Deep Learning for Computer Vision (Discriminative way)

Andrii Liubonko

2025

Intro

Era of
Human-Crafter
Features

Era of
Deep Learning

Era of
LLMs
(Foundational
Models)

2012
AlexNet

2022
ChatGPT

traditional ML

Old days: one model for one purpose

prompt-based ML

Now: one model for multiple purposes

"Foundation model"

prompt-based ML
Prompt model
Deploy in production
does it really work ?
YES
can we make it cheaper ?
NO
YES
on device
YES
"traditional" ML/DL
Get labeled data
Develop model
Deploy in production

APIs multimodal LLMs

Local multimodal LLMs

GEMMA, LLAMA, ...

CV foundational models

DINO, SAM, ...

CV DL fundamentals

"classical" CV

math

CV DL fundamentals

Convolutional Neural Networks (CNN)

Transformer-based

Big Picture

Net

Big Picture

Net

↑

Blocks

Net

↑

Blocks

↑

Units (layers)

- **Layers**
  - **Layers [Convolution] [Receptive field]**
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  ResNet, MobileNet, EfficientNet, ConvNeXt

Layers [Convolution][motivation]

- Images are high-dimensional

  224×224 RGB => #parameters = (224×224×3)^2 = 22 billion

- The interpretation of an image is stable under geometric transformations

  => invariance and equivariance

- Nearby image pixels are statistically related

$$\left\{ f(\boldsymbol{x};\boldsymbol{\theta}) = \mathbf{W}_L \boldsymbol{\sigma}_L(\mathbf{W}_{L-1} \cdots \boldsymbol{\sigma}_2(\mathbf{W}_2 \boldsymbol{\sigma}_1(\mathbf{W}_1 \boldsymbol{x}))) \;\middle|\; \boldsymbol{\theta} = \{\mathbf{W}_1, \ldots, \mathbf{W}_L\} \right\}$$

Layers [Convolution][1d]

single filter -
single output channel:

* image from Understanding Deep Learning, book by Simon J.D. Prince, 2023

Layers [Convolution][1d]

two filters -
two output channel:

Layers [Convolution][1d]

- kernel_size
- stride
- padding
- number of filters

Output = (Input - Size + 2 * Padding) / Stride + 1

Layers [Convolution][2d]

Layers [Convolution][2d on image]

Layers [Convolution][2d on image]

spatial extent is controlled by **stride**, **kernel_size**, **padding**

Layers [Convolution]

$$H_{out} = (H_{in} - K + 2 * Pad) / Stride + 1$$

$$W_{out} = (W_{in} - K + 2 * Pad) / Stride + 1$$

$$C_{out} = \#filters$$

Layers [Convolution]

- kernel size K ?
- padding size Pad ?
- stride Stride ?

Layers [Convolution]

- kernel size K = 4x4
- padding size Pad = 2
- stride Stride = 1

Layers [Convolution]

- kernel size, K = 3x3
- padding size, Pad = 1
- stride, Stride = 2

output size ?

Layers [Convolution]

- kernel size, K = 3x3
- padding size, Pad = 1
- stride, Stride = 2

$$H_{out} = (H_{in} - K + 2 * Pad) / Stride + 1$$

$$W_{out} = (W_{in} - K + 2 * Pad) / Stride + 1$$

$$C_{out} = \#filters$$

Layers [Convolution][Receptive field][2d]

The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by)

Layers [Convolution][Receptive field]

The **receptive field** is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by)

Layers [Convolution][Receptive field][1d]

- Layers
  - Layers [Convolution, Receptive field]
  - **Layers [Dilated Convolution, Deformable Convolution]**
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*
  - EfficientNet

- Architectures
  AlexNet, VGG, Inception,
  ResNet*, MobileNet*, EfficientNet

Layers [Dilated Convolution]

Also known as
“atrous convolutions”.

Dilated convolutions “inflate” the kernel by inserting spaces between the kernel elements. The dilation “rate” is controlled by an additional hyperparameter **d**.

Implementations may vary, but there are usually **d - 1** spaces inserted between kernel elements such that **d = 1** corresponds to a regular convolution.

Size = 3
Stride = 2
Dilation =1

Size = 3
Stride = 1
Dilation = 2

Layers [Dilated Convolution]

- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2

Layers [Dilated Convolution]

- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2

For 'normal' convolution we have:

$H_{out} = (H_{in} - K + 2 * Pad) / Stride + 1$

$W_{out} = (W_{in} - K + 2 * Pad) / Stride + 1$

C_{out} = #filters

How this will change ?

Layers [Dilated Convolution]

- input size I = 7x7
- kernel size F = 3x3,
- padding size P = 0,
- stride S = 1
- dilation rate **d** = 2

For 'normal' convolution we have:

$H_{out} = (H_{in} - K + 2 * Pad) / Stride + 1$

$W_{out} = (W_{in} - K + 2 * Pad) / Stride + 1$

C_{out} = #filters

How this will change ? $K^* = K + (K - 1) * (d - 1)$

Layers [Dilated Convolution]

- Detection of fine-details by processing inputs in higher resolutions.
- Broader view of the input to capture more contextual information.
- Faster run-time with less parameters

Layers [Deformable Convolution]

$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$$

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

1703.06211 , 1811.11168 , 2211.05778 , 2401.06197

Layers [Deformable Convolution]

$$\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$$

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

1703.06211 , 1811.11168 , 2211.05778 , 2401.06197

Layers [Deformable Convolution]

1703.06211 , 1811.11168 , 2211.05778 , 2401.06197

Layers [Deformable Convolution]

1703.06211 , 1811.11168 , 2211.05778 , 2401.06197

Layers [Deformable Convolution]

1703.06211 , 1811.11168 , 2211.05778 , 2401.06197

Layers [Deformable Convolution]

1703.06211 , 1811.11168 , **2211.05778** , 2401.06197

Content of today lecture

- **Layers**
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - **Layers [Group Convolutions and its variants]**
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Layers [Convolution]

Layers [Convolution]

The number of parameters: of standard convolution is $C_{in}KK \cdot C_{out}$ when 'bias' is False and $C_{in}(KK+1) \cdot C_{out}$ otherwise.

The computational cost: of standard convolution is $C_{in}KK \cdot H_{out}W_{out}C_{out}$

1. the spatial size of the output feature map $(H_{out}W_{out})$
2. the size of conv. kernel K^2
3. the number of input and output channels $C_{in} \cdot C_{out}$

Layers [Convolution]

Layers [Convolution]

Normal Convolution:

conv 3x3

Layers [1x1 Convolution, Pointwise Convolution]

conv 1x1

A special name:
point-wise convolutions

Layers [1x1 Convolution, Pointwise Convolution]

The number of parameters: $C_{in} \cdot C_{out}$ (no bias)

The computational cost: $C_{in} \cdot H_{out} W_{out} C_{out}$ (1/9 reduction in comparison to 3x3 kernel)

Layers [Group Convolution]

gconv 3x3 (g=2)

Layers [Group Convolution]

The number of parameters: $C_{in}/G\ KK \cdot C_{out}$ (no bias), the tensor (of kernel parameters) has the following sizes: $[C_{out}, C_{in}/G, K, K]$

The computational cost: $C_{in}/G\ KK \cdot H_{out} W_{out} C_{out}$

Layers [Group Convolution]

gconv 3x3 (g=3)

Layers [Group Convolution, depth-wise convolutions]

gconv 3x3 (g=6)

special name -
depth-wise convolutions
or
channel-wise convolutions

Layers [Group Convolution]

The number of parameters: $KK \cdot C_{out}$ (no bias), the tensor has the following sizes: $[C_{out}, 1, K, K]$

The computational cost: $KK \cdot H_{out} W_{out} C_{out}$

Content of today lecture

- **Layers**
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - **Layers [Downsampling, Upsampling]**
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Layers [Downsampling]

* image from Understanding Deep Learning, book by Simon J.D. Prince, 2023

Layers [Upsampling]

Layers [Upsampling]

- **Nearest Neighbor**

1	2
3	4

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Input: [2 x 2]

Output: [4 x 4]

- **Bilinear**

1	2
3	4

1.00	1.25	1.75	2.00
1.50	1.75	2.00	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: [2 x 2]

Output: [4 x 4]

Layers [Upsampling: Transpose Convolution]

* image from Understanding Deep Learning, book by Simon J.D. Prince, 2023

Content of today lecture

- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Upsampling, Learnable Upsampling]
  - **Layers [Norm, Dropout, Stochastic depth]**
- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*
  - EfficientNet
- Architectures
  VGG, Inception, ResNet*, MobileNet*, EfficientNet
- Neural Architecture Search (NAS)
- Summary

Layers [Batch Normalization]

Batch normalization:

$$\mu_j = \frac{1}{N_b}\sum_{i=1}^{N_b} x_{ij}$$

$$\sigma_j^2 = \frac{1}{N_b}\sum_{i=1}^{N_b} (x_{ij} - \mu_j)^2$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

Layer normalization:

$$\mu_i = \frac{1}{N_f}\sum_{j=1}^{N_f} x_{ij}$$

$$\sigma_i^2 = \frac{1}{N_f}\sum_{j=1}^{N_b} (x_{ij} - \mu_i)^2$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

Layers [Batch Normalization]

BatchNorm	LayerNorm
Normalizes across the batch for each channel.	Normalizes across the features for each sample.
Statistics (μ,σ) computed from all samples in batch.	Statistics computed from each individual sample.
Depends on batch size (unstable for small batch sizes).	Independent of batch size.
Often used in CNNs.	Often used in Transformers, NLP models, ConvNeXt.

Layers [Dropout]

- In-network ensembling
- Reduce overfitting

Layers [Drop Path / Stochastic Depth]

Regularization technique, especially used in ResNet-like blocks and ConvNeXt.

Instead of dropping individual neurons like Dropout, it randomly drops entire residual branches during training.

- In a residual block:

  $y = x + F(x)$

- With stochastic depth:

  $y = x + \boldsymbol{mask} * F(x)$

  where ***mask*** is 0 (drop block) with probability p and 1 with probability (1 − p).

This makes training deeper networks easier by effectively shortening them stochastically for each sample, reducing overfitting and helping gradient flow.

Convolutions

Computer Vision · **Image Feature Extractors** · 57 methods

Edit

Convolutions are a type of operation that can be used to learn representations from images. They involve a learnable kernel sliding over the image and performing element-wise multiplication with the input. The specification allows for parameter sharing and translation invariance. Below you can find a continuously updating list of convolutions.

Methods

Add a Method

Method	Year	Papers
Convolution	1980	13119
1x1 Convolution Network In Network	2013	4047
Depthwise Convolution	2016	795
Pointwise Convolution	2016	785
Depthwise Separable Convolution Xception: Deep Learning With Depthwise Separable Convolutions	2017	696
Grouped Convolution ImageNet Classification with Deep Convolutional Neural Networks	2012	514
Dilated Convolution Multi-Scale Context Aggregation by Dilated Convolutions	2015	459
3D Convolution	2015	156
Deformable Convolution Deformable Convolutional Networks	2017	97
SAC DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution	2020	84
Invertible 1x1 Convolution Glow: Generative Flow with Invertible 1x1 Convolutions	2018	50
1D CNN Convolutional Neural Network and Rule-Based Algorithms for Classifying 12-lead ECGs	2020	43
Groupwise Point Convolution ESPNetv2: A Light-weight, Power Efficient, and General Purpose Convolutional Neural Network	2018	42
Transposed convolution Fully Convolutional Networks for Semantic Segmentation	2016	25

- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- **Blocks**
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  [historical]: AlexNet, VGG, Inception,
  [modern]: ResNet, MobileNet, EfficientNet

Blocks [LeNet, AlexNet]

NIPS Paper
https://proceedings.neurips.cc › paper › 4824-im... PDF

ImageNet Classification with Deep Convolutional Neural ...

by A Krizhevsky · Cited by 147898 — We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 ...

9 pages

Blocks [VGG]

1409.1556

arXiv
https://arxiv.org › cs

Very Deep Convolutional Networks for Large-Scale Image ...

by K Simonyan · 2014 · Cited by 150488 — In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting.

Blocks [GoogleNet / Inception]

- to find out optimal local **sparse structure** and to repeat it spatially
- to split operations for cross-channel correlations and at spatial correlations into a series of independently operations.
- split-transform-merge strategy

Bottleneck

1409.4842

Blocks [GoogleNet / Inception]

1409.4842

Blocks [ResNet]

y = x + F(x)

In the basic design, F(x) contains two 3×3 convolution layers along with a batch normalization and/or a rectified linear unit activation function.

1512.03385

arXiv
https://arxiv.org › cs

[1512.03385] Deep Residual Learning for Image Recognition

by K He · 2015 · Cited by 286922 — We present a **residual learning framework** to ease the training of networks that are substantially deeper than those used previously. We ...

Blocks [ResNet, bottleneck]

Blocks [ResNet, bottleneck]

For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to Inception)

1512.03385

Blocks [ResNet, bottleneck]

1512.03385

Blocks [ResNeXt]

1611.05431

- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - **MobileNet***

- Architectures
  AlexNet, VGG, Inception,
  ResNet, MobileNet, EfficientNet

Blocks [depth-wise separable]

Blocks [depth-wise separable]

Blocks [depth-wise separable]

$$\frac{C_{in} \cdot H_{out} W_{out} C_{out} + KK \cdot H_{out} W_{out} C_{out}}{C_{in} KK \cdot H_{out} W_{out} C_{out}} = \frac{1}{K^2} + \frac{1}{C_{in}}$$

Blocks [mobile inverted bottleneck convolution (MBConv)]

- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- **Architectures**
  AlexNet, VGG, Inception,
  ResNet, MobileNet, EfficientNet

Net [AlexNet]

NIPS Paper
https://proceedings.neurips.cc › paper › 4824-im... PDF

ImageNet Classification with Deep Convolutional Neural ...

by A Krizhevsky · Cited by 147898 — We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 ...

9 pages

Net [AlexNet]

224×224×3
55×55×96
27×27×256
13×13×384
13×13
×256
4096
4096
1000
1000
Conv11×11
Stride 4
MaxPool
Conv 5 × 5
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

Net [VGG16]

1409.1556

arXiv
https://arxiv.org › cs

Very Deep Convolutional Networks for Large-Scale Image ...

by K Simonyan · 2014 · Cited by 150488 — In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting.

Net [VGG16]

224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
4096
4096
1000
1000
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool
Conv 3 × 3
MaxPool + FC
FC
FC
Softmax

Net [GoogLeNet]

1409.4842

Net [GoogLeNet]

Net [ResNet]

1512.03385

Net [ResNet]

BN
ReLU
Conv 1×1
BN
ReLU
Conv 3×3
BN
ReLU
Conv 1×1
224 × 224 × 3
112 × 112 × 64
ResBlock
56 × 56 × 256(64)
ResBlock
28 × 28 × 512(128)
ResBlock
14 × 14 × 1024(256)
ResBlock
7 × 7 × 2048(512)
1000
1000
24 blocks
total
36 blocks
total
Conv 7×7
Stride 2
MaxPool
Subsample
Subsample
Subsample
AvgPool + FC
Softmax

Net [ResNet]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Net [DenseNet]

1608.06993

Net [DenseNet]

1608.06993

Net [MobileNet V2]

MBConv block

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

V1: 1704.04861 V2: **1801.04381** V3: 1905.02244, V4: 2404.10518 (pytorch on HF)

Net [EfficientNet]

1905.11946, 2104.00298

Net [EfficientNet]

(a) baseline

(e) compound scaling

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

1905.11946, 2104.00298

Net [EfficientNet]

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

Methods

Add a Method

Method	Year	Papers
ResNet Deep Residual Learning for Image Recognition	2015	2005
VGG Very Deep Convolutional Networks for Large-Scale Image Recognition	2014	468
DenseNet Densely Connected Convolutional Networks	2016	382
VGG-16 Very Deep Convolutional Networks for Large-Scale Image Recognition	2014	351
MobileNetV2 MobileNetV2: Inverted Residuals and Linear Bottlenecks	2018	311
AlexNet ImageNet Classification with Deep Convolutional Neural Networks	2012	282
EfficientNet EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks	2019	251
Darknet-53 YOLOv3: An Incremental Improvement	2018	240
Xception Xception: Deep Learning With Depthwise Separable Convolutions	2017	151
ResNeXt Aggregated Residual Transformations for Deep Neural Networks	2016	125
GoogLeNet Going Deeper with Convolutions	2014	121

Net [ConvNeXt]

[2201.03545] A ConvNet for the 2020s

	output size	• ResNet-50	• ConvNeXt-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4
res2	56×56	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} d7\times7, 96 \\ 1\times1, 384 \\ 1\times1, 96 \end{bmatrix} \times 3$
res3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} d7\times7, 192 \\ 1\times1, 768 \\ 1\times1, 192 \end{bmatrix} \times 3$
res4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} d7\times7, 384 \\ 1\times1, 1536 \\ 1\times1, 384 \end{bmatrix} \times 9$
res5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} d7\times7, 768 \\ 1\times1, 3072 \\ 1\times1, 768 \end{bmatrix} \times 3$
FLOPs		4.1×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6

[2201.03545] A ConvNet for the 2020s

SOTA

50
0
Top 1 Error (%)
2012
2014
2016
2018
2020
2022
2024
Year
AlexNet
VGG
ResNet-200
DenseNet
ImageGPT
ViT
SWIN
DaViT

- Layers
  - Layers [Convolution] [Receptive field]
  - Layers [Dilated Convolution, Deformable Convolution]
  - Layers [Group Convolutions and its variants]
  - Layers [Upsampling, Learnable Upsampling]
  - Layers [Normalization, Batch Norm, Dropout]

- Blocks
  - VGG, Inception
  - ResNet*
  - MobileNet*

- Architectures
  AlexNet, VGG, Inception,
  ResNet, MobileNet, EfficientNet

CV DL fundamentals

Convolutional Neural Networks (CNN)

Transformer-based

Net

↑

Blocks

↑

Units (layers)

Tweaking Neural Net
Parameters
MEMECENTER.COM