Introduction to Computer Vision with Convolutional Neural Networks

Mohammad Haghighat
Senior Manager, CoreAI
eBay

Outline

- High level introduction to AI
  - Conventional vs. deep learning
- Neural networks and deep learning
  - Fully connected networks
  - Elements of a neural network
  - Neural network training
- Convolutional neural networks (CNNs)
  - Building blocks of CNNs
  - Applications of CNNs
  - Popular CNN architectures
  - Mobile CNN architectures

High-level introduction to AI

Machine Learning (ML) Model

person

ML Model

ML Model

person dancing on the beach

“Nothing to love about this presentation.”

ML Model

negative feedback

begining

ML Model

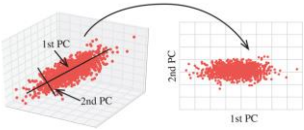
beginning

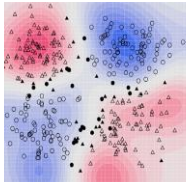
ML Model

“Let‘s go for lunch”

Classical learning vs deep learning

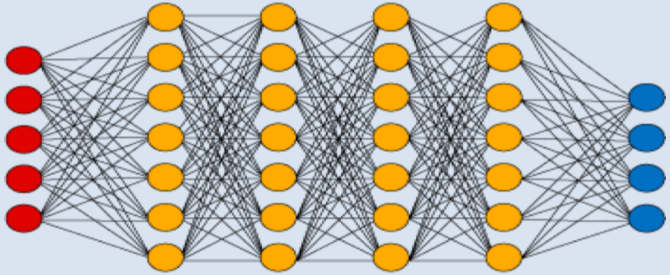
Input Data (e.g., image) → Feature Extraction (e.g., edges) → Dimensionality Reduction (e.g., PCA*) → Classifier (e.g., SVM*) → Output





Dog

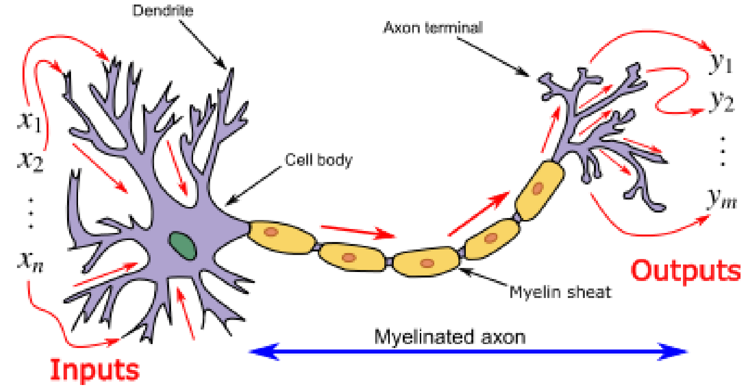
Input Data (e.g., image) →

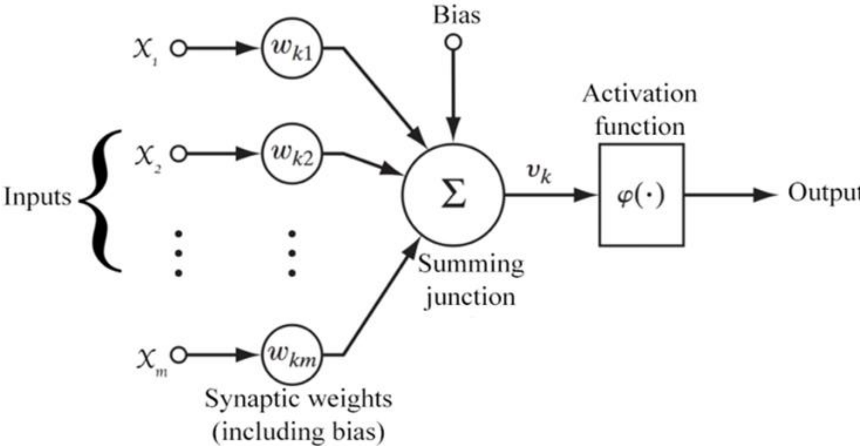


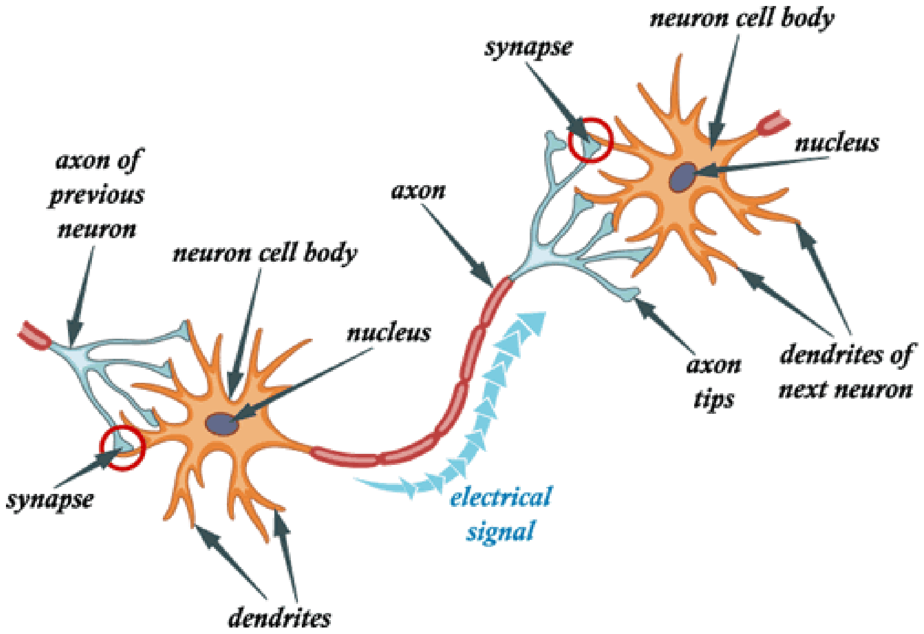
→ Output

*PCA: Principal Component Analysis
*SVM: Support Vector Machines

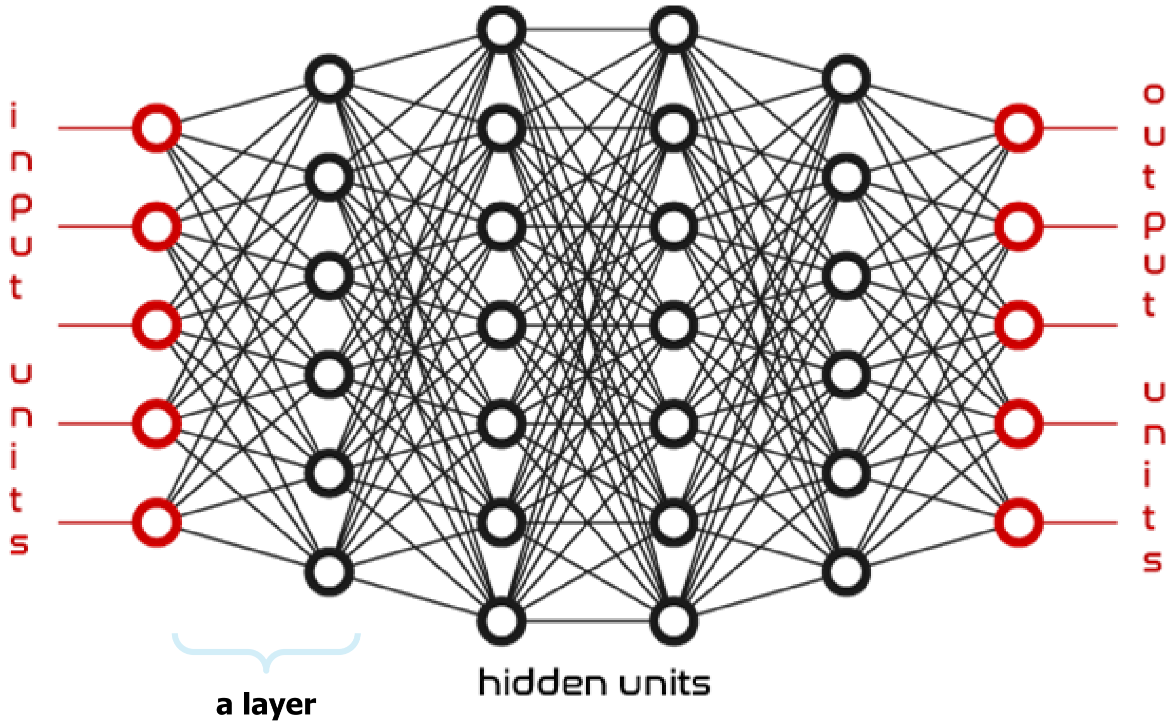
What are neurons?







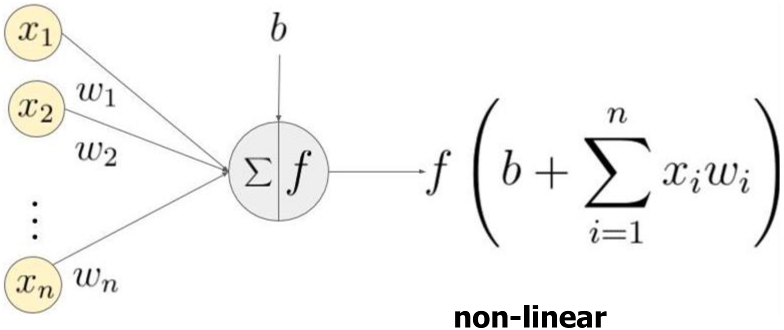
… and what are neural networks?



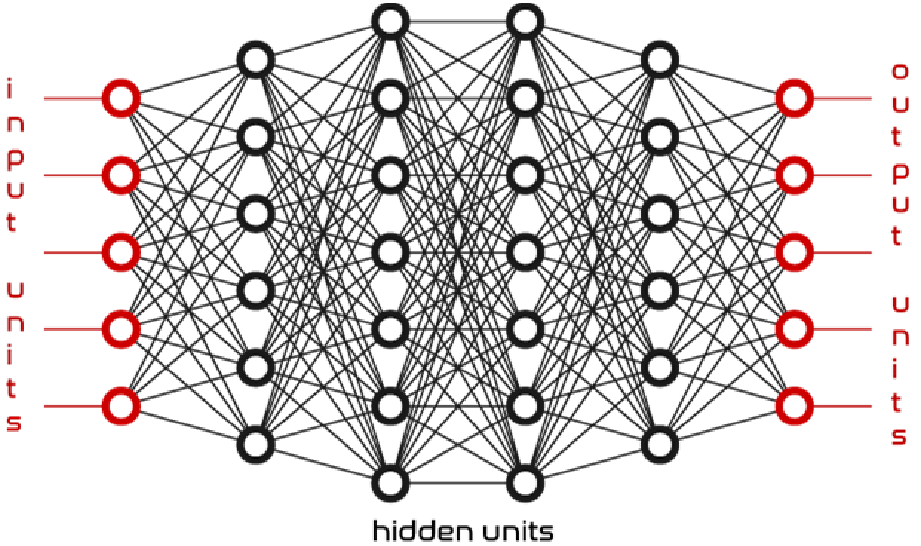
Neural networks as a vehicle for deep learning

Universal Approximation Theorem

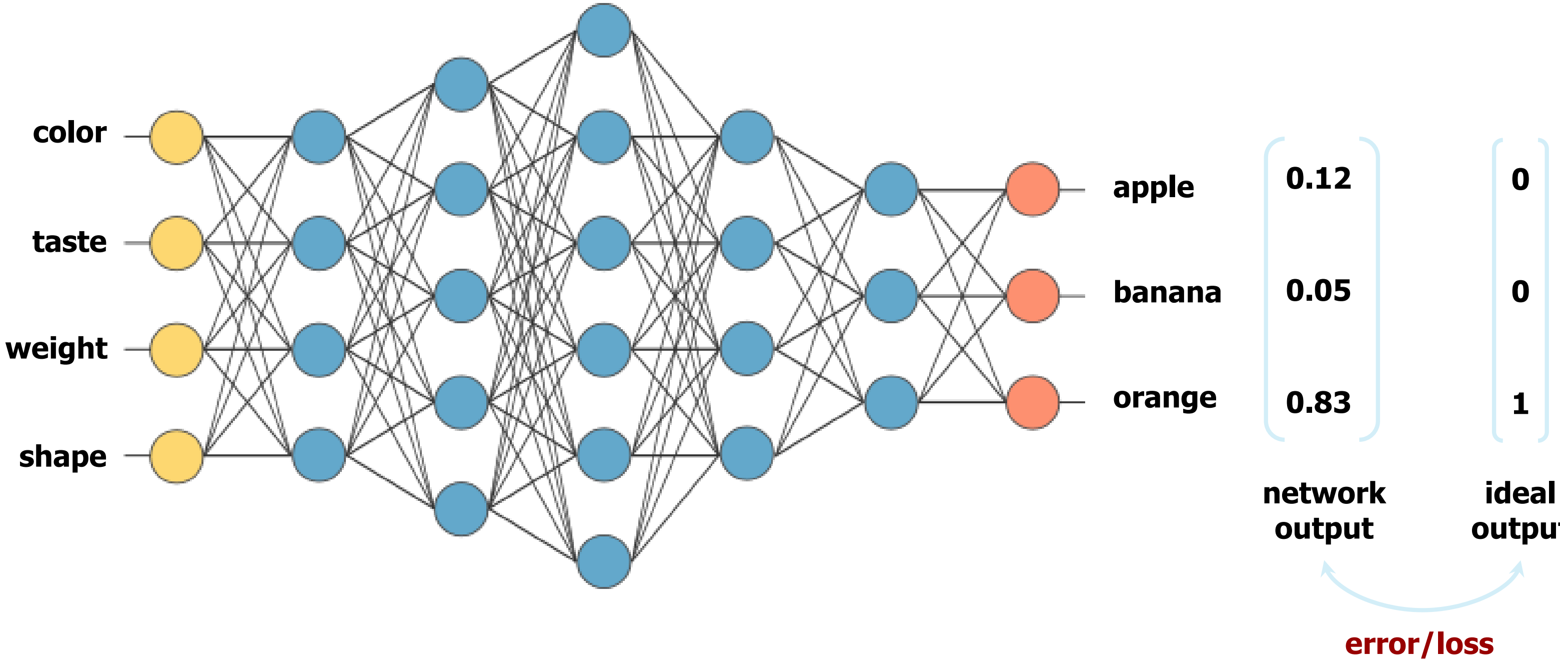
A one-hidden-layer neural network with enough neurons can approximate ***any*** continuous function within the given input range.



non-linear
activation function

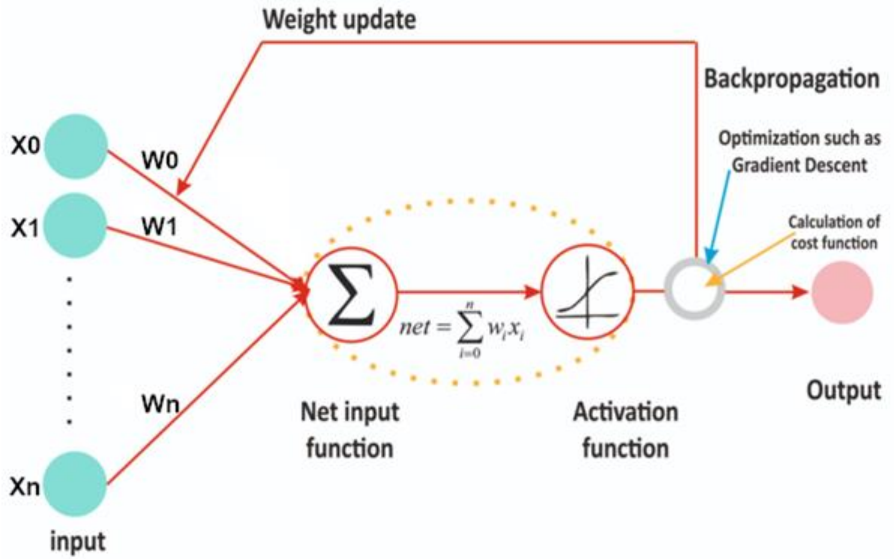


Neural network-based classifier



Neural network training

Loss and gradient descent algorithm



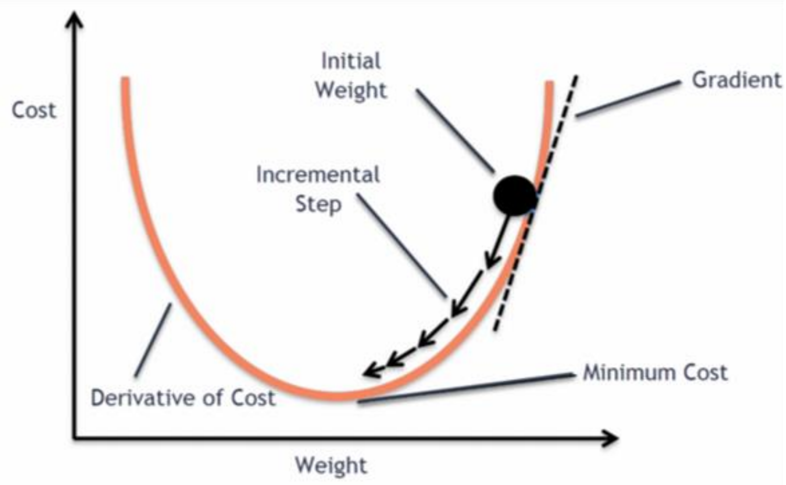
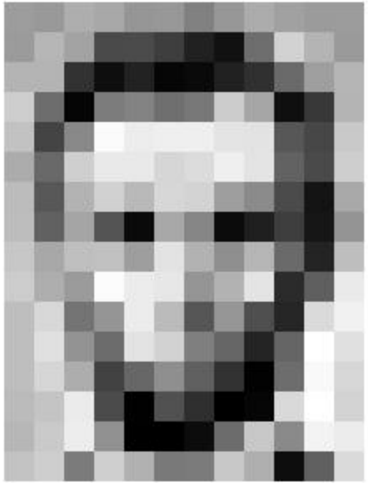
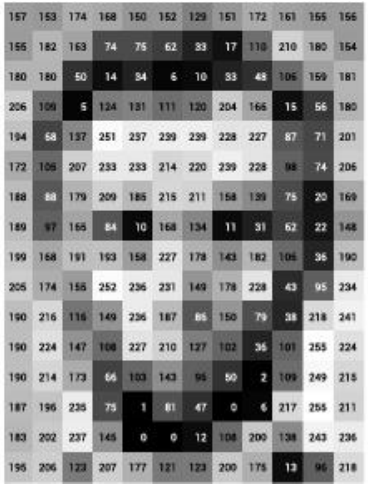


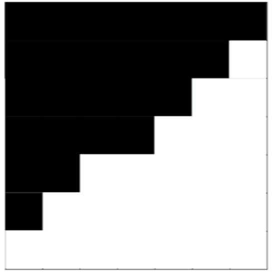
Image as an input data



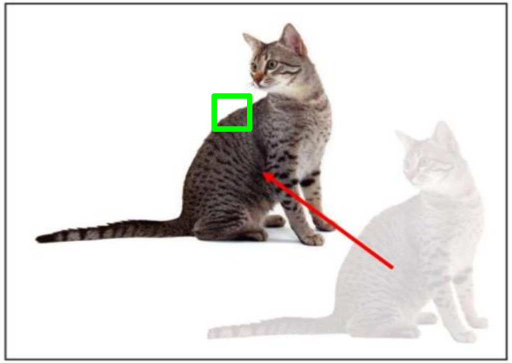


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

0	0	0	0	0	0	0
0	0	0	0	0	0	255
0	0	0	0	0	255	255
0	0	0	0	255	255	255
0	0	255	255	255	255	255
0	255	255	255	255	255	255
255	255	255	255	255	255	255



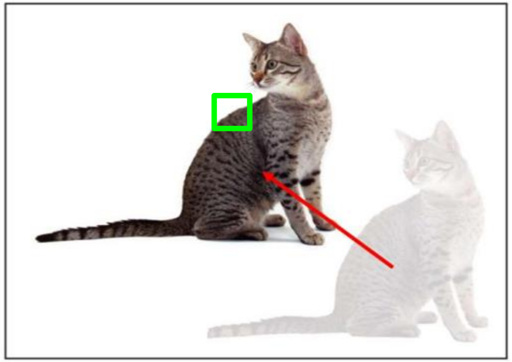
How computer sees an edge

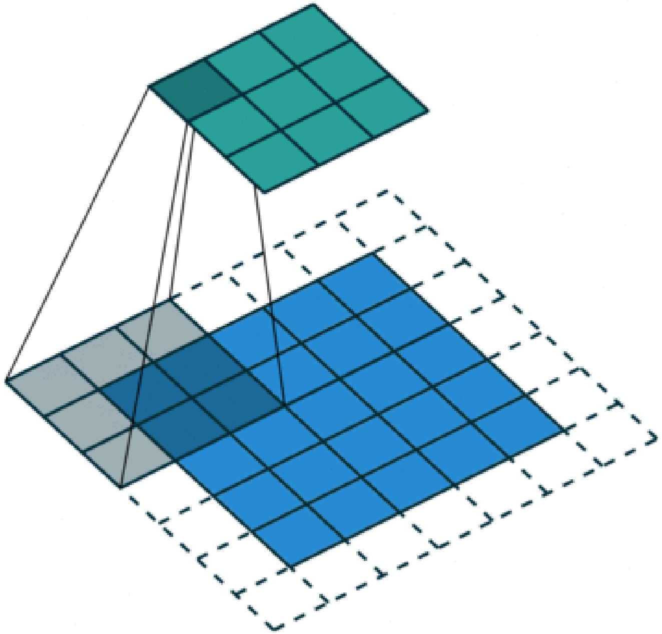


Convolutional vs fully connected

Convolutional layer

- Capture local patterns and spatial relationships between pixels
- Parameter efficiency: shared weights
- Better generalization: translation invariance





Introduction to CNNs

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

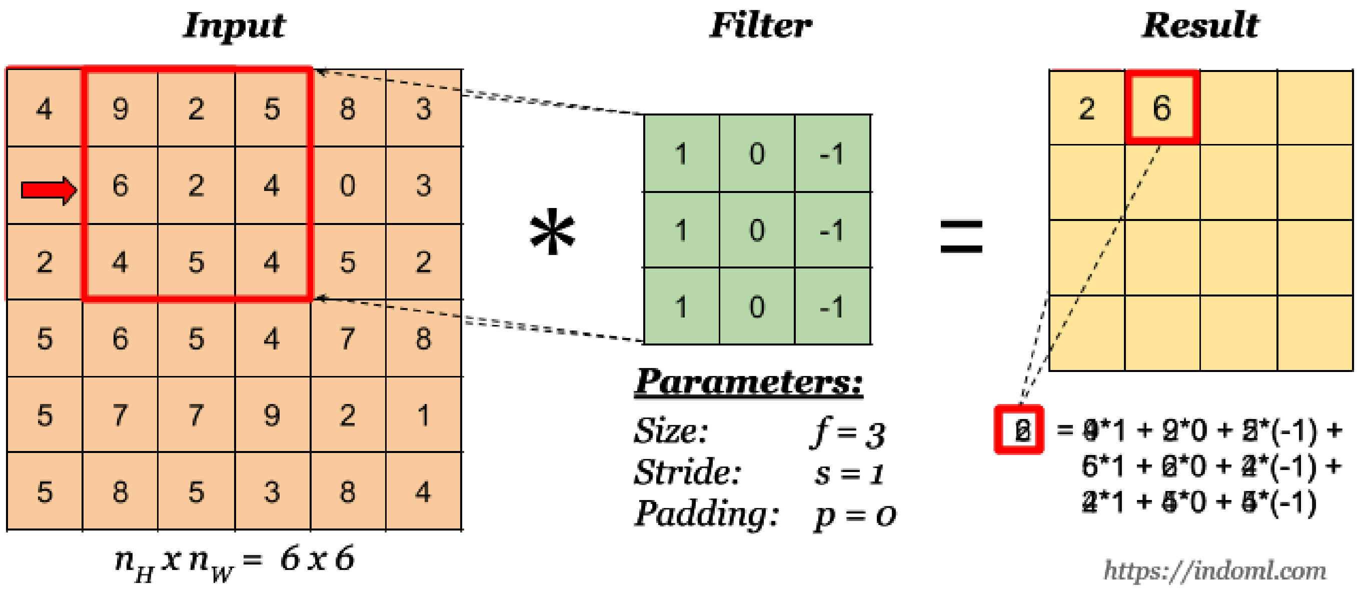
$$308 + -498 + 164 + 1 = -25$$

Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Building blocks of CNNs



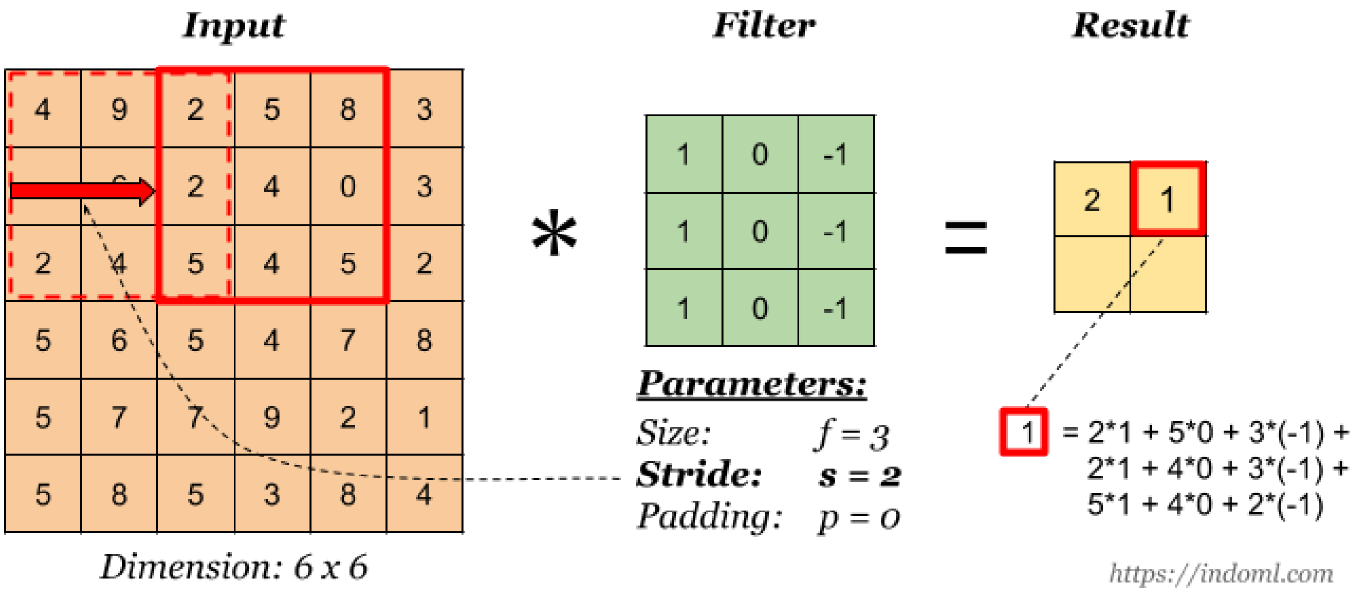
Size (kernel size): The dimensions of the convolutional filter/kernel

Stride: The number of pixels the convolutional filter/kernel moves across the input image or feature map during each convolution operation.

Padding: Additional pixels added around the borders of the input image or feature map to control the output size after convolution and preserve spatial information.

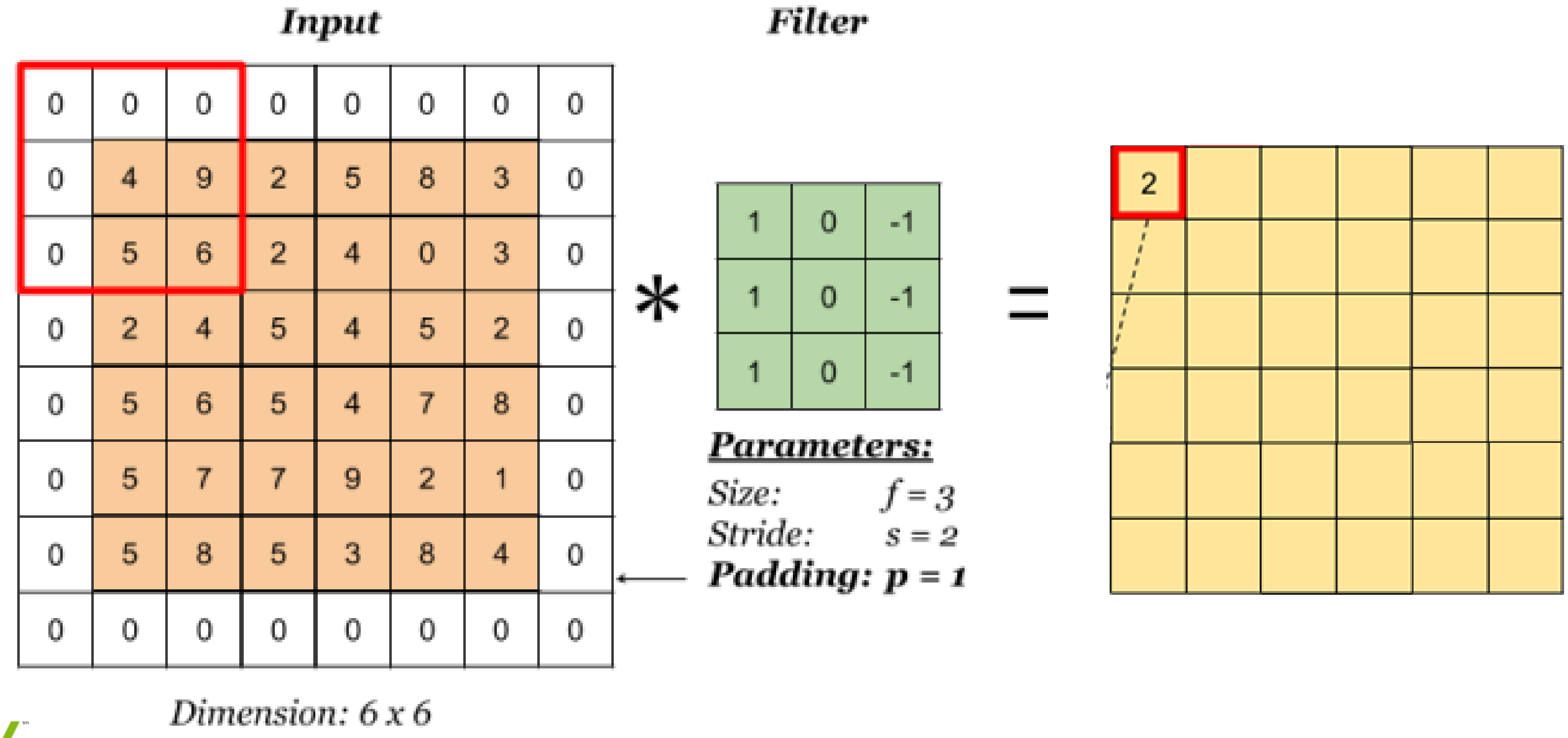
Building blocks of CNNs

Stride

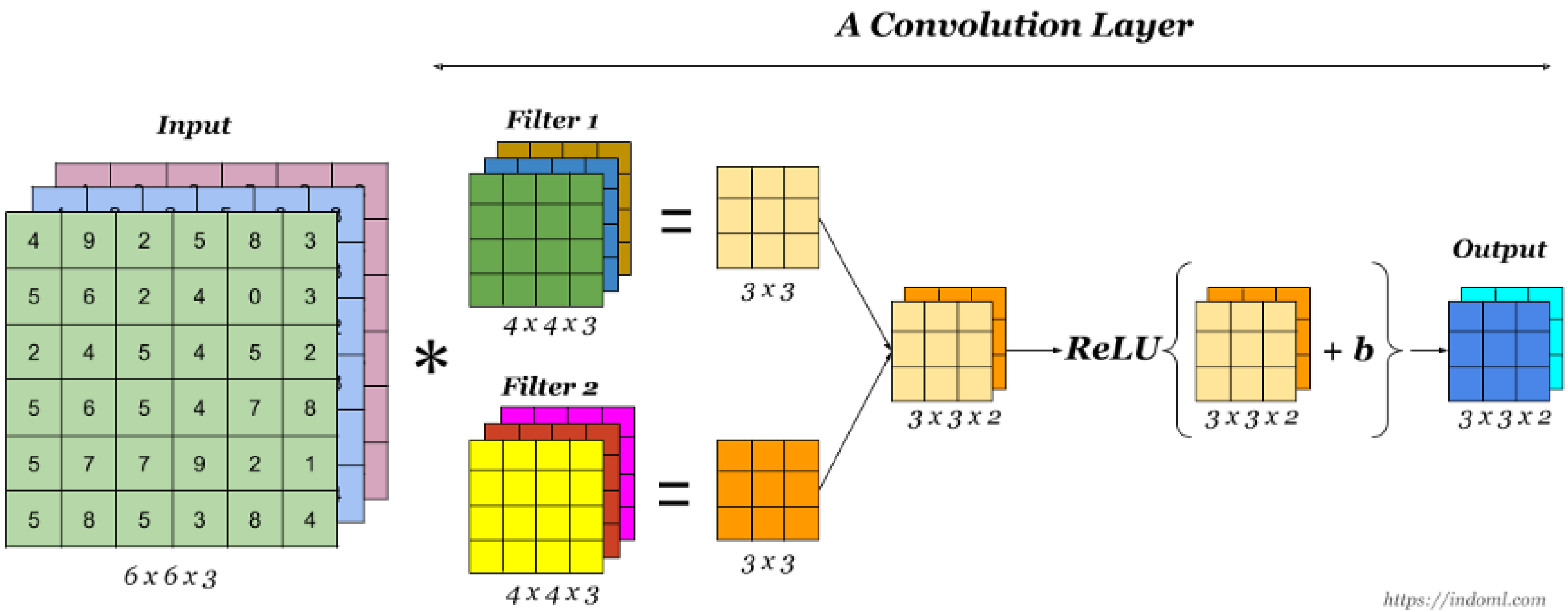


Building blocks of CNNs

Padding: same vs. valid



Building blocks of CNNs

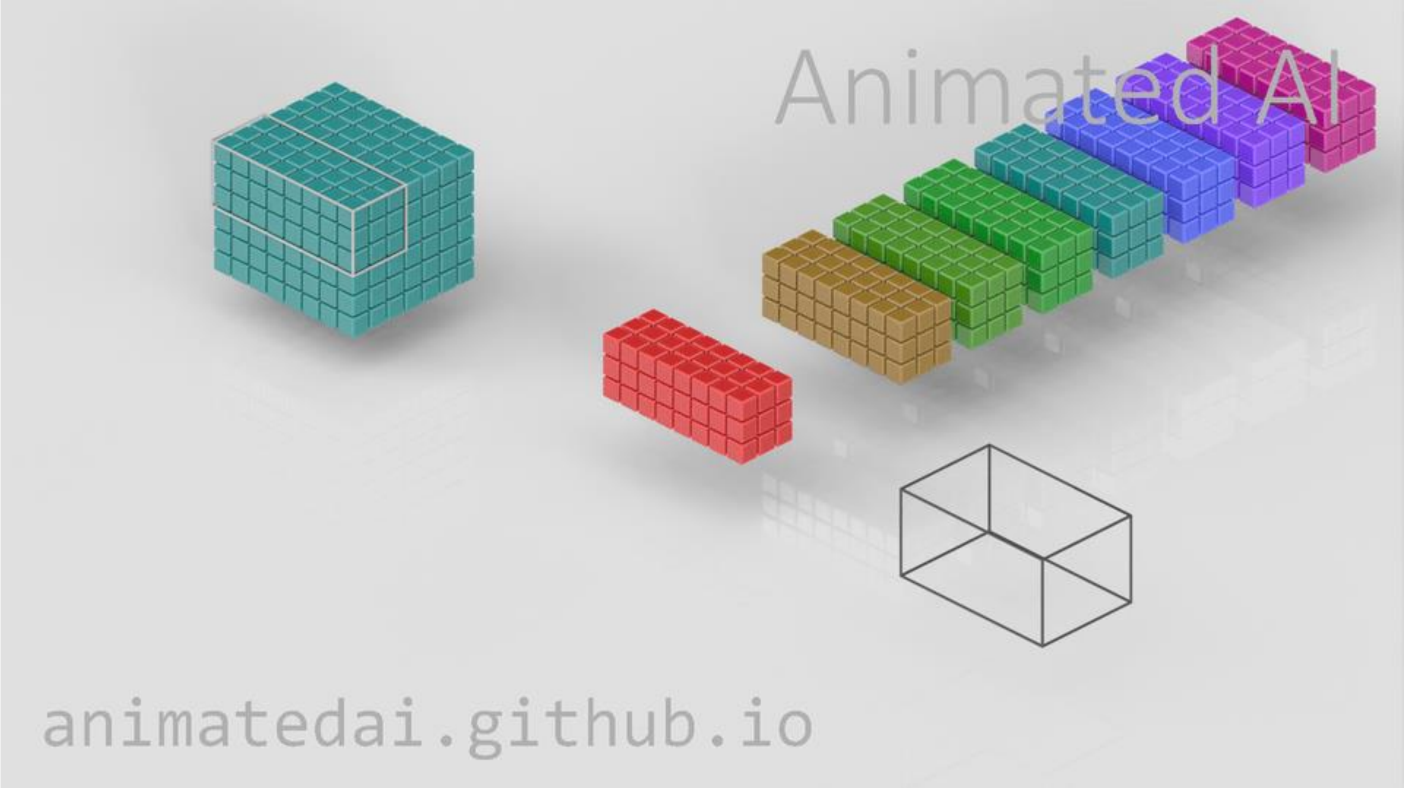


Number of parameters in a convolutional layer

Number of parameters for a K×K kernel:

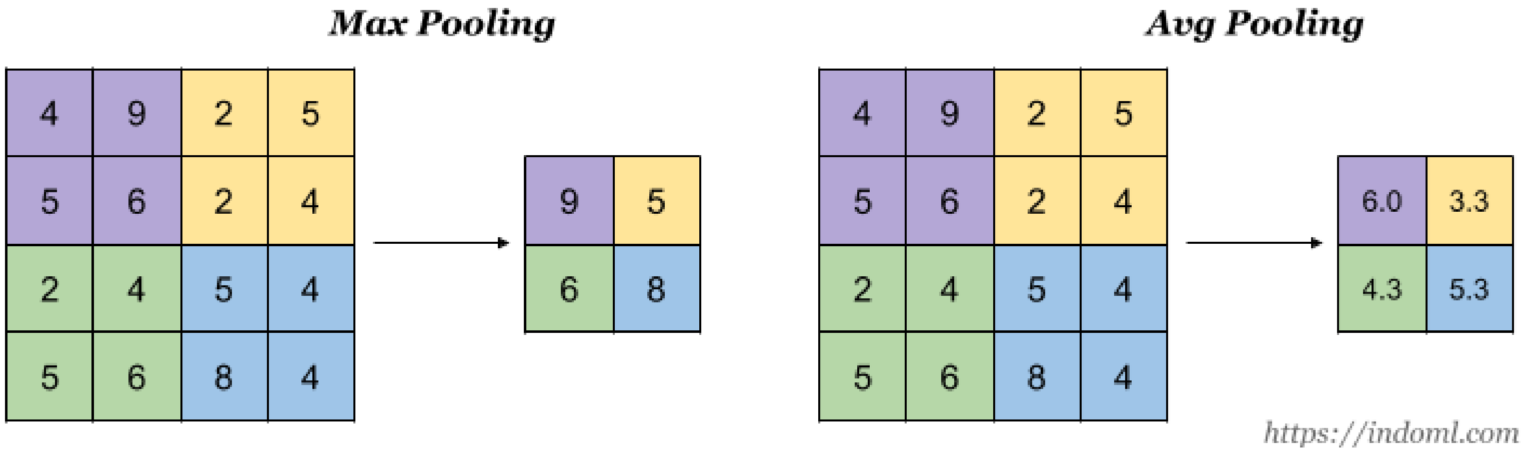
$(K \times K \times N + 1) \times M$

N: input depth
M: output depth



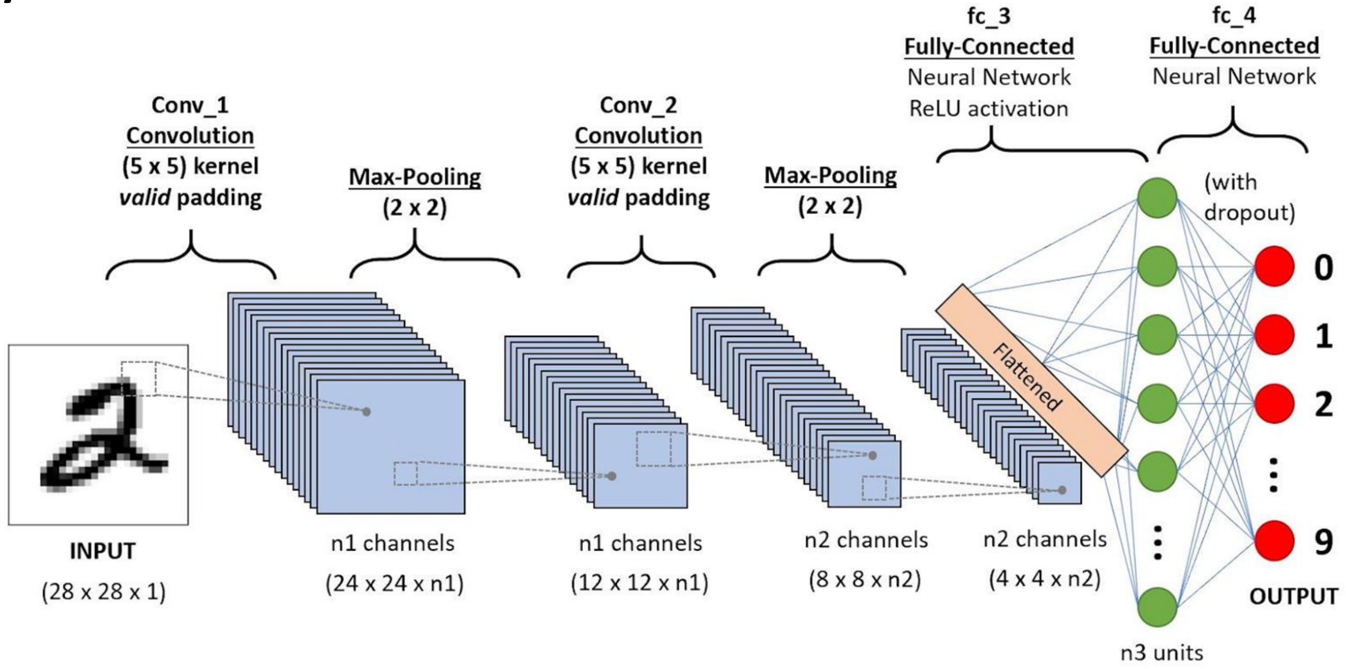
Building blocks of CNNs

Pooling layer

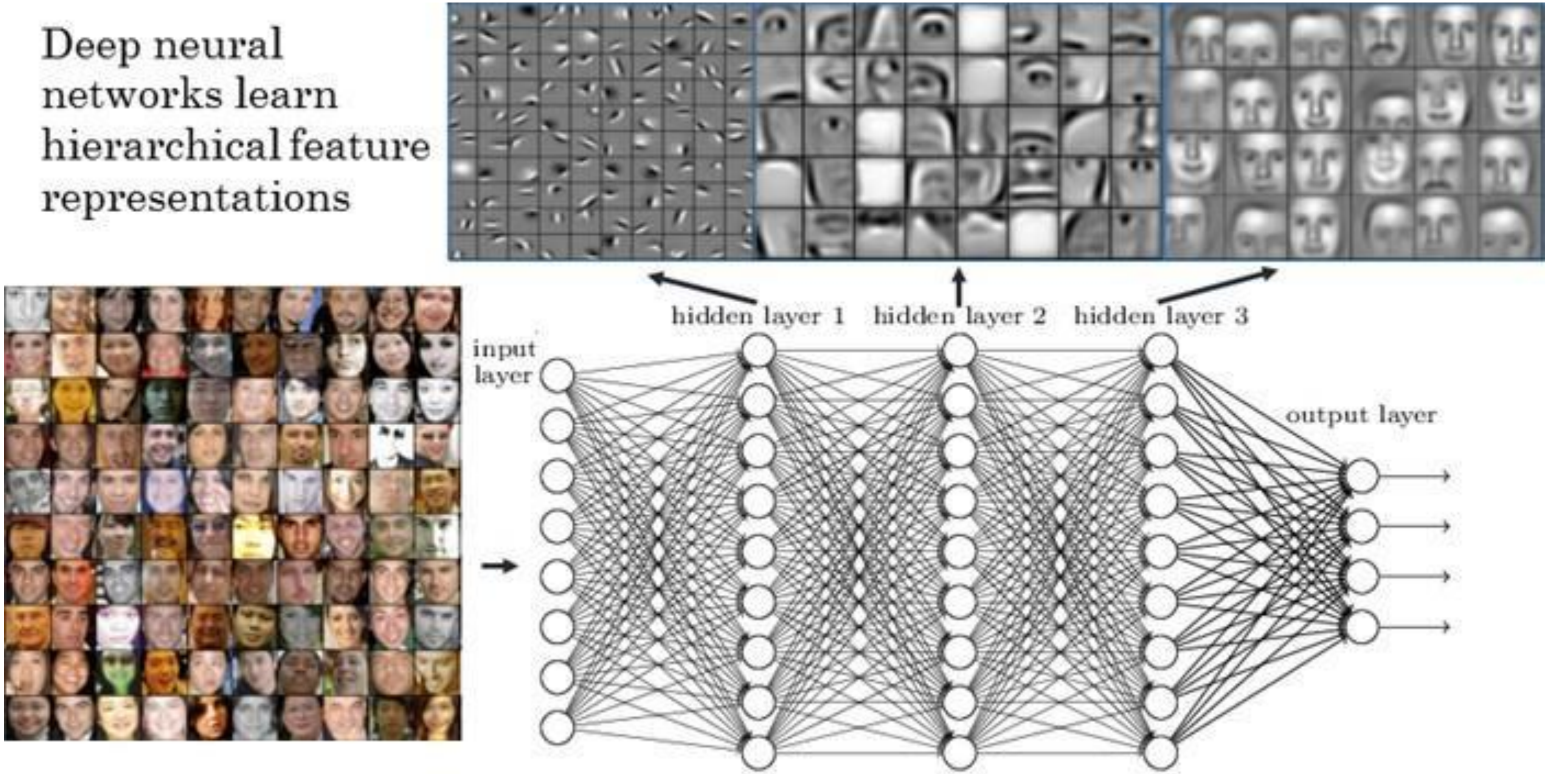


Building blocks of CNNs

A multi-layer CNN

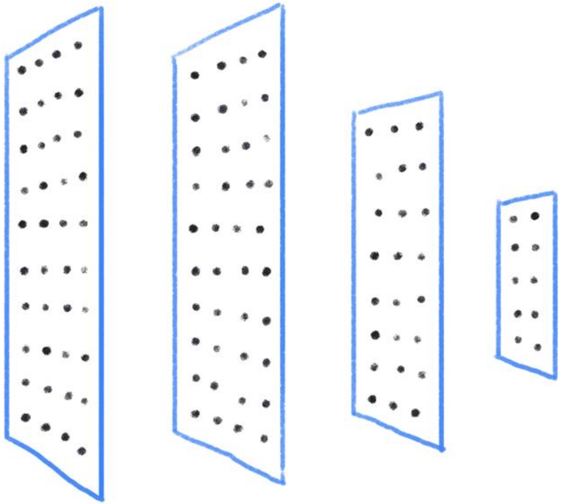


Deep learning is representation learning (a.k.a. feature learning)



Applications of CNNs

Image Classification

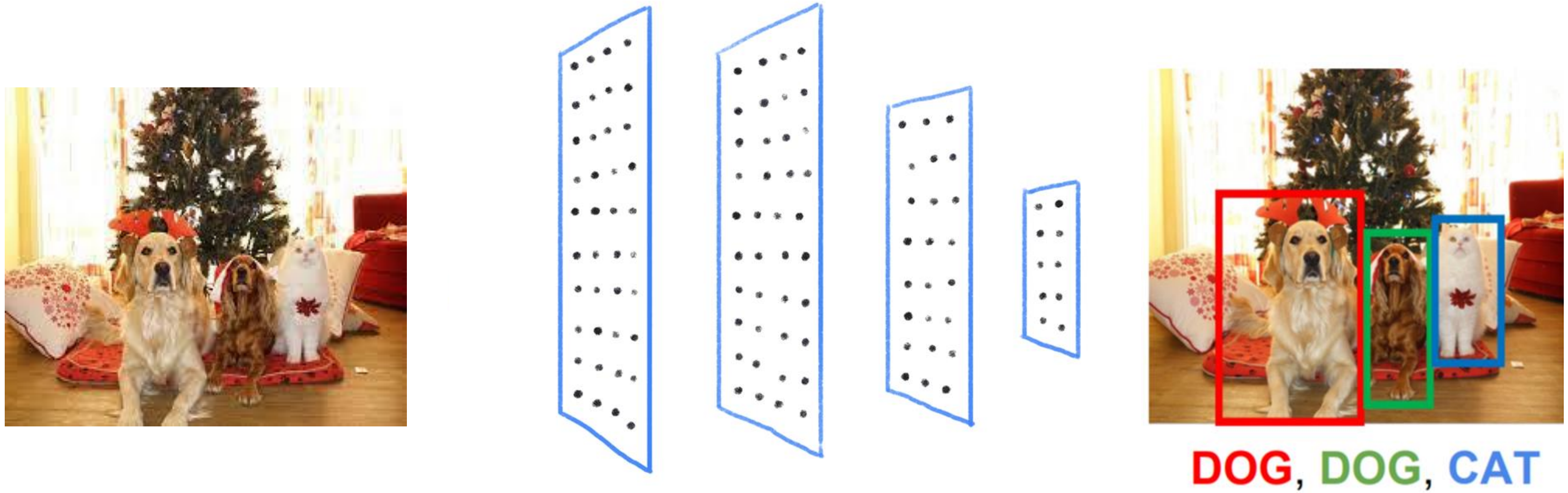


$P_{dog} = 0.9$

$P_{cat} = 0.1$

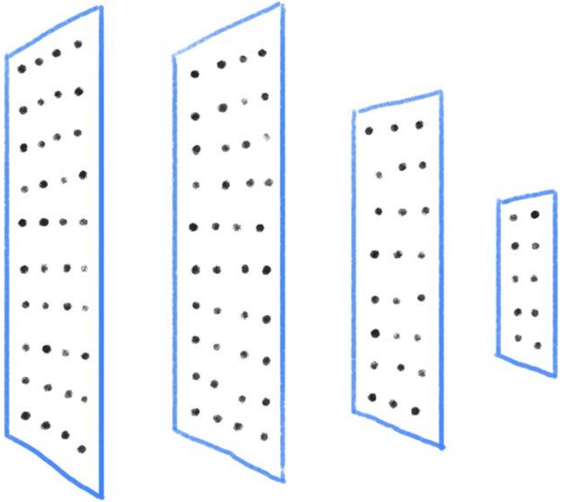
Applications of CNNs

Object Detection



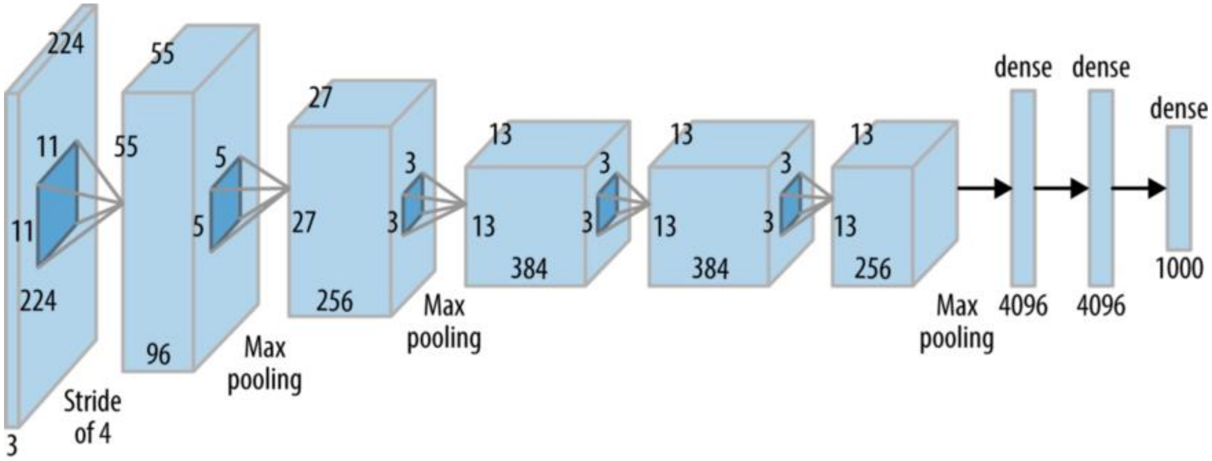
Applications of CNNs

Instance Segmentation



Popular CNN architectures

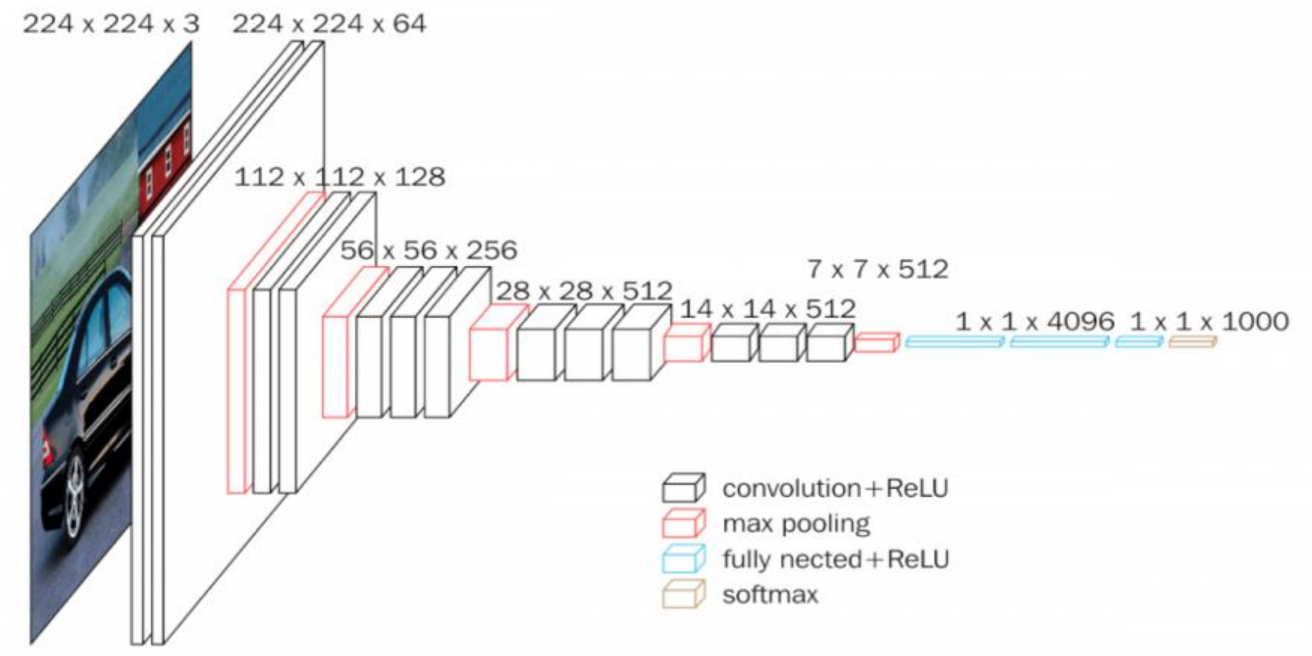
AlexNet (2012) – Top-5 Error 15.3% on ImageNet



"The neural network, which has **60 million parameters** and 500,000 neurons, consists of **five convolutional layers**, some of which are followed by **max-pooling layers**, and **two globally connected layers** with a final **1000-way softmax**."

Popular CNN architectures

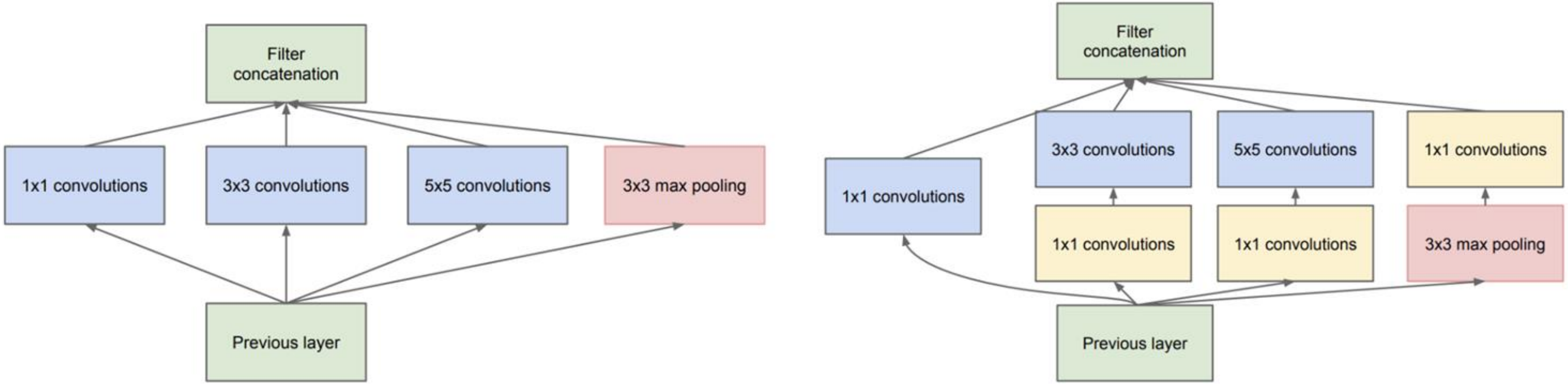
VGG16 (2014) – Top-5 Error 7.32% on ImageNet



Popular CNN architectures

Inception (2014)

Motivation: let the network decide what filter size to put in a layer

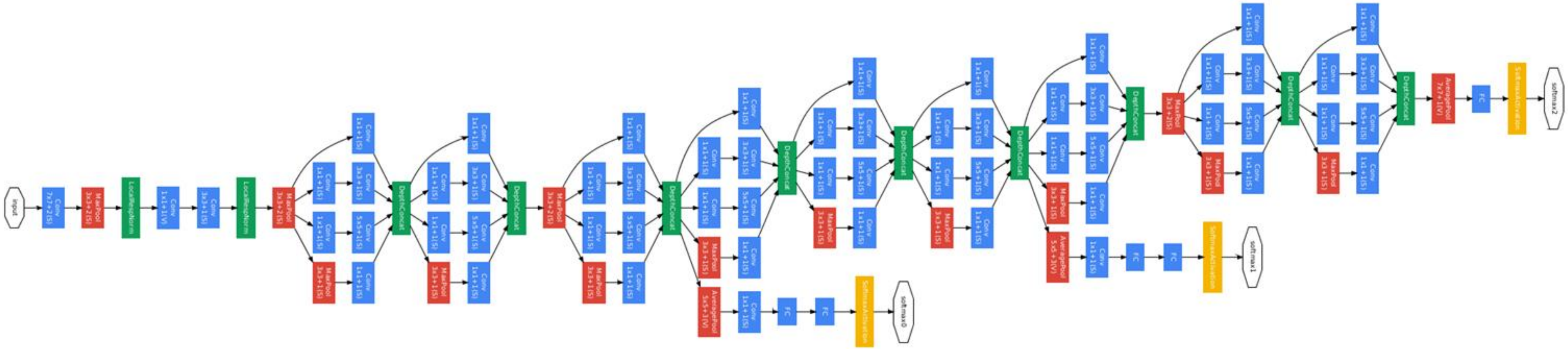


(a) Inception module, naïve version

(b) Inception module with dimension reductions

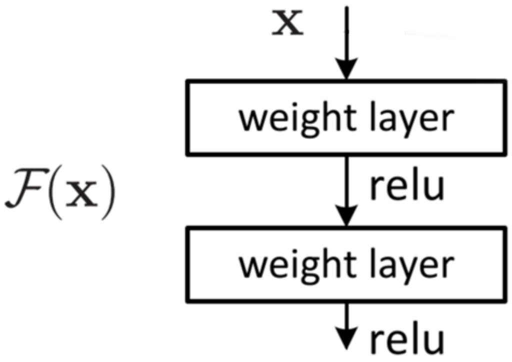
Popular CNN architectures

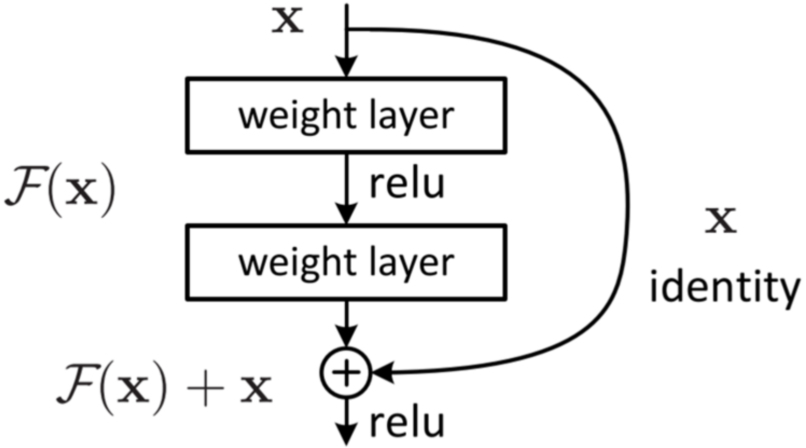
GoogleNet (2014) - Top-5 Error 6.67% on ImageNet



Popular CNN architectures

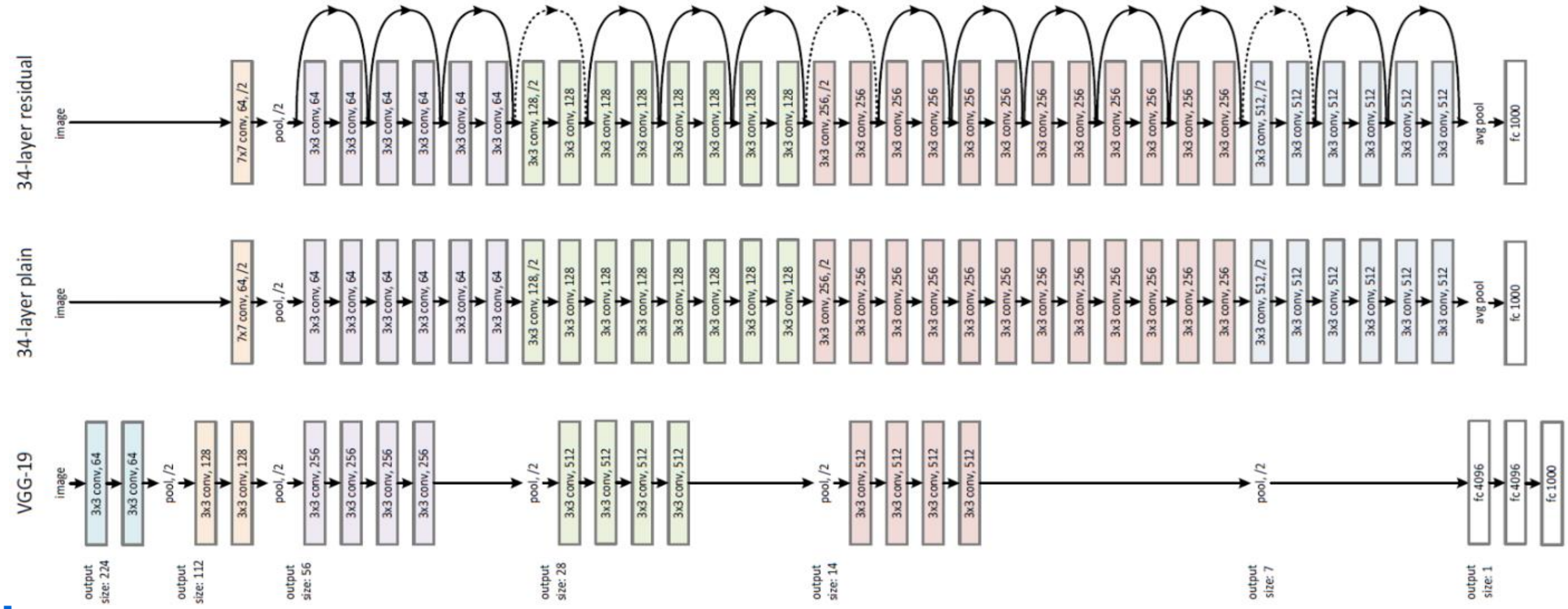
Residual block with a skip connection



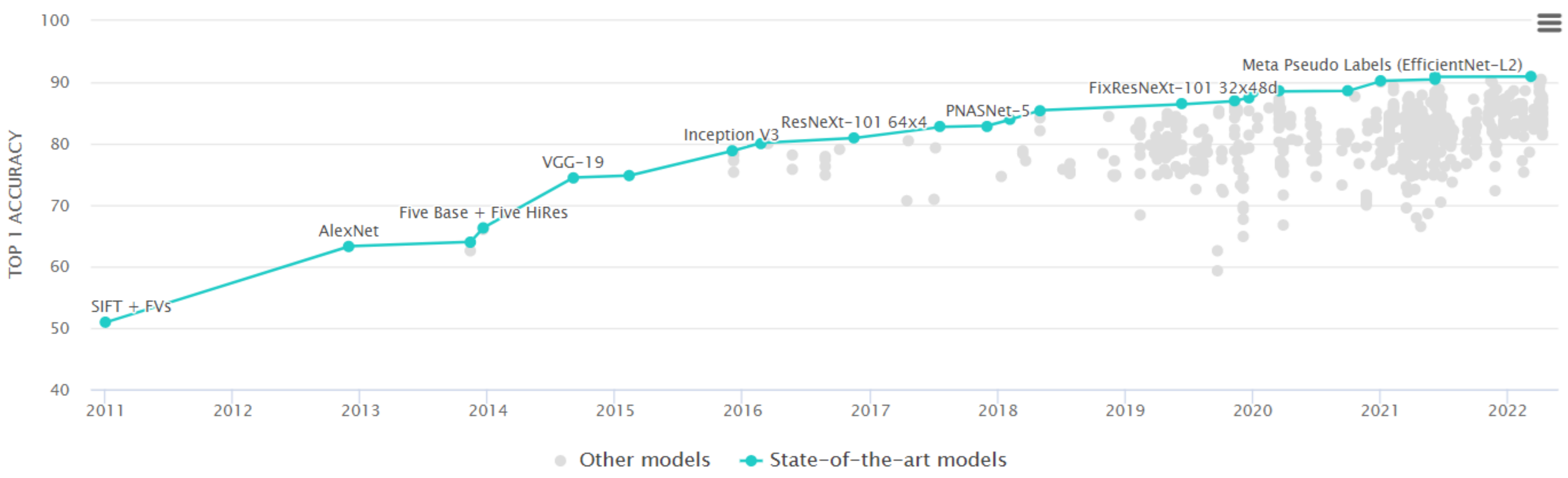


Popular CNN architectures

ResNet (2015) – Top-5 Error 3.57% on ImageNet for ResNet-152

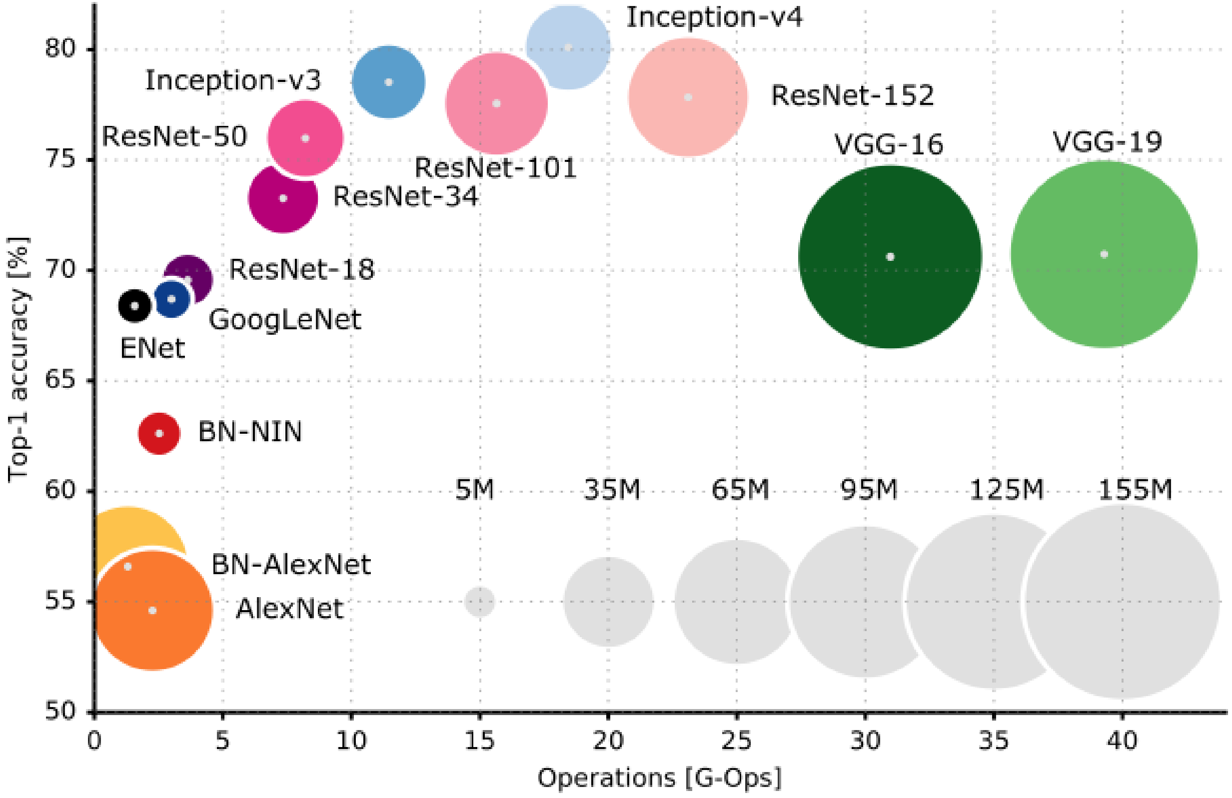


Trend of CNN-based classifiers



https://paperswithcode.com

Trend of CNN-based classifiers

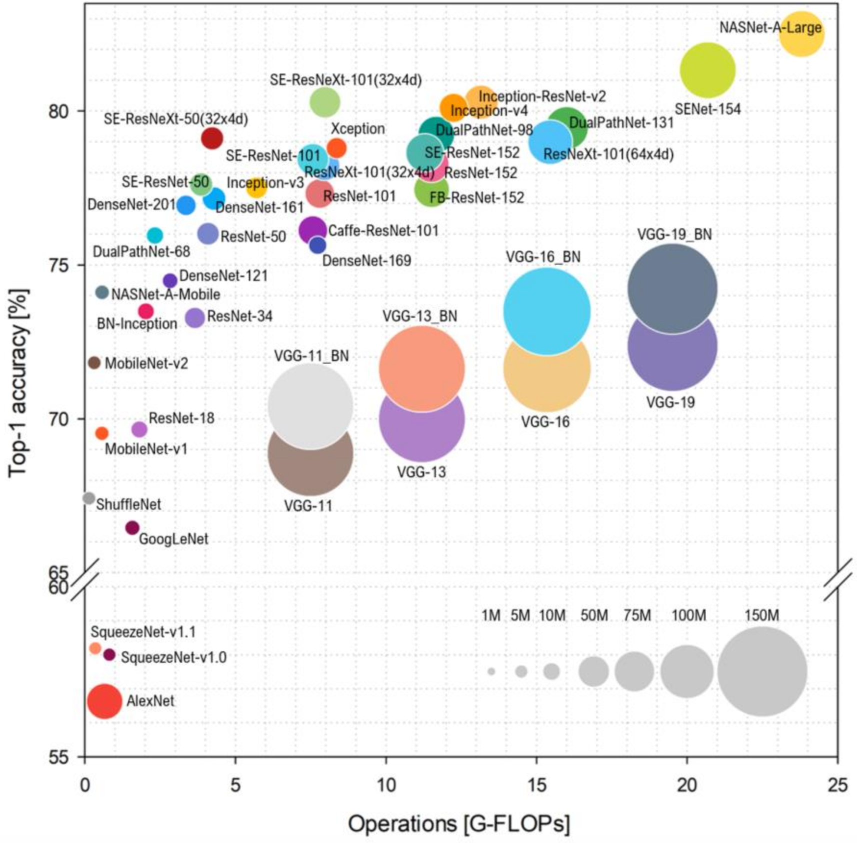


Comparison of popular CNN architectures. The vertical axis shows top 1 accuracy on ImageNet classification. The horizontal axis shows the number of operations needed to classify an image. Circle size is proportional to the number of parameters in the network.

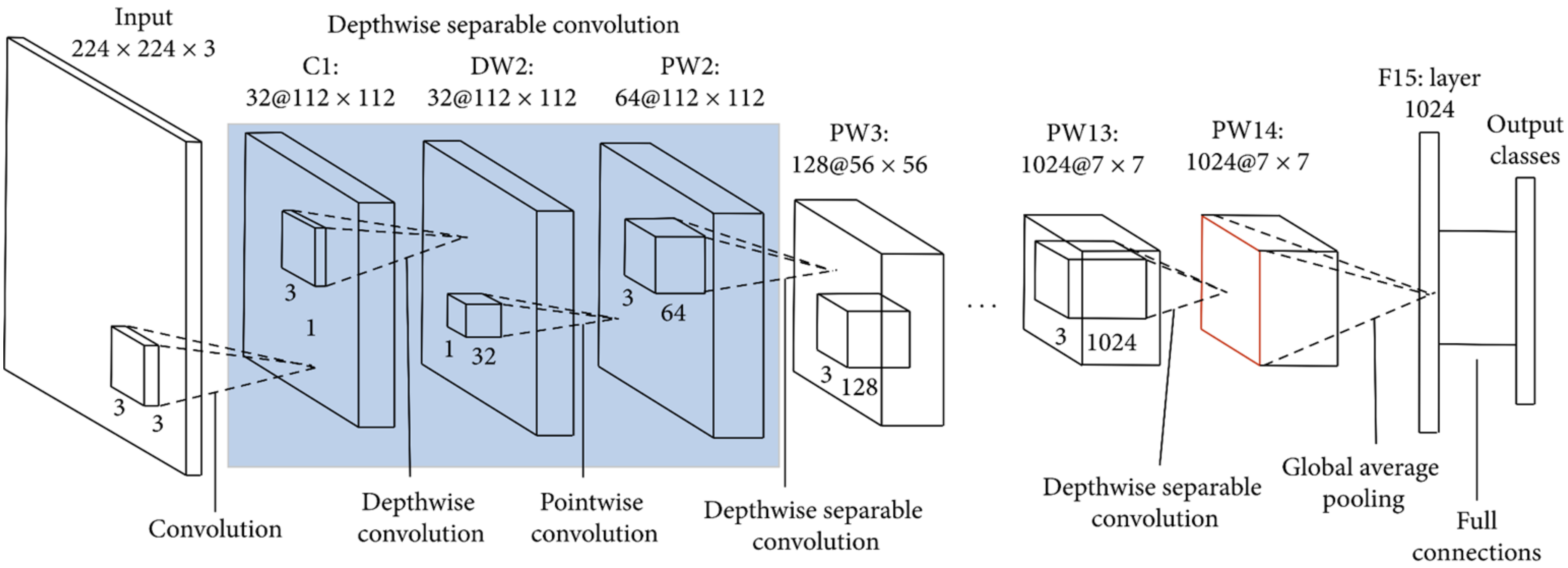
CNNs for edge devices

What do we want on edge?

- Low computational complexity
- Small model size for small memory
- Low energy usage
- Good enough accuracy (depends on application)
- Deployable on embedded processors
- Easily updatable (over-the-air)



MobileNets



Input
224 × 224 × 3
Depthwise separable convolution
C1:
32@112 × 112
DW2:
32@112 × 112
PW2:
64@112 × 112
PW3:
128@56 × 56
PW13:
1024@7 × 7
PW14:
1024@7 × 7
F15: layer
1024
Output
classes
3
3
3
1
1
32
3
64
3
128
3
1024
...
Convolution
Depthwise
convolution
Pointwise
convolution
Depthwise separable
convolution
Depthwise separable
convolution
Global average
pooling
Full
connections

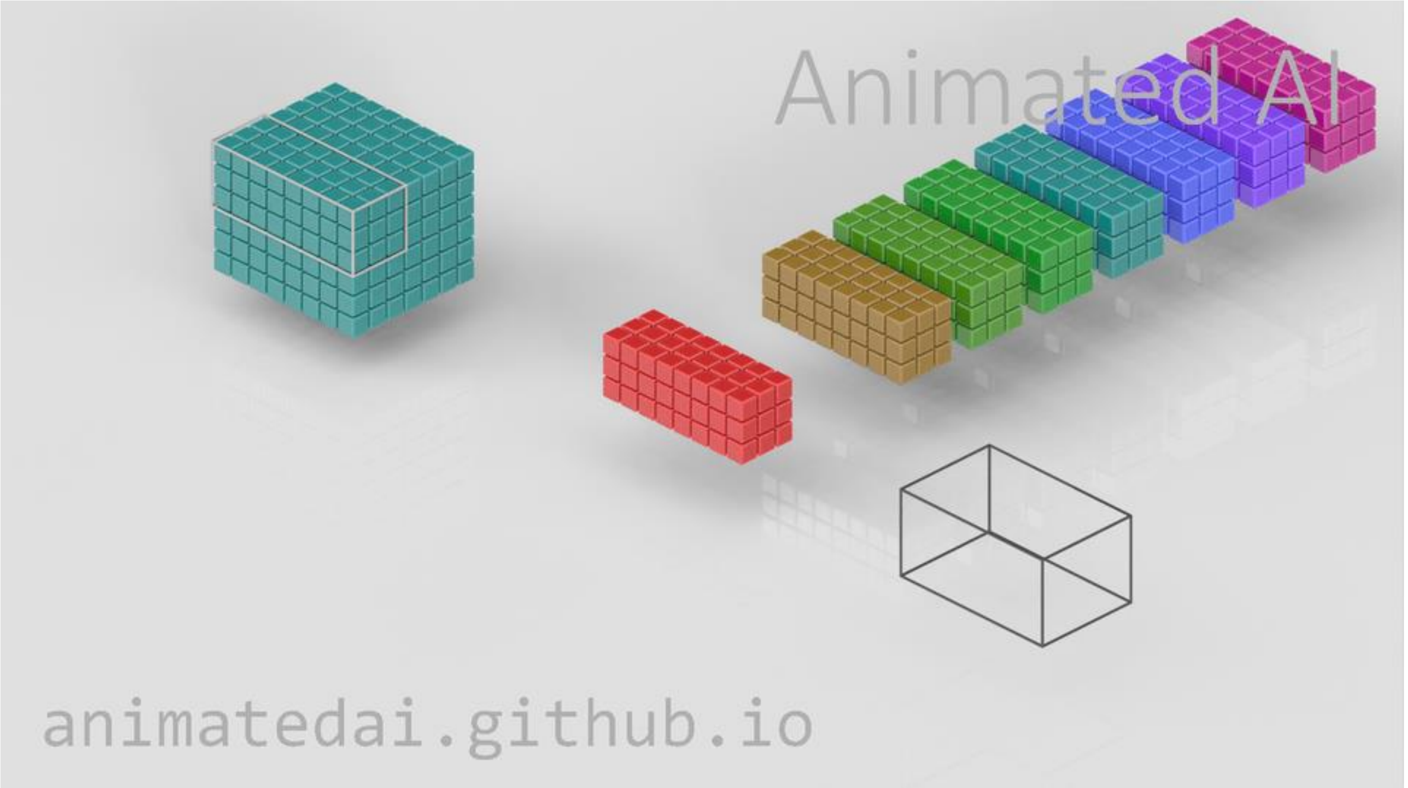
MobileNets

Regular convolution

Number of parameters for a K×K kernel:

$K \times K \times N \times M$

N: input depth
M: output depth



MobileNets

Depthwise separable convolution

Number of parameters:

Depthwise:

- $K \times K \times N$

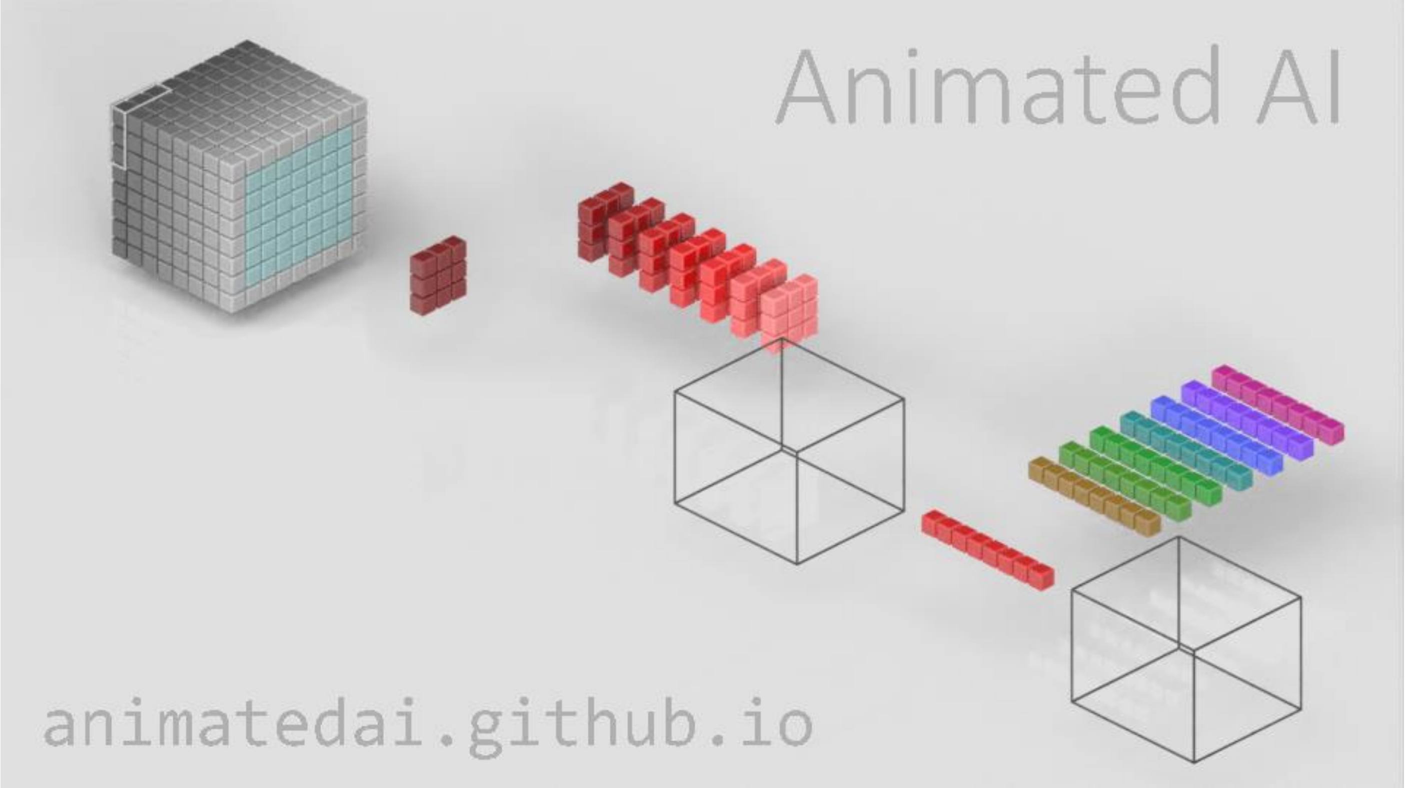
Pointwise:

- $1 \times 1 \times M$

Total:

- $K \times K \times N + M$

N: input depth
M: output depth



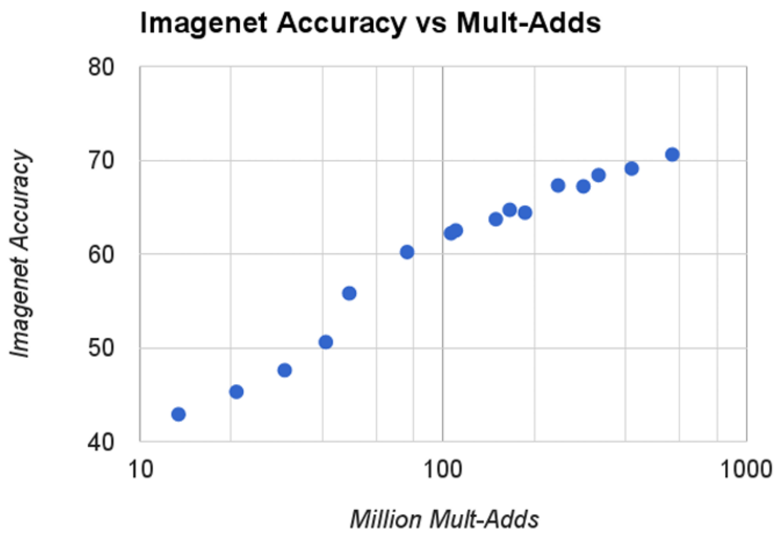
MobileNets

Model shrinking hyperparameter

Depth Multiplier :: Width Multiplier :: alpha :: α

To thin a network uniformly at each layer
Number of channels: M → αM

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5



Log linear dependence between accuracy and computation

EfficientNets

Let's uniformly scale network width, depth, and resolution with a set of fixed scaling coefficients

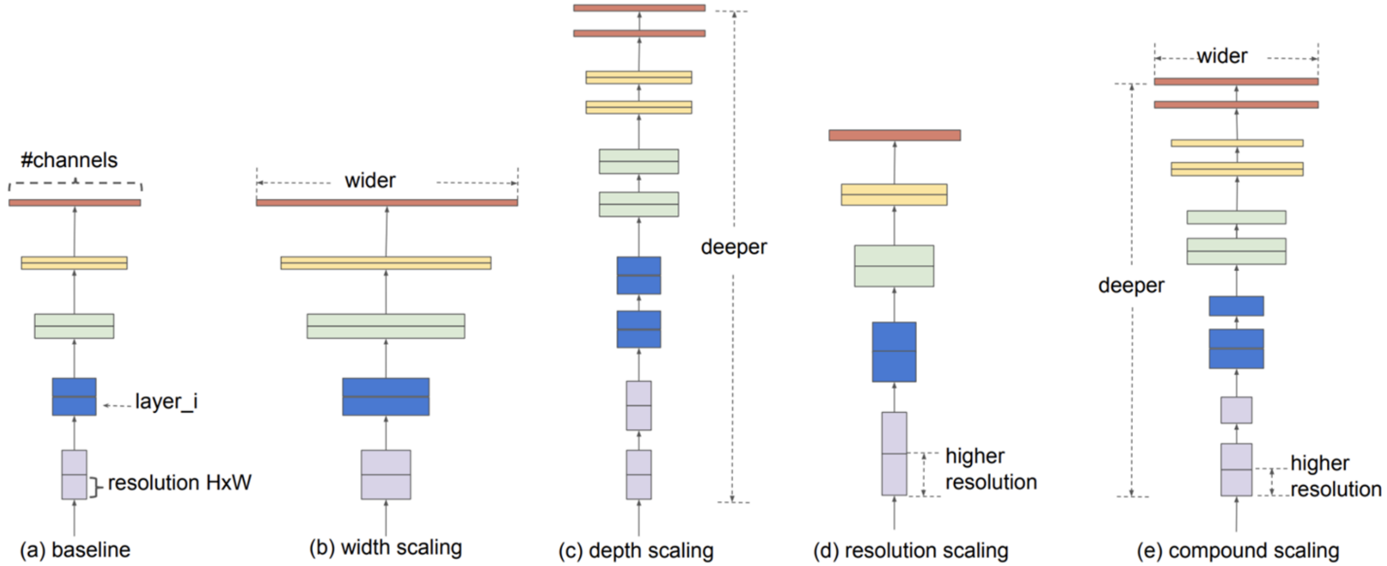
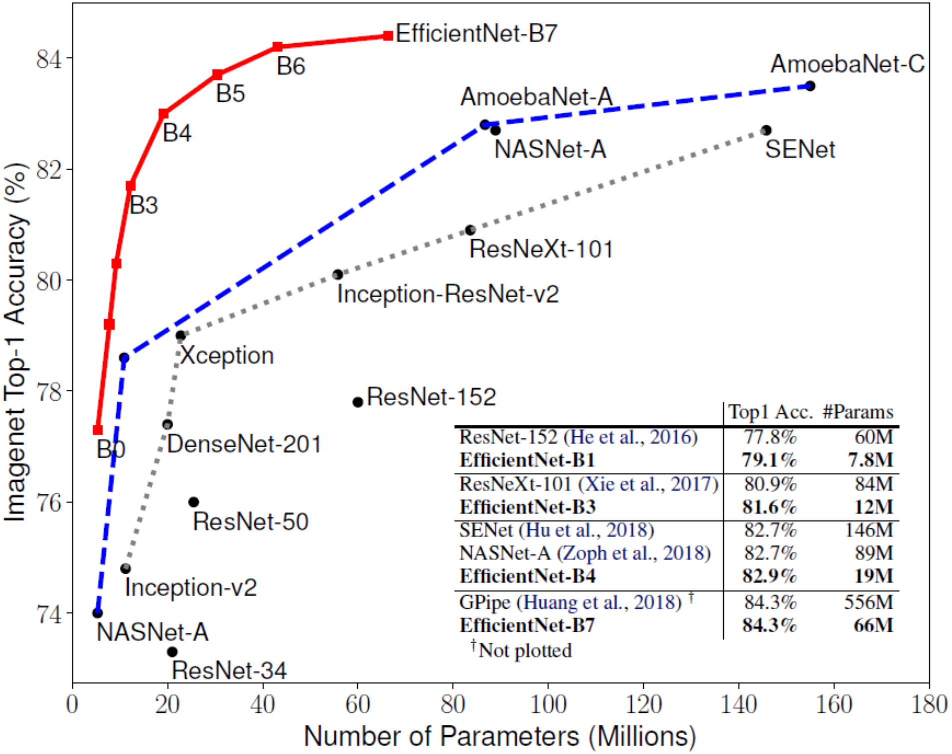


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

EfficientNets



	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**

Note: the baseline B0 architecture is designed using neural architecture search (NAS).

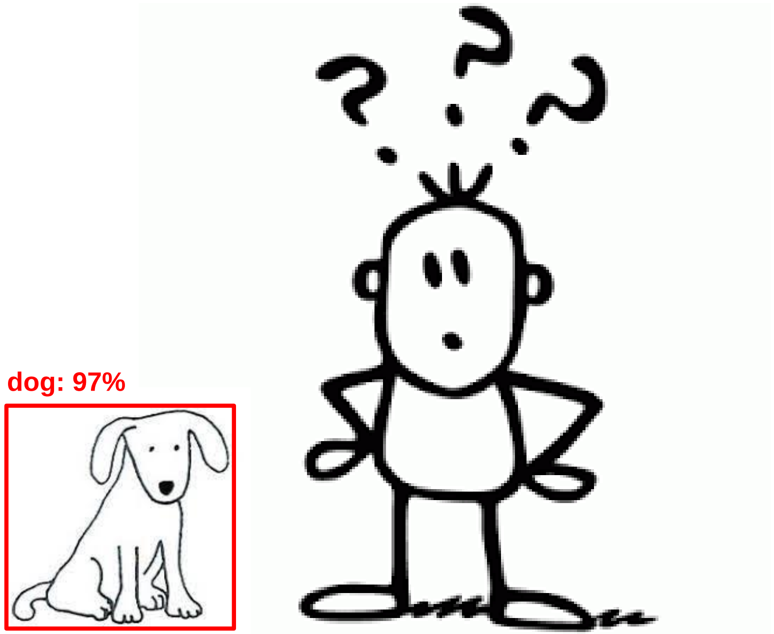
Conclusions

We talked about:

- Deep neural networks and CNNs as the network of choice for computer vision
- The building blocks of CNNs: Convolution layer, pooling layer, padding, stride, etc.
- Application of CNNs in computer vision: image classification, object detection, segmentation, etc.
- CNN architectures: AlexNet, VGG, GoogleNet, ResNet
- Edge-optimized CNNs architectures: MobileNets & EfficientNets

Choosing the right model for an application and hardware is crucial for accuracy and efficiency.

Any Questions?



Resources

- EfficientNet: https://arxiv.org/abs/1905.11946
- Papers With Code: https://paperswithcode.com
- Understanding of MobileNet: https://wikidocs.net/165429
- New mobile neural network architectures https://machinethink.net/blog/mobile-architectures/
- An Analysis of Deep Neural Network Models for Practical Applications: https://arxiv.org/abs/1605.07678
- Deep Learning Equivariance and Invariance: https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/equivariance.pdf
- IndoML Student Notes: Convolutional Neural Networks (CNN) Introduction: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/
- Beginners Guide to Convolutional Neural Networks: https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d
- A Comprehensive Guide to Convolutional Neural Networks: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53