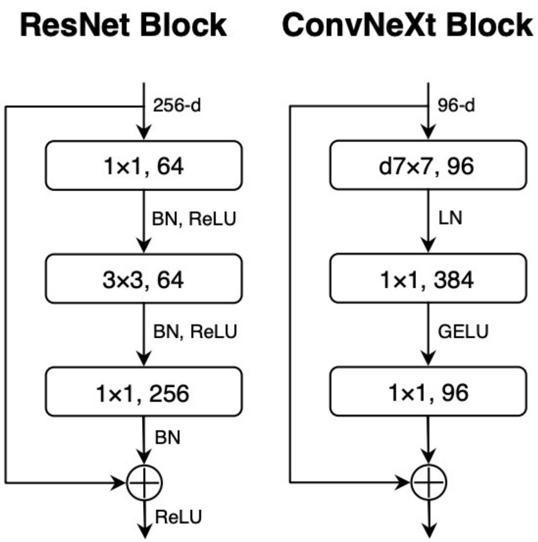
Stage	#Tokens	Layer Specification		PoolFormer S12	PoolFormer S24	PoolFormer S36	PoolFormer M36	PoolFormer M48
1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	Patch Size	7×7, stride 4				
			Embed. Dim.	64			96	
		PoolFormer Block	Pooling Size	3×3, stride 1				
			MLP Ratio	4				
			# Block	2	4	6	6	8
2	$\frac{H}{8} \times \frac{W}{8}$	Patch Embedding	Patch Size	3×3, stride 2				
			Embed. Dim.	128			192	
		PoolFormer Block	Pooling Size	3×3, stride 1				
			MLP Ratio	4				
			# Block	2	4	6	6	8
3	$\frac{H}{16} \times \frac{W}{16}$	Patch Embedding	Patch Size	3×3, stride 2				
			Embed. Dim.	320			384	
		PoolFormer Block	Pooling Size	3×3, stride 1				
			MLP Ratio	4				
			# Block	6	12	18	18	24
4	$\frac{H}{32} \times \frac{W}{32}$	Patch Embedding	Patch Size	3×3, stride 2				
			Embed. Dim.	512			768	
		PoolFormer Block	Pooling Size	3×3, stride 1				
			MLP Ratio	4				
			# Block	2	4	6	6	8
Parameters (M)				11.9	21.4	30.8	56.1	73.4
MACs (G)				1.8	3.4	5.0	8.8	11.6

Ablation	Variant	Params (M)	MACs (G)	Top-1 (%)
Baseline	None (PoolFormer-S12)	11.9	1.8	77.2
Token mixers	Pooling → Identity mapping	11.9	1.8	74.3
	Pooling → Global random matrix* (extra 21M frozen parameters)	11.9	3.3	75.8
	Pooling → Depthwise Convolution [9, 38]	11.9	1.8	78.1
	Pooling size 3 → 5	11.9	1.8	77.2
	Pooling size 3 → 7	11.9	1.8	77.1
	Pooling size 3 → 9	11.9	1.8	76.8
Normalization	Modified Layer Normalization† → Layer Normalization [1]	11.9	1.8	76.5
	Modified Layer Normalization† → Batch Normalization [28]	11.9	1.8	76.4
	Modified Layer Normalization† → None	11.9	1.8	46.1
Activation	GELU [25] → ReLU [41]	11.9	1.8	76.4
	GELU → SiLU [18]	11.9	1.8	77.2
Other components	Residual connection [25] → None	11.9	1.8	0.1
	Channel MLP → None	2.5	0.2	5.7
Hybrid Stages	[Pool, Pool, Pool, Pool] → [Pool, Pool, Pool, Attention]	14.0	1.9	78.3
	[Pool, Pool, Pool, Pool] → [Pool, Pool, Attention, Attention]	16.5	2.5	81.0
	[Pool, Pool, Pool, Pool] → [Pool, Pool, Pool, SpatialFC]	11.9	1.8	77.5
	[Pool, Pool, Pool, Pool] → [Pool, Pool, SpatialFC, SpatialFC]	12.2	1.9	77.9

*source : [Yu et al., 2022] MetaFormer is Actually What You Need for Vision, arxiv 2021

General Patch-based Architectures: ConvNext

- **ConvNext [Liu et al, 2022]** reveals that introducing X-former (e.g., transformers) architectural characteristic to CNNs is effective
- **Patch-based input projection**
  - In the input layer of ResNet, a 7×7 convolution is applied (overlapping patches)
  - In vision transformers, a more aggressive strategy is used:
    - A linear transform of patch as tokens (i.e., non-overlapping convolution)
- **Wide feed-forward MLP**
  - Note that FFN in ViT is effectively 1×1 convolution with $4\times$ channel width as the input
  - Design principle is opposite to that of ResNet (i.e., the bottleneck block)



General Patch-based Architectures: ConvNext

- **ConvNext [Liu et al, 2022]** reveals that introducing X-former (e.g., transformers) architectural characteristic to CNNs is effective

- There are various design transfers from X-former to CNN in ConvNext (refer to the paper for details)

- Simply **transferring design principles from X-former to CNNs** could make them outperfor m vision transformers

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**

Part 1. Basics

- Evolution of CNN architectures
- Batch normalization and ResNet
- Attention module in CNNs
- Vision transformers

Part 2. Advanced Topics

- Toward automation of network design
- Flexible architectures
- Observational study on network architectures
- Deep spatial-temporal models

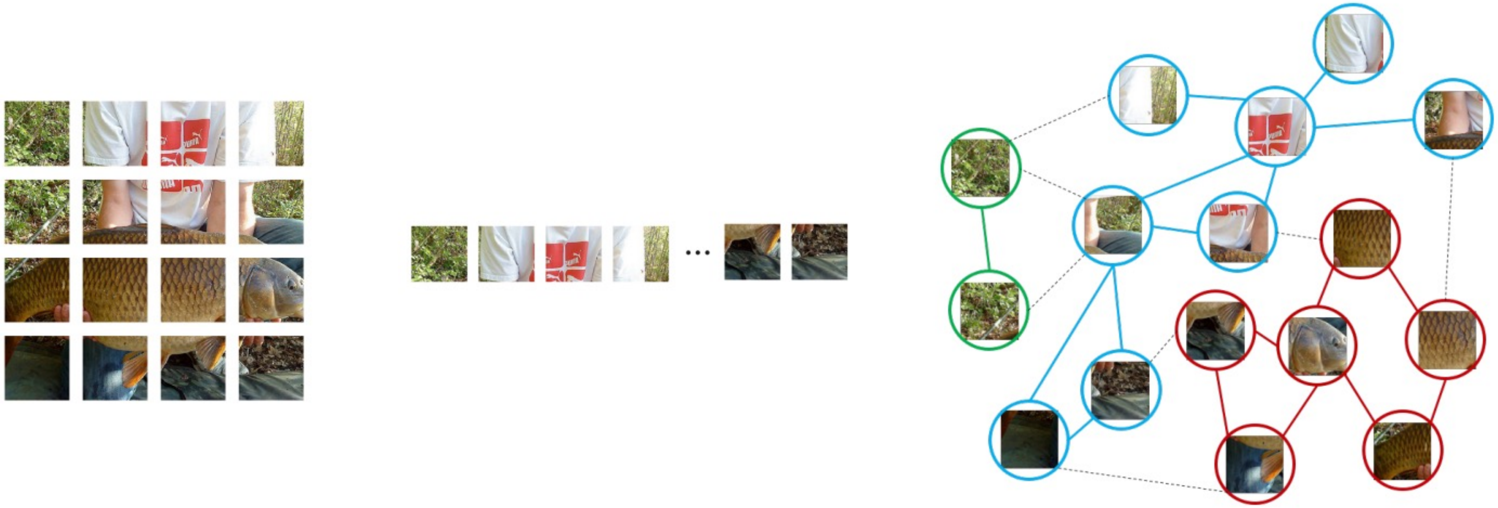
Part 3. Beyond CNNs and Vision Transformers

- Patch-based architectures for vision
- New design paradigms

- **VisionGNN [Han et al, 2022]**

Motivation: Can we go beyond **grid-based representation** of images?

- **Grid (and sequence) of image patches** can be views a **s a special case of graph**
- VisionGNN represents images as a **graph** $(\boldsymbol{V}, \boldsymbol{E})$ with image patch as nodes $(\boldsymbol{V})$ and learnable edges $(\boldsymbol{E})$



(a) Grid structure. (b) Sequence structure. (b) Graph structure.

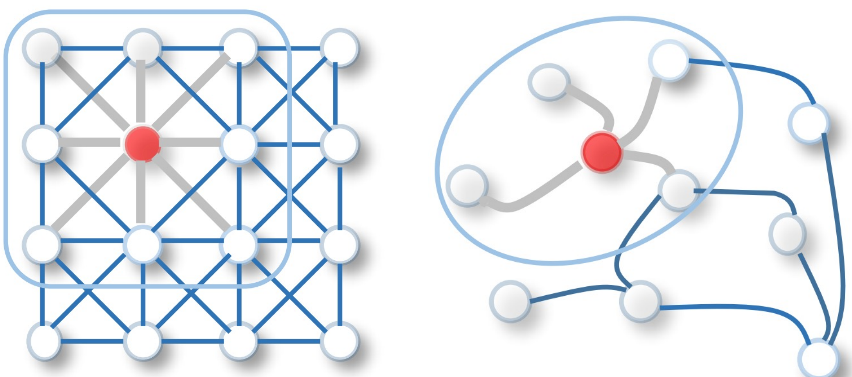
New design paradigms

- **VisionGNN [Han et al, 2022]**

Motivation: Can we go beyond **grid-based representation** of images?

- **Grid (and sequence) of image patches** can be views a **s a special case of graph**
- VisionGNN represents images as a **graph** $(\boldsymbol{V}, \boldsymbol{E})$ with image patch as nodes $(\boldsymbol{V})$ and learnable edges $(\boldsymbol{E})$

- For modeling graph-based representation, a new graph model base-on Graph Convolution Networks is proposed
  - Graph Convolution Networks
    - Graph convolutional operation aggregates value of the node features of neighbors (Note that there is no ordering between nodes)



$$\begin{aligned}\mathcal{G}' &= F(\mathcal{G}, \mathcal{W}) \\ &= Update(Aggregate(\mathcal{G}, W_{agg}), W_{update}),\end{aligned}$$

- **VisionGNN [Han et al, 2022]**

Motivation: Can we go beyond **grid-based representation** of images?

- **Grid (and sequence) of image patches** can be views a **s a special case of graph**
- VisionGNN represents images as a **graph** $(\boldsymbol{V}, \boldsymbol{E})$ with image patch as nodes $(\boldsymbol{V})$ and learnable edges $(\boldsymbol{E})$

- VisionGNN can outperform vision transformers and CNNs

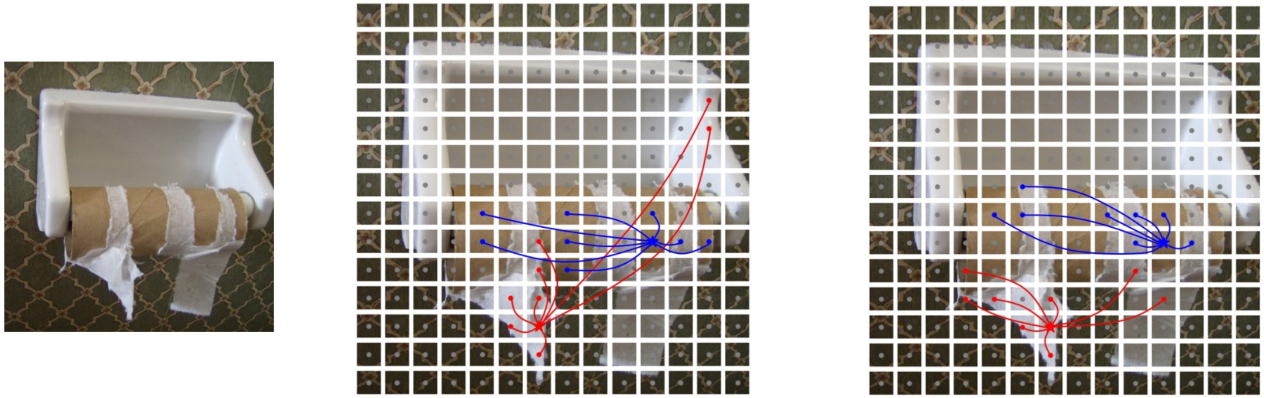
Model	Resolution	Params (M)	FLOPs (B)	Top-1	Top-5
♠ ResMLP-S12 conv3x3 [50]	224×224	16.7	3.2	77.0	-
♠ ConvMixer-768/32 [52]	224×224	21.1	20.9	80.2	-
♠ ConvMixer-1536/20 [52]	224×224	51.6	51.4	81.4	-
♦ ViT-B/16 [9]	384×384	86.4	55.5	77.9	-
♦ DeiT-Ti [51]	224×224	5.7	1.3	72.2	91.1
♦ DeiT-S [51]	224×224	22.1	4.6	79.8	95.0
♦ DeiT-B [51]	224×224	86.4	17.6	81.8	95.7
■ ResMLP-S24 [50]	224×224	30	6.0	79.4	94.5
■ ResMLP-B24 [50]	224×224	116	23.0	81.0	95.0
■ Mixer-B/16 [49]	224×224	59	11.7	76.4	-
★ ViG-Ti (ours)	224×224	7.1	1.3	**73.9**	**92.0**
★ ViG-S (ours)	224×224	22.7	4.5	**80.4**	**95.2**
★ ViG-B (ours)	224×224	86.8	17.7	**82.3**	**95.9**

- **VisionGNN [Han et al, 2022]**

Motivation: Can we go beyond **grid-based representation** of images?

- **Grid (and sequence) of image patches** can be views a **s a special case of graph**
- VisionGNN represents images as a **graph (V, E)** with image patch as nodes (V) and learnable edges (E)

- More importantly, **the graph structure** naturally provides **interpretability** in the hidden layers
  - **Earlier blocks** connects **low-level features** (e.g., colors) and local features
  - **Later blocks** connect **semantically-related** (e.g., same category) features



(a) Input image. (b) Graph connection in the 1st block. (c) Graph connection in the 12th block.

- **VisionGNN [Han et al, 2022]**

Motivation: Can we go beyond **grid-based representation** of images?

- **Grid (and sequence) of image patches** can be views a **s a special case of graph**
- VisionGNN represents images as a **graph** $(\boldsymbol{V}, \boldsymbol{E})$ with image patch as nodes $(\boldsymbol{V})$ and learnable edges $(\boldsymbol{E})$

- More importantly, **the graph structure** naturally provides **interpretability** in the hidden layers
  - **Earlier blocks** connects **low-level features** (e.g., colors) and local features
  - **Later blocks** connect **semantically-related** (e.g., same category) features

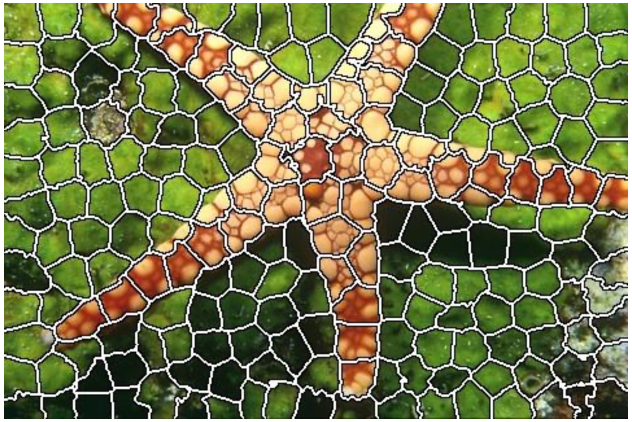
- However, **nodes are still regular-shaped** in VisionGNN
  - Can we make more flexible model?
  - **Treating each pixel as a node** which will result in **too many nodes (>10K)**

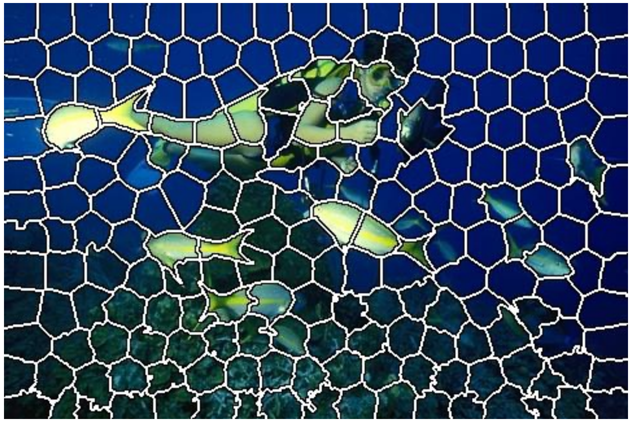
- **Context Clusters (CoC) [Ma et al, 2023]**

Motivation: Can we go beyond **grid-based patches** of images?

- Context Clusters view an image as a set of unorganized points and extract features via simplified clustering algorithm

- **n** points $\boldsymbol{P} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{d}}$ are clustered using **SuperPixel** method
  - **SuperPixel SLIC [Achanta et al., 2013]**
    - For inputs, **n** is the number of all pixels, however, an initial $\mathbf{4}\times\mathbf{4}$ **convolution** projects them to feature space, **reducing # points to** $\boldsymbol{\frac{n}{16}}$
    - For clustering $\boldsymbol{c}$ **centers** are evenly proposed, and each point is assigned to the nearest center (**feature cosine similarity** is used as the distance metric)
    - After clustering, each cluster can have **variable number** of points (even 0 is possible)





- **Context Clusters (CoC) [Ma et al, 2023]**

Motivation: Can we go beyond **grid-based patches** of images?

- Context Clusters view an image as a set of unorganized points and extract features via simplified clustering algorithm

- Assuming a cluster has $\boldsymbol{m}$ points, **aggregation** and **dispatching** are done **within the cluster**
- The **cosine similarity $\boldsymbol{s} \in \mathbb{R}^m$** between **$\boldsymbol{m}$ points** and **the cluster center** is used as **weights**:
  - **Feature aggregation** (g)
    - (note that v_i is **MLP projection** of each point p_i and α, β are **learnable scalars**)

$$g = \frac{1}{C}\left(v_c + \sum_{i=1}^{m} \operatorname{sig}\left(\alpha s_i + \beta\right) * v_i\right), \qquad \text{s.t.,} \quad C = 1 + \sum_{i=1}^{m} \operatorname{sig}\left(\alpha s_i + \beta\right).$$

  - **Feature dispatching**

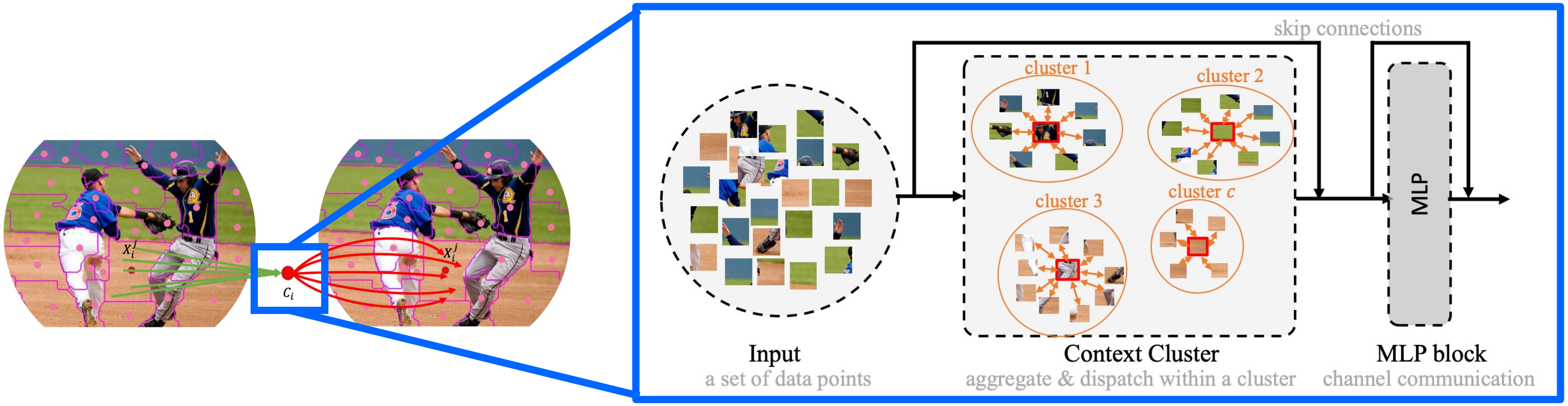
$$p_i' = p_i + \operatorname{FC}\left(\operatorname{sig}\left(\alpha s_i + \beta\right) * g\right).$$

- **Context Clusters (CoC) [Ma et al, 2023]**

Motivation: Can we go beyond **grid-based patches** of images?

- Context Clusters view an image as a set of unorganized points and extract features via simplified clustering algorithm

- Assuming a cluster has m points, **aggregation** and **dispatching** are done **within the cluster**
- Finally, additional **MLP block** is applied for **channel-wise mixing** in each point



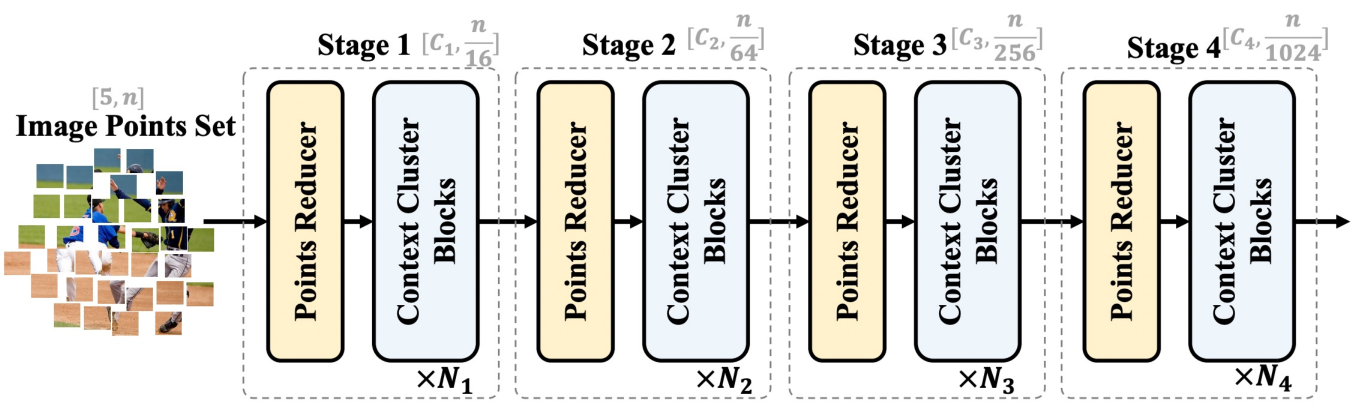
- **Context Clusters (CoC) [Ma et al, 2023]**

Motivation: Can we go beyond **grid-based patches** of images?

- Context Clusters view an image as a set of unorganized points and extract features via simplified clustering algorithm

- Assuming a cluster has m points, **aggregation** and **dispatching** are done **within the cluster**
- Finally, additional **MLP block** is applied for **channel-wise mixing** in each point

- To save the computation, some stages of **Points Reducer** is applied
  - Reducing is simply done by regular **convolution operations** (e.g., 4×4) over the spatial grid

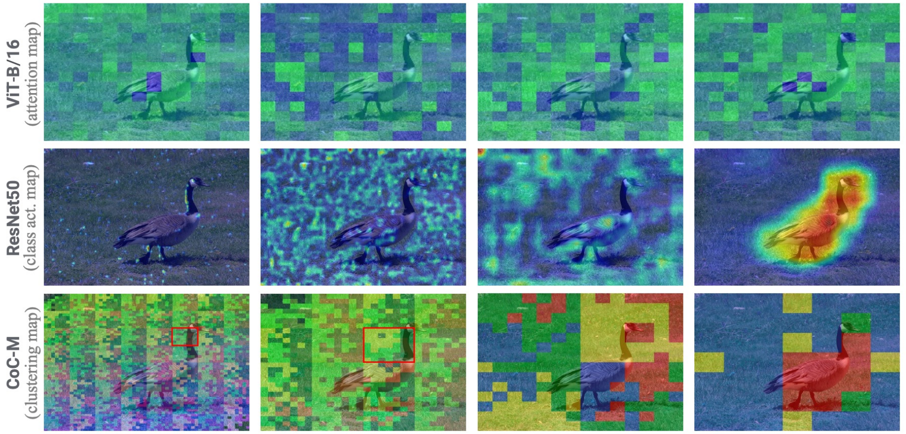


- **Context Clusters (CoC) [Ma et al, 2023]**

Motivation: Can we go beyond **grid-based patches** of images?

- Context Clusters view an image as a set of unorganized points and extract features via simplified clustering algorithm
- CoC can **outperform CNNs and Transformers**
  - More importantly, CoC shows clustering **with the semantics in image**

	Method	Param.	GFLOPs	Top-1	Throughputs (images/s)
MLP	♣ ResMLP-12 (Touvron et al., 2021a)	15.0	3.0	76.6	511.4
	♣ ResMLP-24 (Touvron et al., 2021a)	30.0	6.0	79.4	509.7
	♣ ResMLP-36 (Touvron et al., 2021a)	45.0	8.9	79.7	452.9
	♣ MLP-Mixer-B/16 (Tolstikhin et al., 2021)	59.0	12.7	76.4	400.8
	♣ MLP-Mixer-L/16 (Tolstikhin et al., 2021)	207.0	44.8	71.8	125.2
	♣ gMLP-Ti (Liu et al., 2021a)	6.0	1.4	72.3	511.6
	♣ gMLP-S (Liu et al., 2021a)	20.0	4.5	79.6	509.4
Attention	♦ ViT-B/16 (Dosovitskiy et al., 2020)	86.0	55.5	77.9	292.0
	♦ ViT-L/16 (Dosovitskiy et al., 2020)	307	190.7	76.5	92.8
	♦ PVT-Tiny (Wang et al., 2021)	13.2	1.9	75.1	-
	♦ PVT-Small (Wang et al., 2021)	24.5	3.8	79.8	-
	♦ T2T-ViT-7 (Yuan et al., 2021a)	4.3	1.1	71.7	-
	♦ DeiT-Tiny/16 (Touvron et al., 2021b)	5.7	1.3	72.2	523.8
	♦ DeiT-Small/16 (Touvron et al., 2021b)	22.1	4.6	79.8	521.3
Convolution	♠ ResNet18 (He et al., 2016)	12	1.8	69.8	584.9
	♠ ResNet50 (He et al., 2016)	26	4.1	79.8	524.8
	♠ ConvMixer-512/16 (Trockman et al., 2022)	5.4	-	73.8	-
	♠ ConvMixer-1024/12 (Trockman et al., 2022)	14.6	-	77.8	-
	♠ ConvMixer-768/32 (Trockman et al., 2022)	21.1	-	80.16	142.9
Cluster	♥ Context-Cluster-Ti (ours)	5.3	1.0	71.8	518.4
	♥ Context-Cluster-Ti‡ (ours)	5.3	1.0	71.7	510.8
	♥ Context-Cluster-Small (ours)	14.0	2.6	77.5	513.0
	♥ Context-Cluster-Medium (ours)	27.9	5.5	81.0	325.2

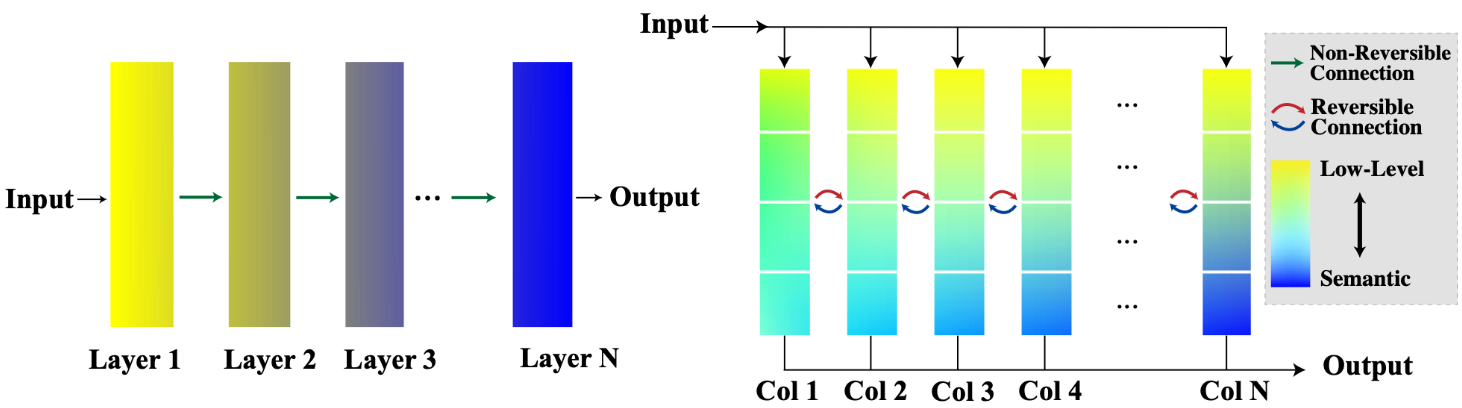


- **RevCol: Reversible Column Networks [Cai et al, 2023]**

Motivation: Can we go beyond **Information Bottleneck (IB)** principle?

- Deep networks (left) are built on the Information Bottleneck
  - **Layers close to the input** contain more **low-level** information
  - **Features close to the output** are rich in **semantics**

- However, **downstream tasks may suffer** if the learned features are **over-compressed** e.g., Transfer learning for object detection

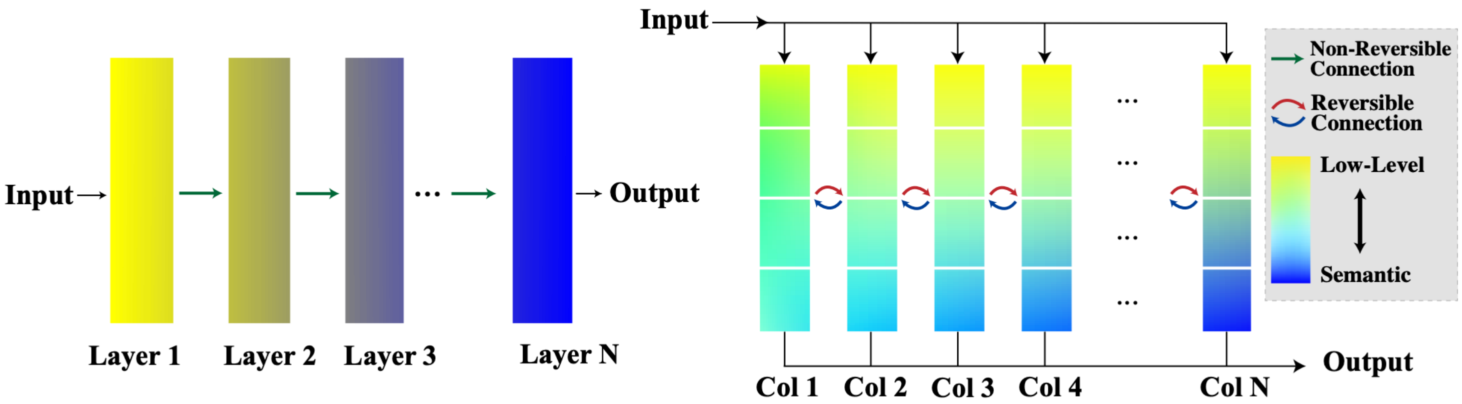


- **RevCol: Reversible Column Networks [Cai et al, 2023]**

Motivation: Can we go beyond **Information Bottleneck (IB)** principle?

- Deep networks (left) are built on the Information Bottleneck
    - **Layers close to the input** contain more **low-level** information
    - **Features close to the output** are rich in **semantics**

- Instead, RevCol suggests a design where information in the ***earlier layer*** could be (approximately) **restored with** information in **the later layers**
    - Then, how is the network designed?

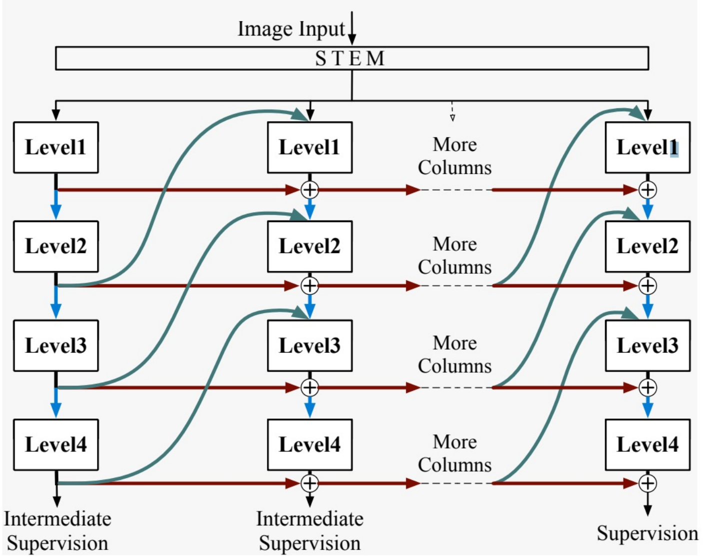


- **RevCol: Reversible Column Networks [Cai et al, 2023]**

Motivation: Can we go beyond **Information Bottleneck (IB)** principle?

- Inspired by ***invertible neural networks*** in Normalizing Flow, reversible operations are defined:
  - t is the depth index, $\boldsymbol{F}_t$ is the layer at t, γ is a simple channel-wise scaling
- Specifically, the output x_t is the **weighted sum** of x_{t-m} and non-linear transform of intermediate states $x_{t-1}, \dots, x_{t-m+1}$
  - Note that the **operation is invertible**
  - Any **deep networks** can implement $\boldsymbol{F}_t$ (e.g., ConvNext is employed)

$$Forward: x_t = \boldsymbol{F}_t(x_{t-1}, x_{t-2}, \dots, x_{t-m+1}) + \gamma x_{t-m}$$
$$Inverse: x_{t-m} = \gamma^{-1}[x_t - \boldsymbol{F}_t(x_{t-1}, x_{t-2}, \dots, x_{t-m+1})],$$



- **RevCol: Reversible Column Networks [Cai et al, 2023]**

Motivation: Can we go beyond **Information Bottleneck (IB)** principle?

- Inspired by ***invertible neural networks*** in Normalizing Flow, reversible operations are defined:
  - t is the depth index, $\boldsymbol{F}_t$ is the layer at t, γ is a simple channel-wise scaling
- Despite the restrictive design, ConvNext with RevCol is **comparable** to the vanilla model
  - More importantly, **transfer learning is improved** in object detection

Model	Image Size	Params (M)	FLOPs (G)	Top-1 Acc.
ImageNet-22K pre-trained models (ImageNet-1K fine-tuned)				
○ Swin-B (Liu et al.	224^2	88	15.4	85.2
○ Swin-B↑ (Liu et al.	384^2	88	47.0	86.4
○ ViT-B↑ (Dosovitskiy et al.)	384^2	86	55.4	84.0
• RepLKNet-31B (Ding et al.)	224^2	79	15.3	85.2
• RepLKNet-31B↑ (Ding et al.)	384^2	79	45.1	86.0
• ConvNeXt-B (Liu et al.)	224^2	89	15.4	85.8
• ConvNeXt-B↑ (Liu et al.)	384^2	89	45.1	86.8
• RevCol-B	224^2	138	16.6	85.6
• RevCol-B↑	384^2	138	48.9	86.7
○ Swin-L (Liu et al.)	224^2	197	34.5	86.3
○ Swin-L↑ (Liu et al.)	384^2	197	103.9	87.3
○ ViT-L↑ (Dosovitskiy et al.)	384^2	307	190.7	85.2
• RepLKNet-31L (Ding et al.)	384^2	172	96.0	86.6
• ConvNeXt-L (Liu et al.)	224^2	198	34.4	86.6
• ConvNeXt-L↑ (Liu et al.)	384^2	198	101.0	87.5
• RevCol-L	224^2	273	39.0	86.6
• RevCol-L↑	384^2	273	116.0	87.6

Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	Params	FLOPs
ImageNet-22K pre-trained								
○ Swin-B (Liu et al.)	53.0	71.8	57.5	45.8	69.4	49.7	145M	982G
• ConvNeXt-B (Liu et al.)	54.0	73.1	58.8	46.9	70.6	51.3	146M	964G
• RepLKNet-B (Ding et al.)	53.0	-	-	46.3	-	-	137M	965G
• RevCol-B	55.0	73.5	59.7	47.5	71.1	51.8	196M	988G
○ Swin-L (Liu et al.)	53.9	72.4	58.8	46.7	70.1	50.8	253M	1382G
• ConvNeXt-L (Liu et al.)	54.8	73.8	59.8	47.6	71.3	51.7	255M	1354G
• RepLKNet-L (Ding et al.)	53.9	-	-	46.5	-	-	229M	1321G
• RevCol-L	55.9	74.1	60.7	48.4	71.8	52.8	330M	1453G
Extra data pre-trained								
• RevCol-H (HTC++)	61.1	78.8	67.0	53.0	76.3	58.7	2.41G	4417G
• RevCol-H (Objects365+DINO)	63.8	81.8	70.2	-	-	-	2.18G	4012G

Summary

- The **larger** the network, the **more difficult** it is to design
    1. Optimization difficulty
    2. Generalization difficulty
    - **ResNet**: Optimization ⇒ Generalization
        - Many variants of ResNet have been emerged
        - Very recent trends towards **network design and scaling**

- Recently, various types of **patch-based architectures** are explored
    - **Vision transformers**, **MLP-mixing models**, etc.

- Many types of **architectures** are explored to capture good representation
    - **Automated network designs** and **flexible model architectures**
    - Many **observational study** supports the advantages of each architecture
    - **Spatial-temporal models** (e.g., 3D CNNs and video transformers)

- A new architectural paradigms are actively searched

    e.g., Graph-based architectures and Reversible networks

References

[Jónsson et al., 1998] Jónsson, H., Mills, G., & Jacobsen, K. W. (1998). Nudged elastic band method for finding minimum energy paths of transitions. In Classical and quantum dynamics in condensed phase simulations (pp. 385-404).
link : https://www.worldscientific.com/doi/abs/10.1142/9789812839664_0016

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
link : https://ieeexplore.ieee.org/abstract/document/726791/

[Bergstra et al., 2012] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.
link : http://www.jmlr.org/papers/v13/bergstra12a.html

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
link : http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[Farabet et al., 2013] Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2013). Learning hierarchical features for scene labeling. IEEE transactions on pattern analysis and machine intelligence, 35(8), 1915-1929.
link : https://ieeexplore.ieee.org/abstract/document/6338939/

[Eigen et al., 2014] Eigen, D., Rolfe, J., Fergus, R., & LeCun, Y. (2013). Understanding Deep Architectures using a Recursive Convolutional Network. ArXiv Preprint ArXiv:1312.1847, 1–9.
link : http://arxiv.org/abs/1312.1847

[Simonyan et al., 2014] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
link: https://arxiv.org/abs/1409.1556

References

[Zeiler et al., 2014] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.
link : https://link.springer.com/chapter/10.1007/978-3-319-10590-1_53

[Ioffe et al., 2015] Ioffe, S. & Szegedy, C.. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, in PMLR 37:448-456
link : http://proceedings.mlr.press/v37/ioffe15.html

[Ren et al., 2015] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*(pp. 91-99).
link : http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks

[Russakovsky et al., 2015] Russakovsky, O. et al. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.
link : https://link.springer.com/article/10.1007/s11263-015-0816-y

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
link : https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html

[Han et al., 2016] Han, D., Kim, J., & Kim, J. (2017, July). Deep pyramidal residual networks. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 6307-6315). IEEE.
link : https://ieeexplore.ieee.org/document/8100151/

[Yu et al., 2016] Yu, F., & Koltun, V.. (2016) Multi-Scale Context Aggregation by Dilated Convolutions. In International Conference on Learning Representations.
link : https://arxiv.org/abs/1511.07122

References

[He et al., 2016a] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
link : https://ieeexplore.ieee.org/document/7780459/

[He et al., 2016b] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In European conference on computer vision (pp. 630-645). Springer, Cham.
link : https://link.springer.com/chapter/10.1007/978-3-319-46493-0_38

[Huang et al., 2016] Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. In European Conference on Computer Vision (pp. 646-661).
link : https://link.springer.com/chapter/10.1007/978-3-319-46493-0_39

[Kolsbjerg et al., 2016] Kolsbjerg, E. L., Groves, M. N., & Hammer, B. (2016). An automated nudged elastic band method. The Journal of chemical physics, 145(9), 094107.
link : https://aip.scitation.org/doi/abs/10.1063/1.4961868

[Targ et al., 2016] Targ, S., Almeida, D., & Lyman, K. (2016). Resnet in Resnet: generalizing residual architectures. arXiv preprint arXiv:1603.08029.
link : https://arxiv.org/abs/1603.08029

[Veit et al., 2016] Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In Advances in Neural Information Processing Systems (pp. 550-558).
link : http://papers.nips.cc/paper/6556-residual-networks-behave-like-ensembles-of-relatively-shallow-networks

[Xie et al., 2016] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017, July). Aggregated residual transformations for deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 5987-5995). IEEE.
link : http://openaccess.thecvf.com/content_cvpr_2017/papers/Xie_Aggregated_Residual_Transformations_CVPR_2017_paper.pdf

References

[Zagoruyko et al., 2016] Zagoruyko, S. and Komodakis, N. (2016). Wide Residual Networks. In Proceedings of the British Machine Vision Conference (pp. 87.1-87.12).
link : http://www.bmva.org/bmvc/2016/papers/paper087/index.html

[Zoph et al., 2016] Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.
link : https://arxiv.org/abs/1611.01578

[Chen et al., 2017] Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., & Feng, J. (2017). Dual path networks. In Advances in Neural Information Processing Systems (pp. 4467-4475).
link : https://papers.nips.cc/paper/7033-dual-path-networks

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017, July). Densely Connected Convolutional Networks. In CVPR (Vol. 1, No. 2, p. 3).
link : http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf

[Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In AAAI (Vol. 4, p. 12).
link : https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/download/14806/14311

[Dai et al., 2017] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 764-773).
link : https://arxiv.org/abs/1703.06211v3

[Chen et al., 2017] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.
link : https://arxiv.org/abs/1706.05587

[Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
link: https://arxiv.org/abs/1704.04861

References

[Draxler et al., 2018] Draxler, F., Veschgini, K., Salmhofer, M. & Hamprecht, F. (2018). Essentially No Barriers in Neural Network Energy Landscape. Proceedings of the 35th International Conference on Machine Learning, in PMLR 80:1309-1318.
link : http://proceedings.mlr.press/v80/draxler18a.html

[Luo et al., 2018] Luo, R., Tian, F., Qin, T., Chen, E. & Liu, T. (2018) Neural Architecture Optimization. arXiv preprint arXiv:1808.07233.
link : https://arxiv.org/abs/1808.07233

[Li et al., 2018] Li, H., Xu, Z., Taylor, G., & Goldstein, T. (2017). Visualizing the loss landscape of neural nets. arXiv preprint arXiv:1712.09913.
link : https://arxiv.org/abs/1712.09913

[Pham et al., 2018] Pham, H., Guan, M., Zoph, B., Le, Q. & Dean, J.. (2018). Efficient Neural Architecture Search via Parameters Sharing. Proceedings of the 35th International Conference on Machine Learning, in PMLR 80:4095-4104
link : http://proceedings.mlr.press/v80/pham18a.html

[Real et al., 2018] Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2018). Regularized evolution for image classifier architecture search. arXiv preprint arXiv:1802.01548.
link : https://arxiv.org/abs/1802.01548

[Zoph et al., 2018] Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). Learning transferable architectures for scalable image recognition. arXiv preprint arXiv:1707.07012, 2(6).
link : http://openaccess.thecvf.com/content_cvpr_2018/papers/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.pdf

[Brock et al., 2018] Brock, A., Lim, T., Ritchie, J. M., & Weston, N. (2018). SMASH: one-shot model architecture search through hypernetworks. In International Conference on Learning Representations.
link : https://openreview.net/forum?id=rydeCEhs-

References

[Hu et al., 2018] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).
link : https://arxiv.org/abs/1709.01507

[Woo et al., 2018] Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 3-19).
link : https://arxiv.org/abs/1807.06521v2

[Tan et al., 2019] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828).
link : https://arxiv.org/abs/1807.11626

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
link : https://arxiv.org/abs/2010.11929

[Liu et al., 2021] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 10012-10022).
link :
https://openaccess.thecvf.com/content/ICCV2021/html/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper.html

[Tolstikhin et al., 2021] Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., ... & Dosovitskiy, A. (2021). Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, *34*.
link : https://proceedings.neurips.cc/paper/2021/hash/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Abstract.html

References

[Heo et al., 2021] Heo, B., Yun, S., Han, D., Chun, S., Choe, J., & Oh, S. J. (2021). Rethinking spatial dimensions of vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 11936-11945).
link : https://arxiv.org/abs/2103.16302

[Li et al., 2021] Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z. H., ... & Yan, S. (2021). Tokens-to-token vit: Training vision transformers from scratch on imagenet. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 558-567).
link : https://arxiv.org/abs/2101.11986

[Raghu et al., 2021] Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C. and Dosovitskiy, A., 2021. Do vision transformers see like convolutional neural networks?. Advances in Neural Information Processing Systems, 34, pp.12116-12128.
link: https://arxiv.org/abs/2108.08810

[Park et al., 2022] Park, Namuk, and Songkuk Kim. "How Do Vision Transformers Work?." In *International Conference on Learning Representations*.
link: https://arxiv.org/abs/2202.06709

[Raghu et al., 2021] Park, Namuk, and Songkuk Kim. "How Do Vision Transformers Work?." In *International Conferenc e on Learning Representations*.
link: https://arxiv.org/abs/2202.06709

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1725-1732).
link : https://ieeexplore.ieee.org/document/6909619

References

[Wang et al., 2016] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Gool, L. V. (2016, October). Temporal segment networks: Towards good practices for deep action recognition. In European conference on computer vision (pp. 20-36).
link : https://arxiv.org/abs/1608.00859

[Carreira and Zisserman, 2017] Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6299-6308).
link : https://arxiv.org/abs/1705.07750

[Tran et al., 2017] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 6450-6459).
link : https://arxiv.org/abs/1711.11248

[Qiu et al., 2018] Qiu, Z., Yao, T., & Mei, T. (2017). Learning spatio-temporal representation with pseudo-3d residual networks. In proceedings of the IEEE International Conference on Computer Vision (pp. 5533-5541).
link : https://arxiv.org/abs/1711.10305

[Tran et al., 2019] Tran, D., Wang, H., Torresani, L., & Feiszli, M. (2019). Video classification with channel-separated convolutional networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5552-5561).
link : https://arxiv.org/abs/1904.02811

References

[Arnab et al., 2021] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021). Vivit: A video vision transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6836-6846).
link : https://arxiv.org/abs/2103.15691

[Bertasius et al., 2021] Bertasius, G., Wang, H., & Torresani, L. (2021). Is space-time attention all you need for video understanding. arXiv preprint arXiv:2102.05095, 2(3), 4.
link : https://arxiv.org/abs/2102.05095

[Liu et al., 2021] Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., & Hu, H. (2021). Video swin transformer. arXiv preprint arXiv:2106.13230.
link : https://arxiv.org/abs/2106.13230

[Bulat et al., 2021] Bulat, A., Perez Rua, J. M., Sudhakaran, S., Martinez, B., & Tzimiropoulos, G. (2021). Space-time mixing attention for video transformer. Advances in Neural Information Processing Systems, 34.
link : https://proceedings.neurips.cc/paper/2021/hash/a34bacf839b923770b2c360eefa26748-Abstract.html

[Li et al., 2022] Li, K., Wang, Y., Gao, P., Song, G., Liu, Y., Li, H., & Qiao, Y. (2022). Uniformer: Unified Transformer for Efficient Spatiotemporal Representation Learning. arXiv preprint arXiv:2201.04676.
link : https://arxiv.org/abs/2201.04676

[Touveron et al., 2022] Touvron, Hugo, Matthieu Cord, and Hervé Jégou. "Deit iii: Revenge of the vit." In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV, pp. 516-533. Cham: Springer Nature Switzerland, 2022.
link: https://arxiv.org/abs/2204.07118

References

[Jiahui et al., 2020] Yu, Jiahui, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. "Bignas: Scaling up neural architecture search with big single-stage models." In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16, pp. 702-717. Springer International Publishing, 2020.
link: https://arxiv.org/abs/2003.11142

[Gong et al., 2021] Gong, Chengyue, Dilin Wang, Meng Li, Xinlei Chen, Zhicheng Yan, Yuandong Tian, and Vikas Chandra. "Nasvit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training." In International Conference on Learning Representations. 2021.
link: https://openreview.net/forum?id=Qaw16njk6L

[Zhu et al., 2021] Zhu, Xizhou, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. "Deformable DETR: Deformable Transformers for End-to-End Object Detection." In International Conference on Learning Representations.
link: https://arxiv.org/abs/2010.04159

[Liang et al., 2022] Liang, Youwei, G. E. Chongjian, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. "EViT: Expediting Vision Transformers via Token Reorganizations." In International Conference on Learning Representations.
link: https://arxiv.org/abs/2202.07800

[Fayyaz et al., 2022] Fayyaz, Mohsen, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. "Adaptive token sampling for efficient vision transformers." In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI, pp. 396-414. Cham: Springer Nature Switzerland, 2022.
link: https://arxiv.org/abs/2111.15667

References

[Kai et al 2022] Han, Kai, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. "Vision GNN: An Image is Worth Graph of Nodes." In *Advances in Neural Information Processing Systems*.
link: https://arxiv.org/abs/2206.00272

[Yu et al., 2022] Yu, Weihao, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. "Metaformer is actually what you need for vision." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10819-10829. 2022.
link: https://arxiv.org/abs/2111.11418

[Liu et al., 2022] Liu, Zhuang, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. "A convnet for the 2020s." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976-11986. 2022.
link: https://arxiv.org/abs/2201.03545

[Ma et al., 2023] Xu Ma and Yuqian Zhou and Huan Wang and Can Qin and Bin Sun and Chang Liu and Yun Fu. "Image as Set of Points" in International Conference on Learning Representations 2023
link: https://openreview.net/forum?id=awnvqZja69

[Cai et al., 2023] Cai, Yuxuan, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. "Reversible Column Networks" in International Conference on Learning Representations 2023
link: https://arxiv.org/abs/2212.11696