Convolutional Neural Networks

Dr. Xiaowei Huang

https://cgi.csc.liv.ac.uk/~xiaowei/

Up to now,

- Overview of Machine Learning
- Traditional Machine Learning Algorithms
- Deep learning
  - Introduction to Tensorflow
  - Introduction to Deep Learning
  - Functional view and features
  - Backward and forward computation (including backpropogation and chain rule)

Topics

- convolutional neural networks (CNN)
  - Fully-connected
  - Convolutional Layer
  - Advantage of Convolutional Layer
  - Zero-padding Layer
  - ReLU Layer
  - Pooling
  - Softmax
- Example
  - LeNet
  - ResNet

```
model = Sequential()

model.add(Convolution2D(nb_filters, nb_conv, nb_conv,
                        border_mode='valid',
                        input_shape=(1, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Convolution2D(nb_filters, nb_conv, nb_conv))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adadelta',
              metrics=['accuracy'])
```

Convolutional layer

ReLU layer

Convolutional layer

ReLU layer

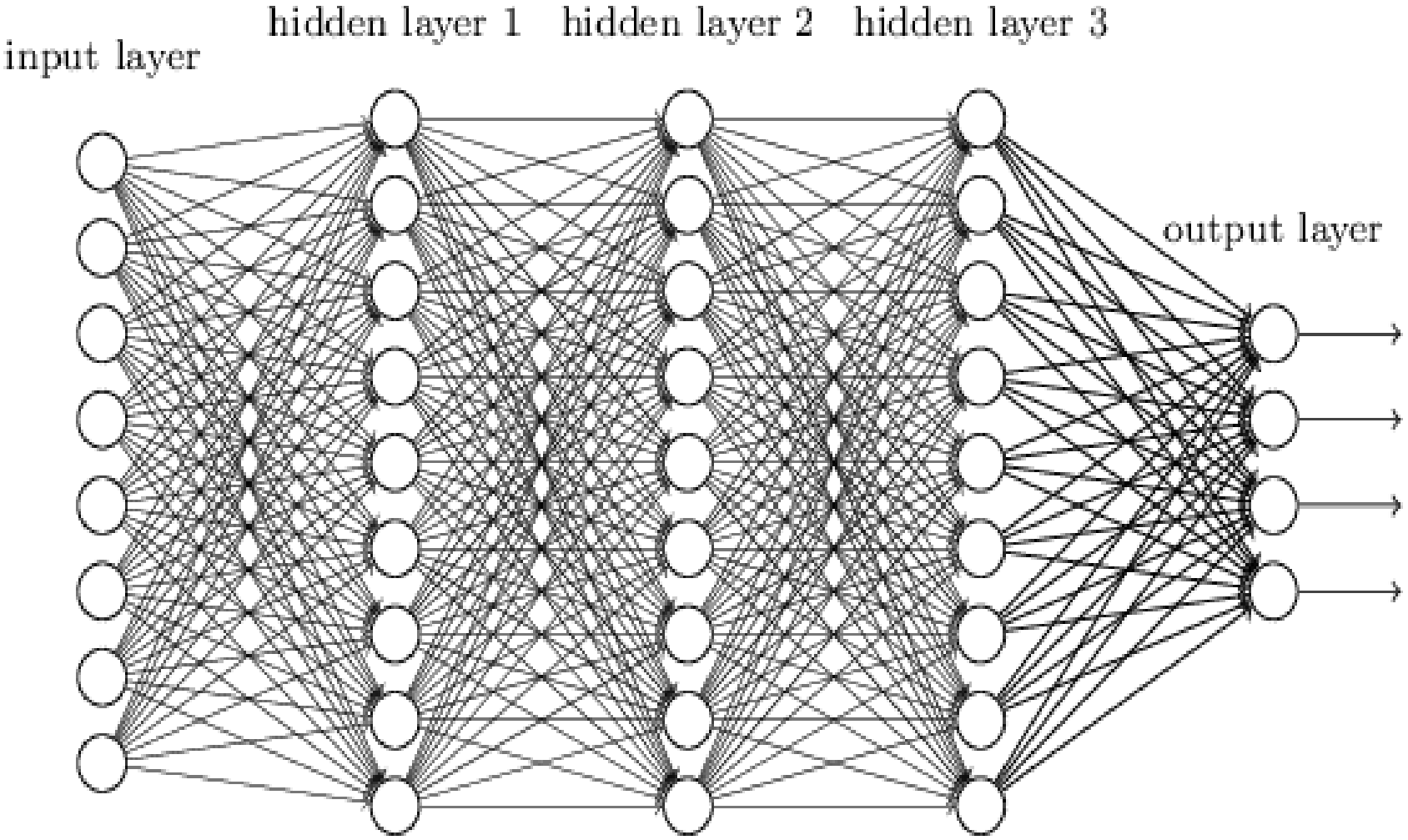
Maxpooling layer

Dropout layer: for regularisation

Flatten layer: from convolutional to fully-connected

Fully-connected layer

Fully-connected


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

Convolution

Convolutional neural networks

- Strong empirical application performance
- Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers

$$h = \sigma(W^T x + b)$$

for a specific kind of weight matrix W

Convolution: math formula

- Given functions $u(t)$ and $w(t)$, their convolution is a function s(t)

$$s(t) = \int u(a)w(t-a)da$$

- Written as

$$s = (u * w) \quad \text{or} \quad s(t) = (u * w)(t)$$

Convolution: discrete version

- Given array u_t and w_t, their convolution is a function s_t

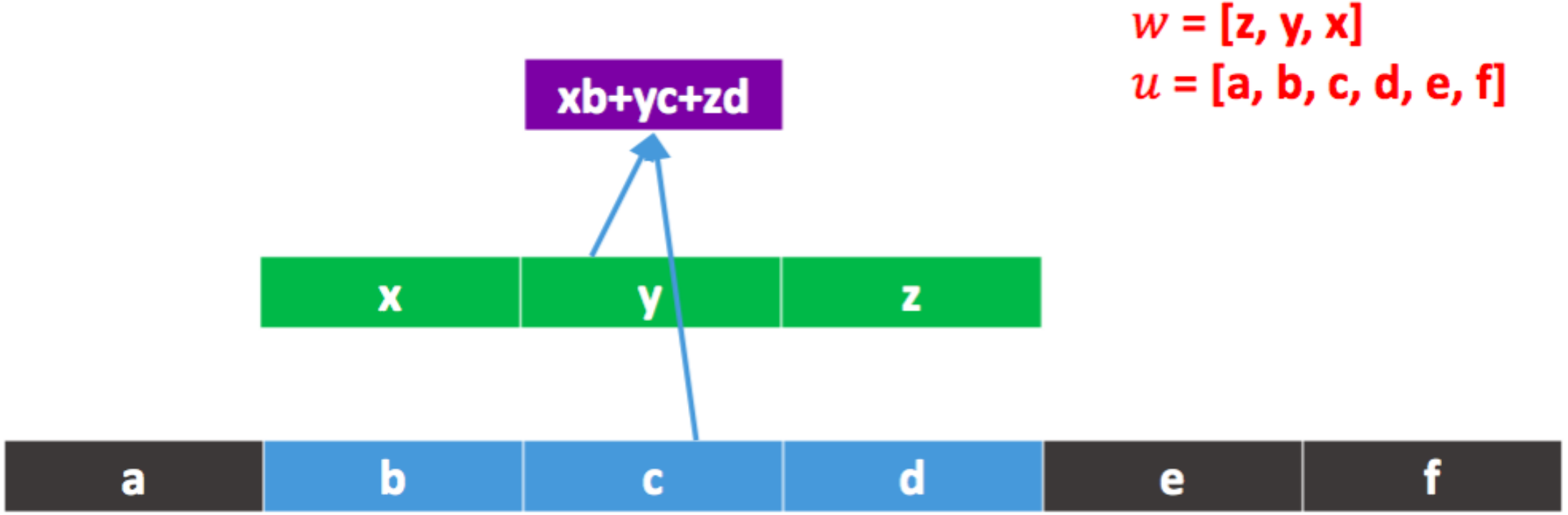
$$s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}$$

- Written as

$$s = (u * w) \quad \text{or} \quad s_t = (u * w)_t$$

- When u_t or w_t is not defined, assumed to be 0

Illustration 1



w = [z, y, x]

u = [a, b, c, d, e, f]

Illustration 1

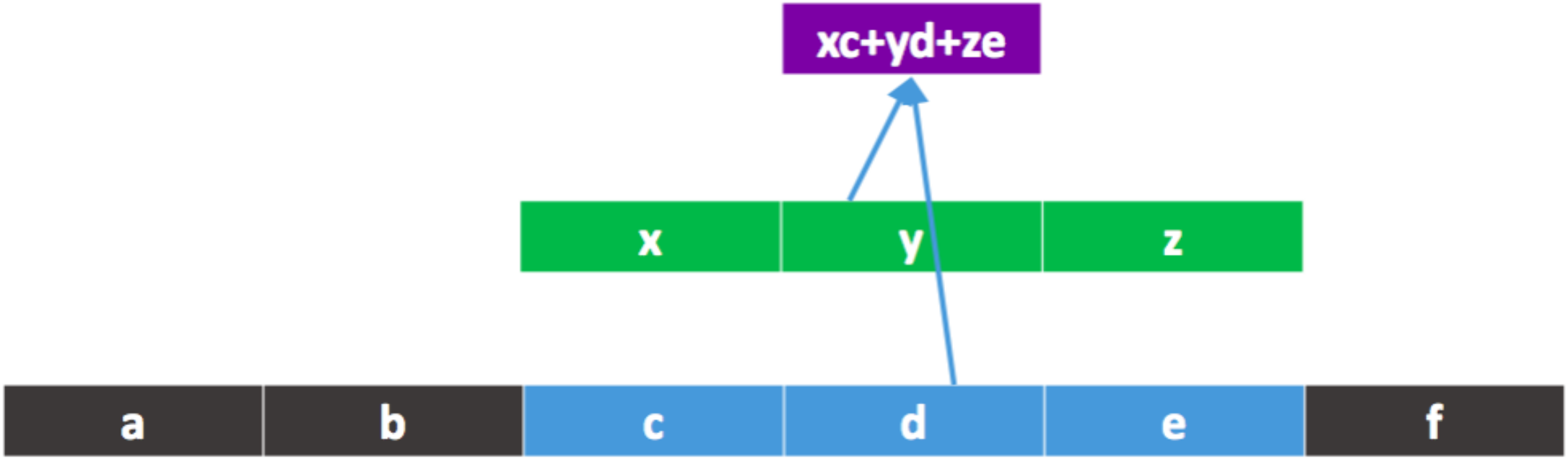


Illustration 1

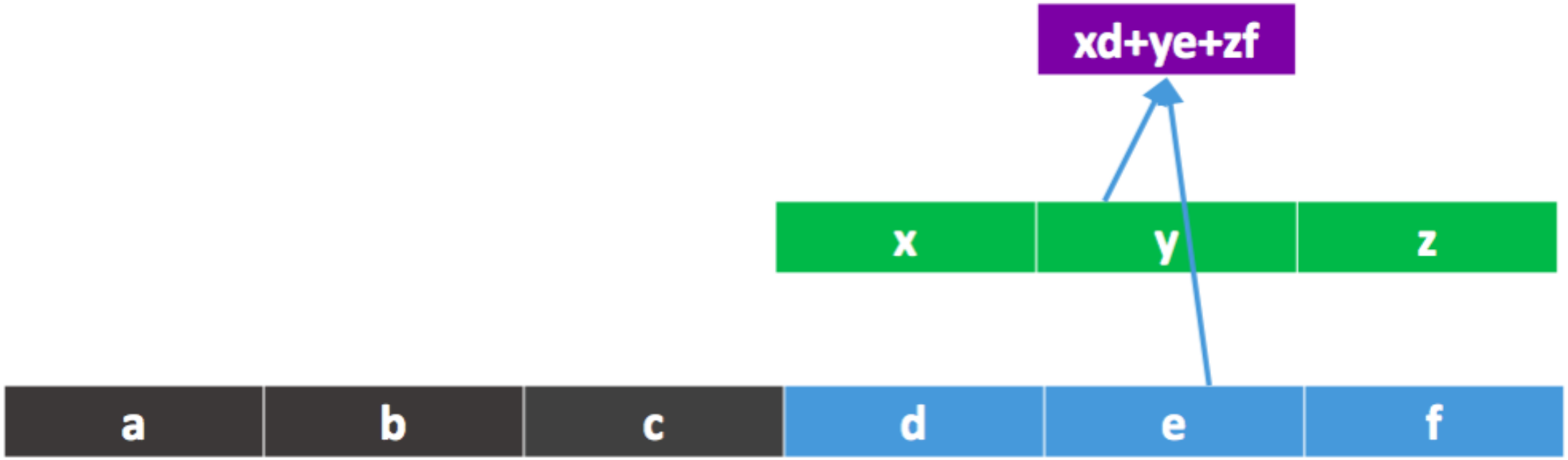


Illustration 1: boundary case

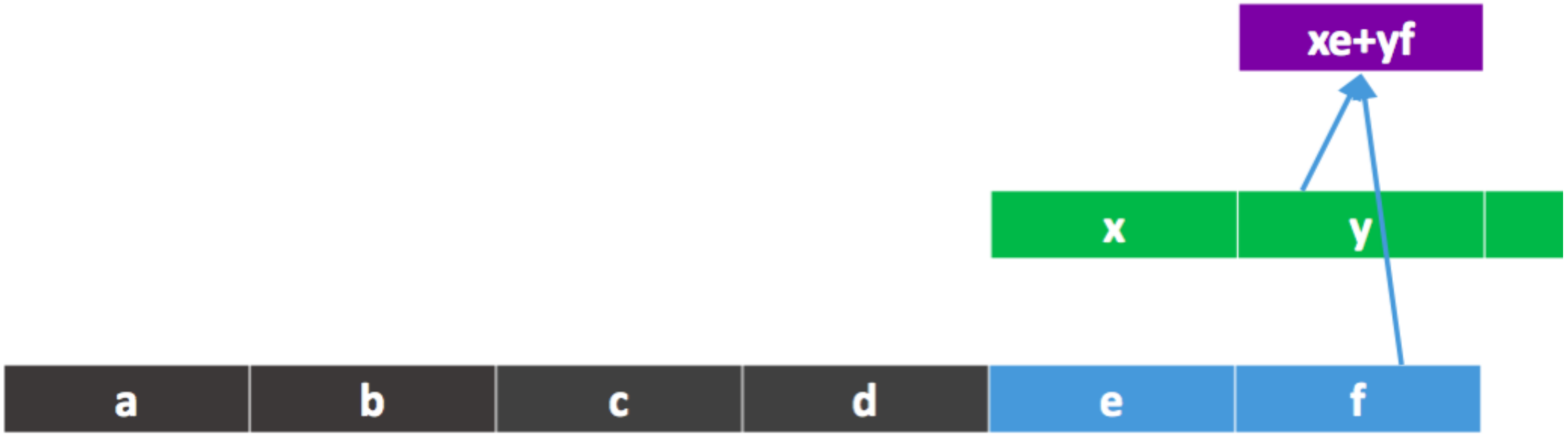


Illustration 1: as matrix multiplication

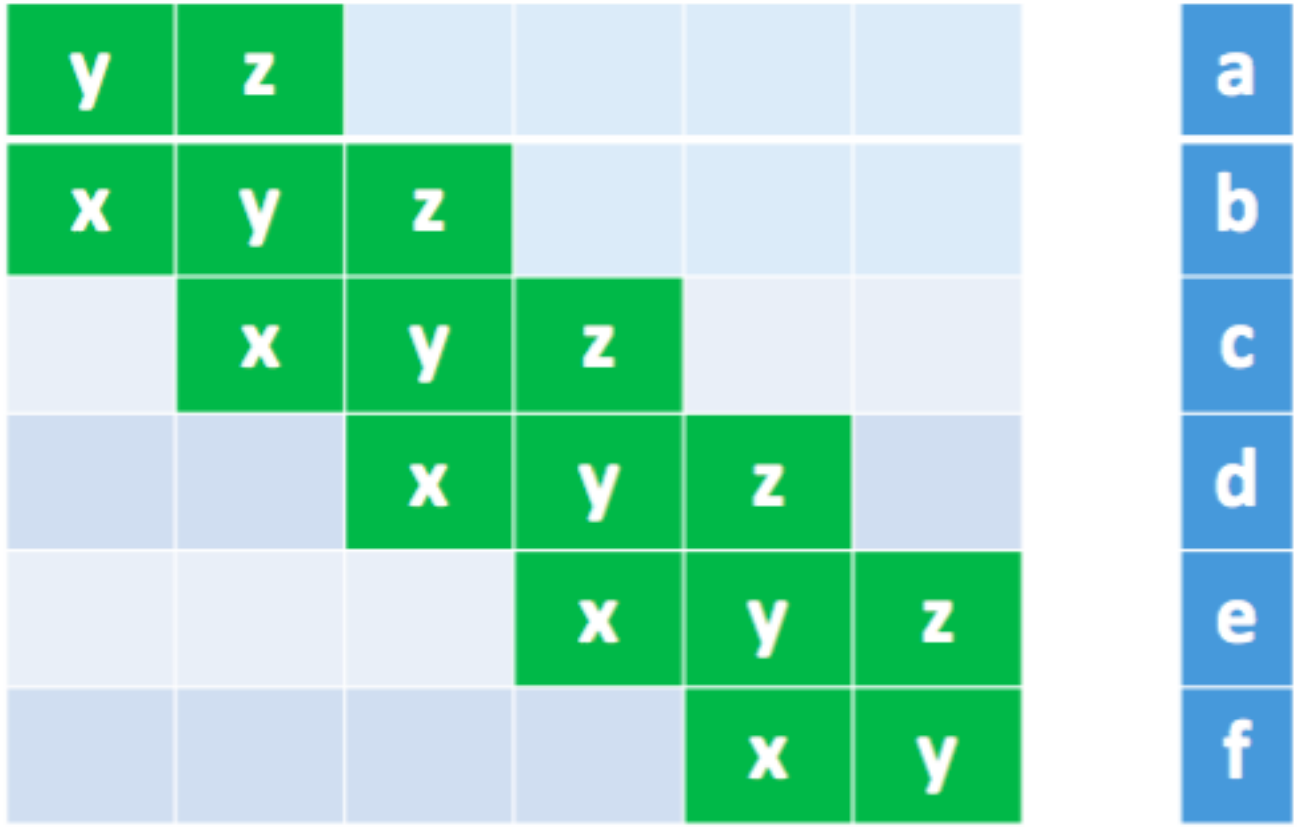


Illustration 2: two dimensional case

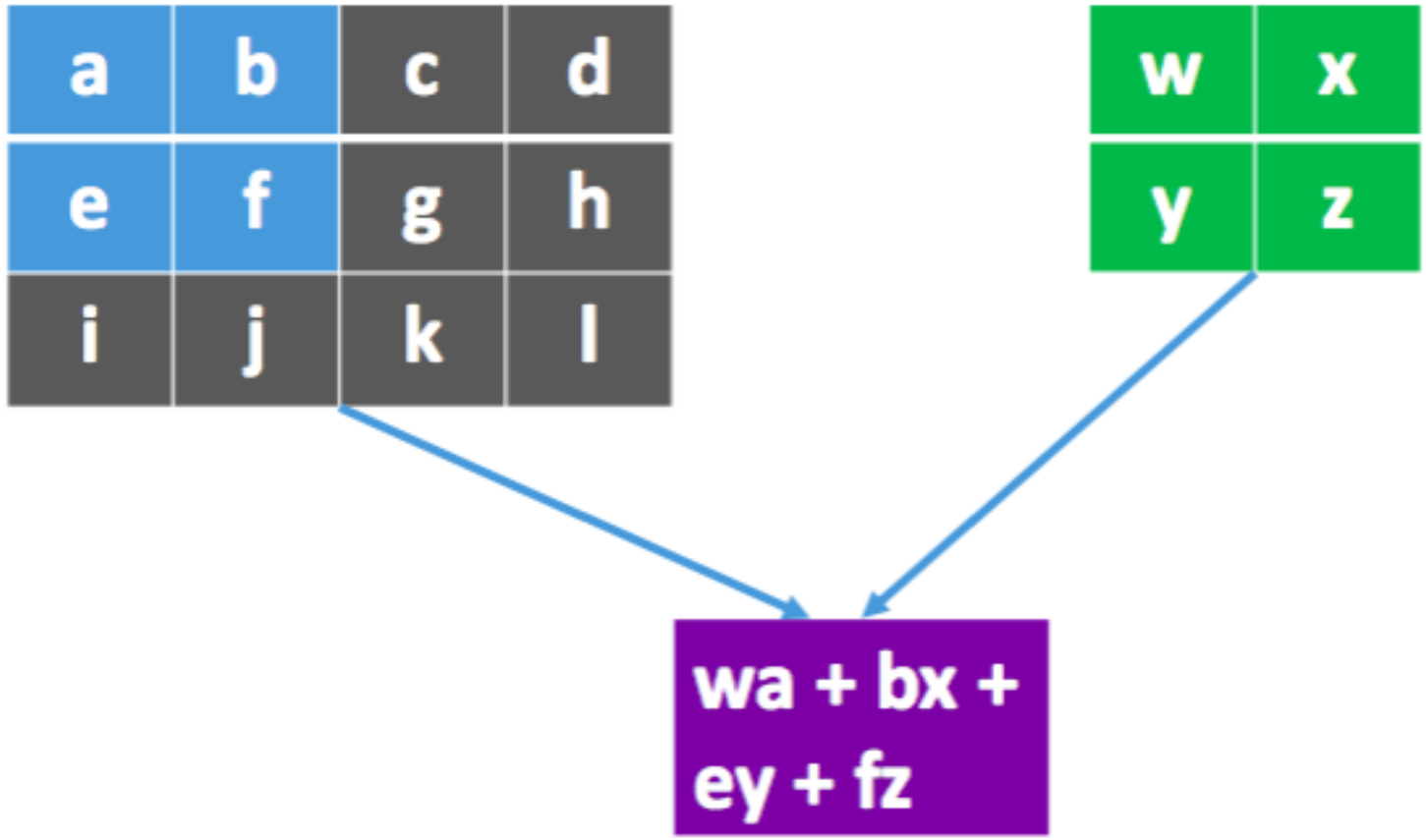


Illustration 2

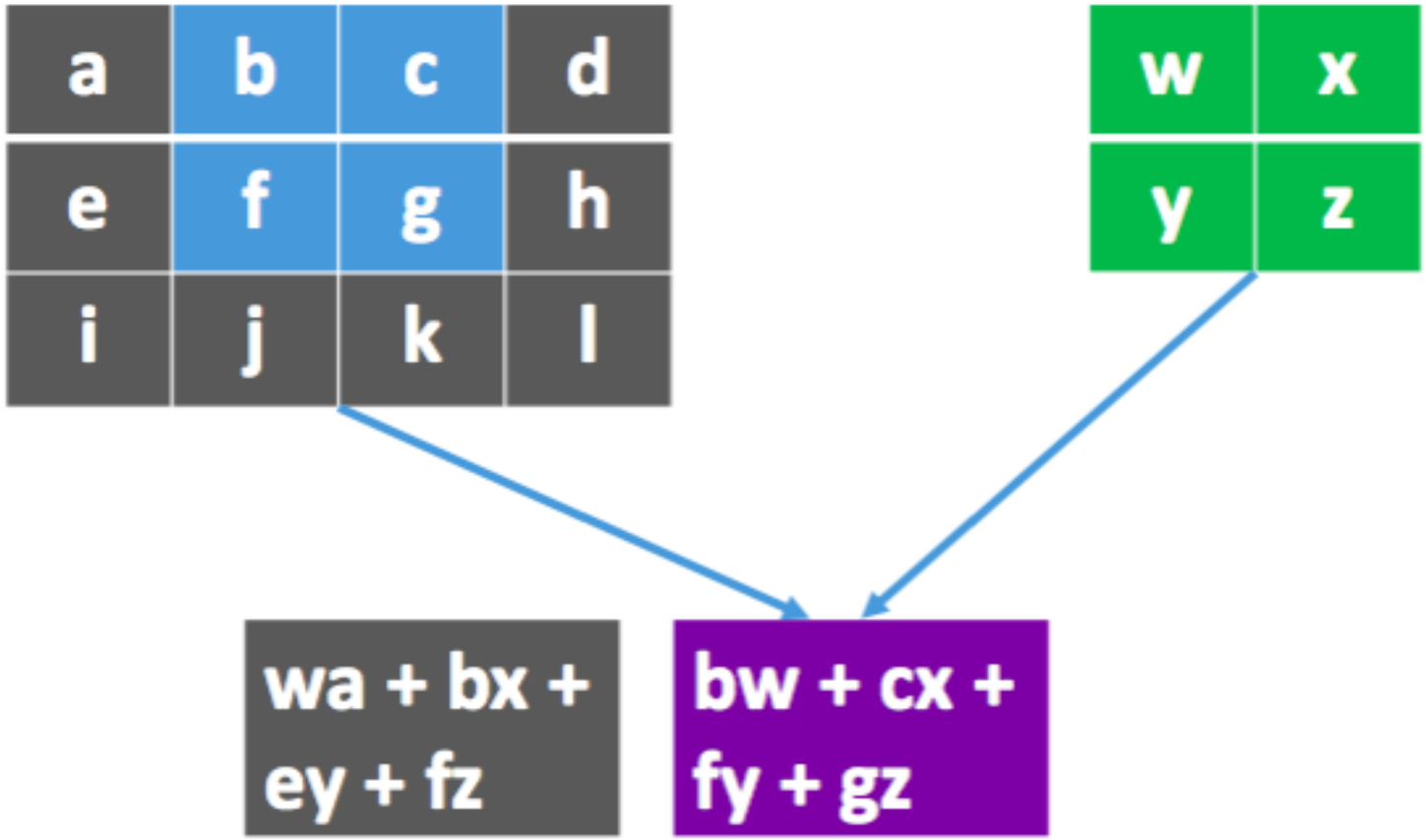
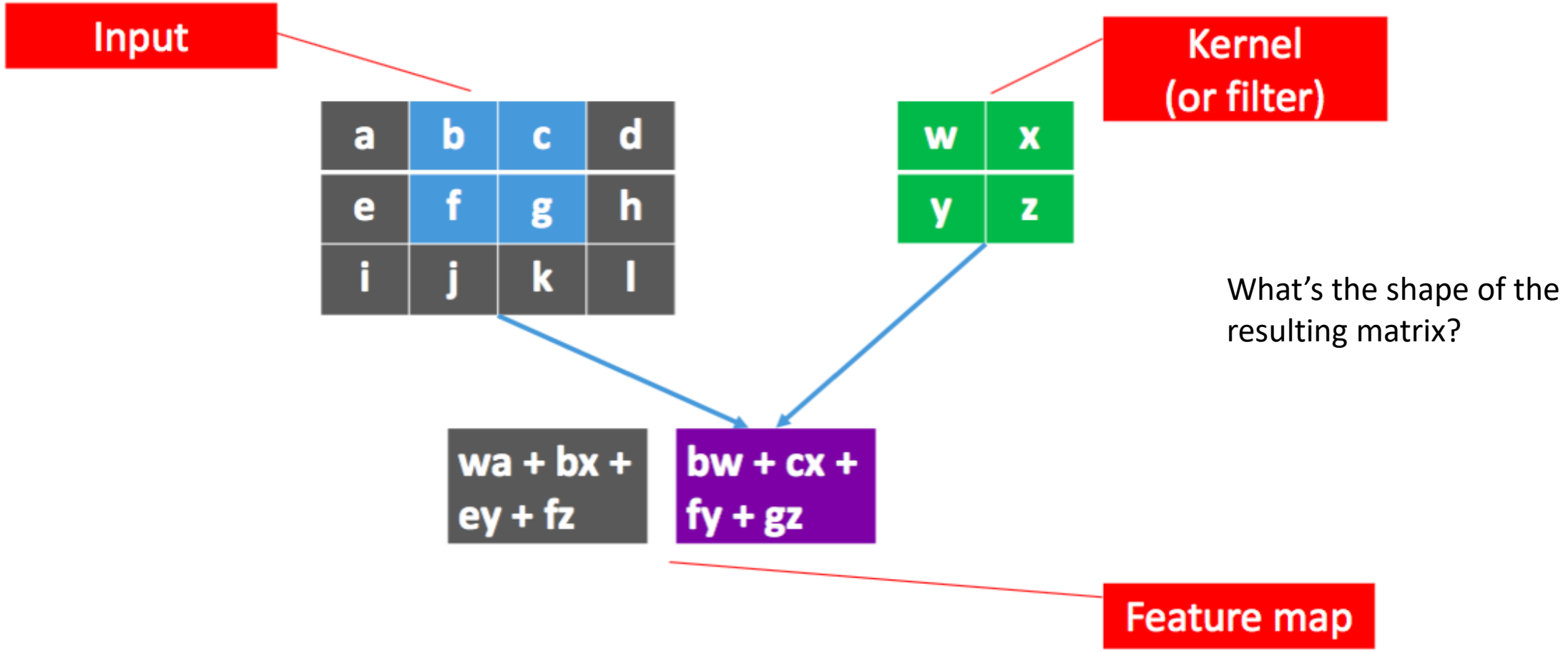


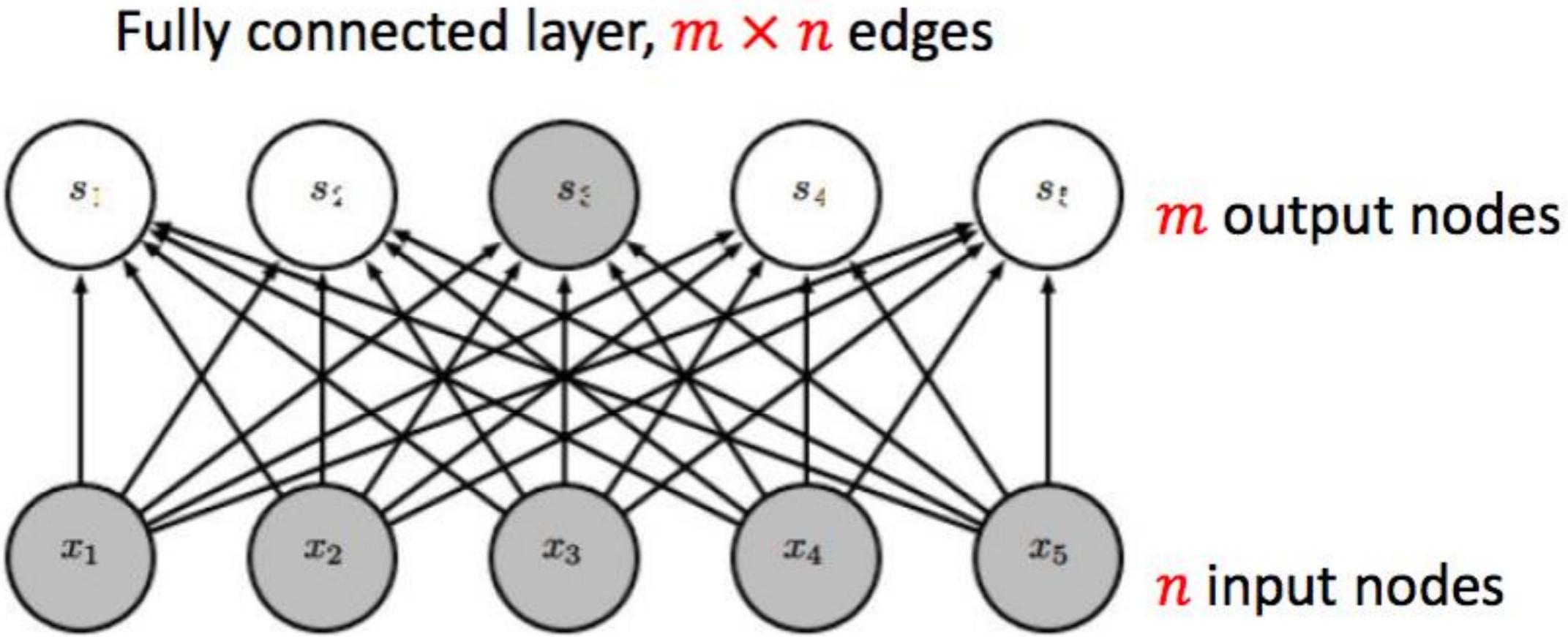
Illustration 2



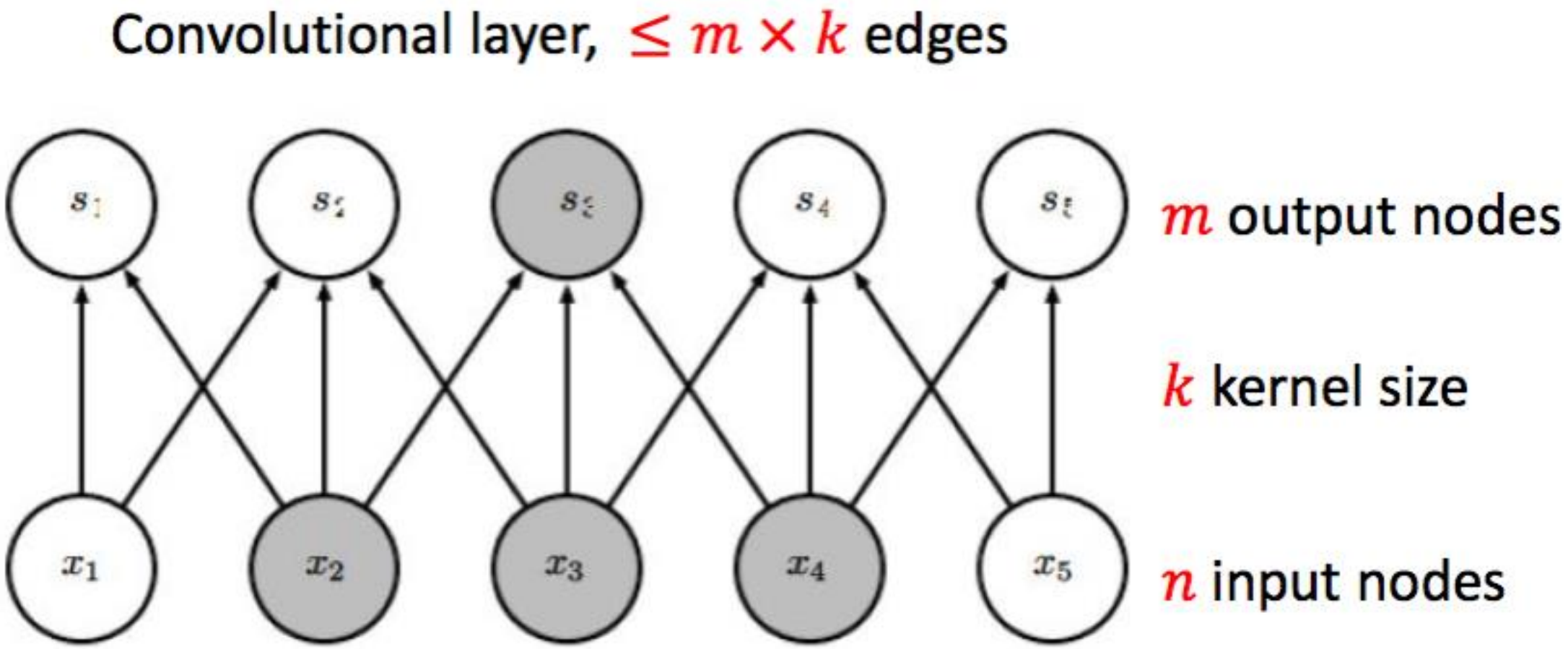
What's the shape of the resulting matrix?

Advantage of Convolutional Layer

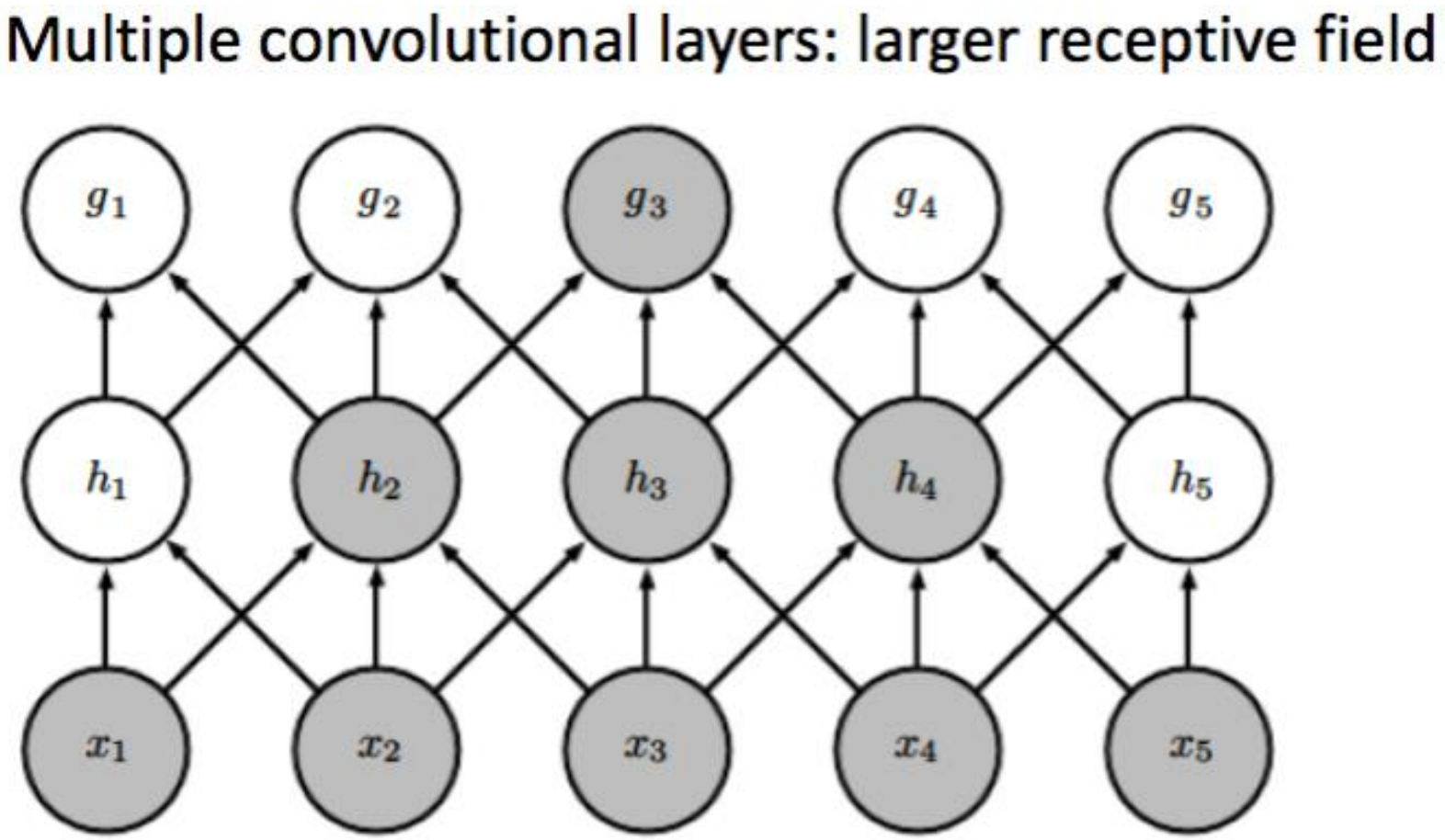
Advantage: sparse interaction



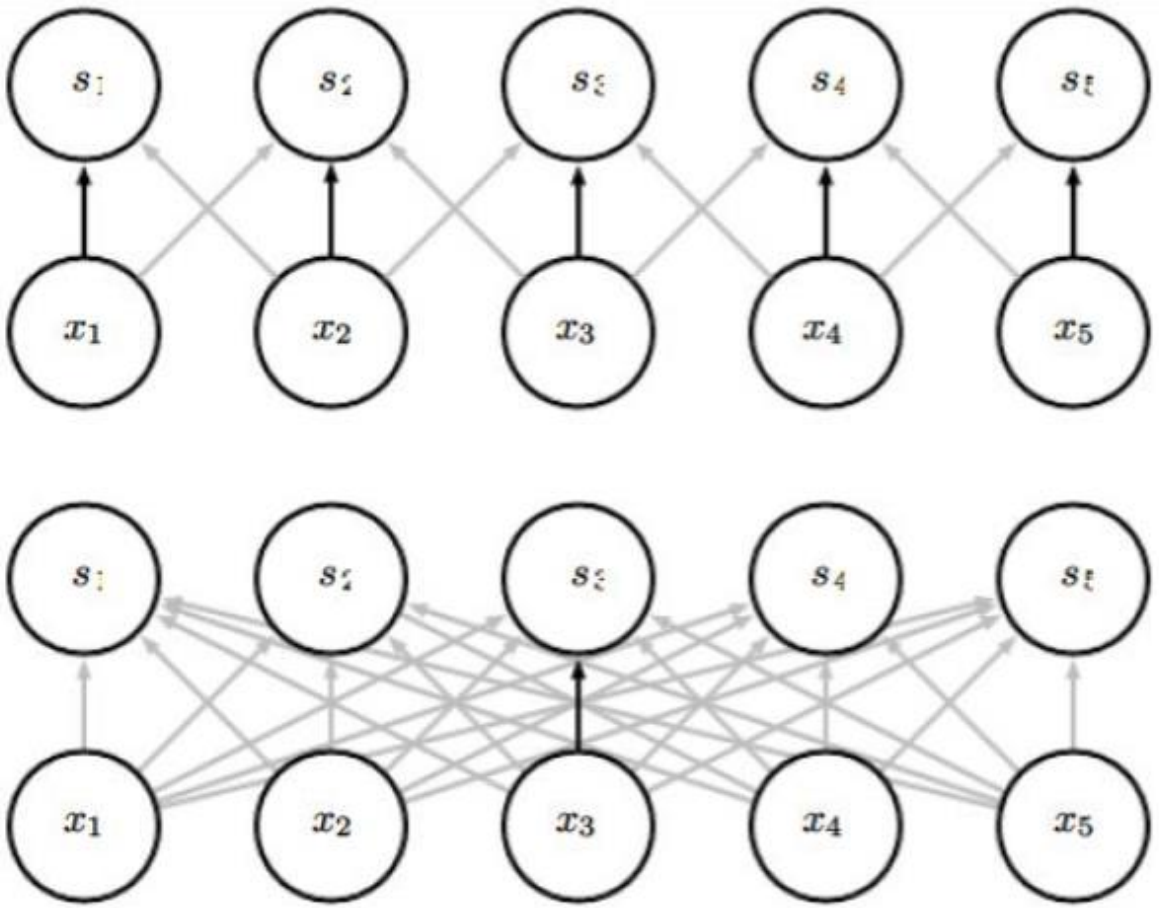
Advantage: sparse interaction



Advantage: sparse interaction



Advantage: parameter sharing/weight tying



The same kernel are used repeatedly. E.g., the black edge is the same weight in the kernel.

Figure from *Deep Learning,* by Goodfellow, Bengio, and Courville

Advantage: equivariant representations

- Equivariant: transforming the input = transforming the output

- Example: input is an image, transformation is shifting
- Convolution(shift(input)) = shift(Convolution(input))

- Useful when care only about the existence of a pattern, rather than the location

Zero-Padding

Zero-Padding

filter

w	**x**
y	z

Input

a	b	c	d
e	f	g	h
i	j	k	l

0	0	0	0	0	0
0	a	b	c	d	0
0	e	f	g	h	0
0	i	j	k	l	0
0	0	0	0	0	0

What's the shape of the resulting matrix?

ReLU

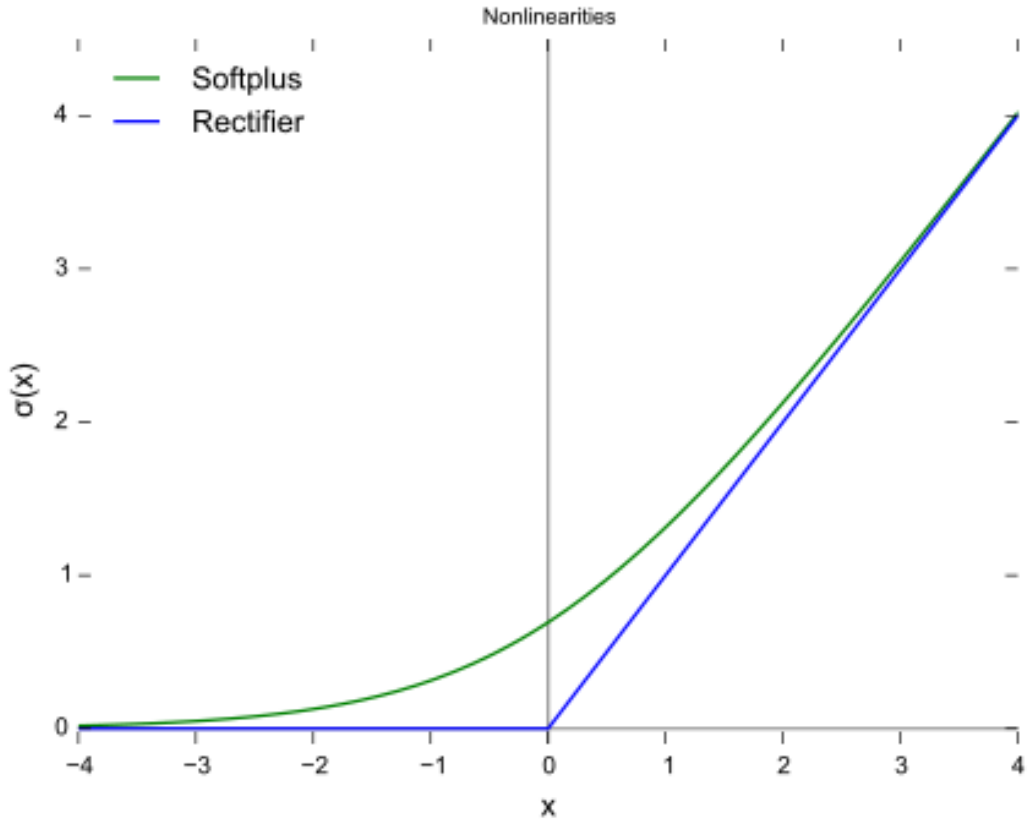
ReLU (**rectified linear unit**)

- **rectifier** is an activation function defined as the positive part of its argument

$$f(x) = \max(0,x)$$

- A smooth approximation to the rectifier is the analytic function

$$f(x) = \log(1+e^x)$$

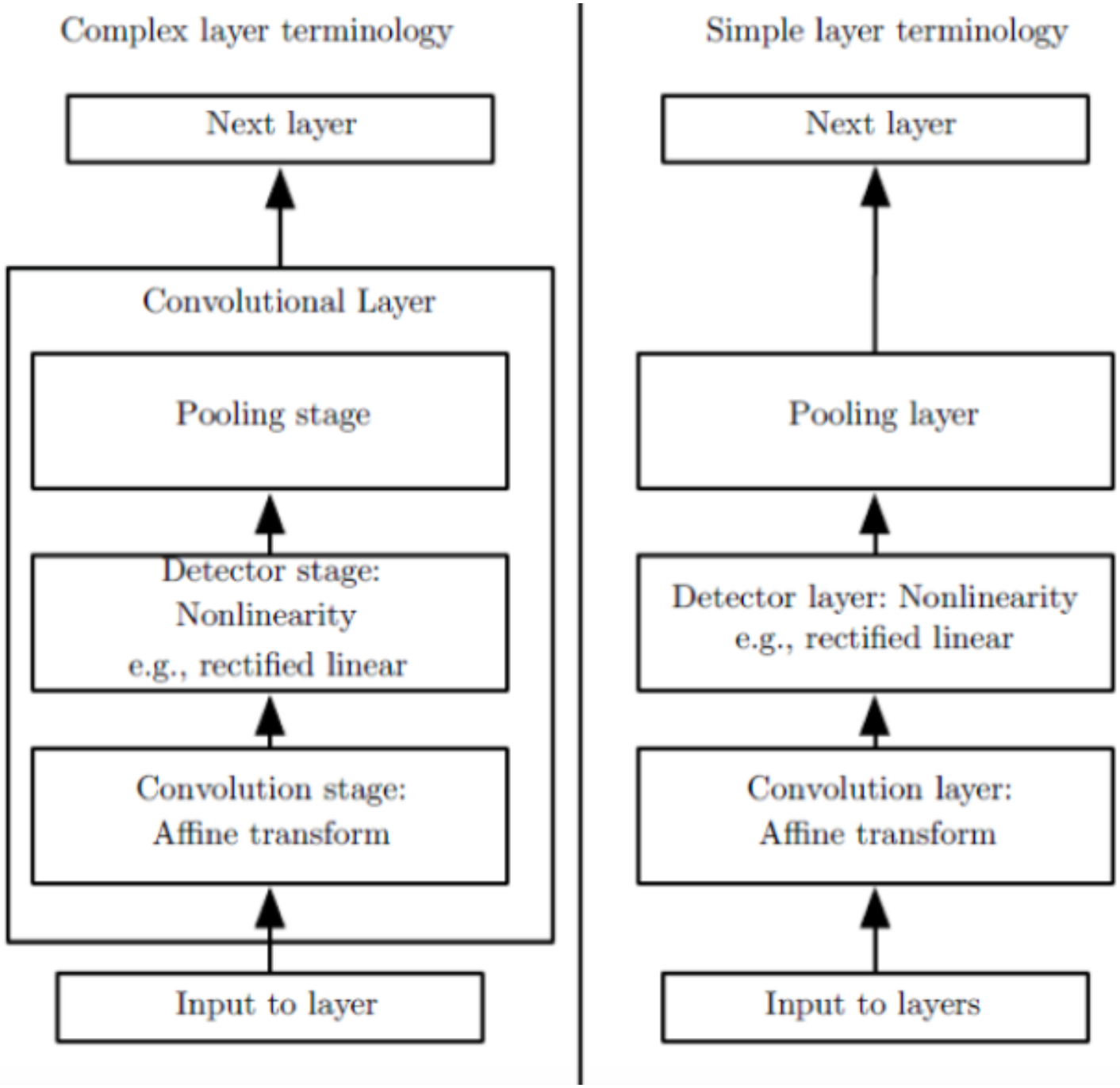


Pooling

Pooling

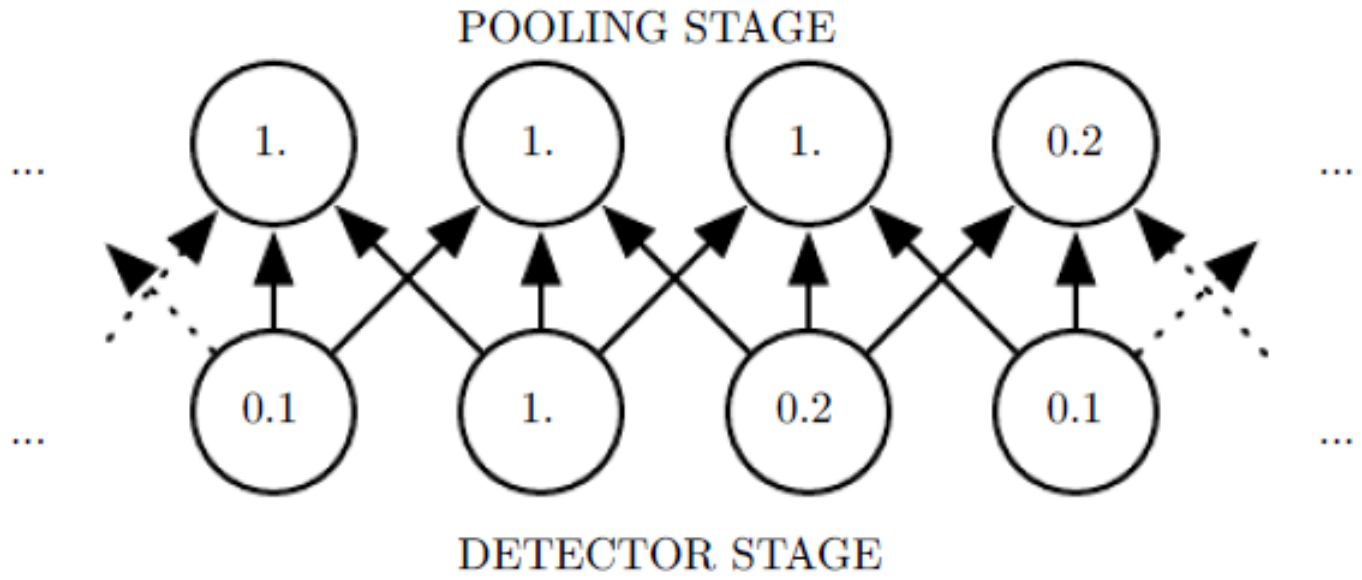
- Pooling layer is frequently used in convulational neural networks with the purpose to progressively reduce the spatial size of the representation to reduce the amount of features and the computational complexity of the network.

Terminology



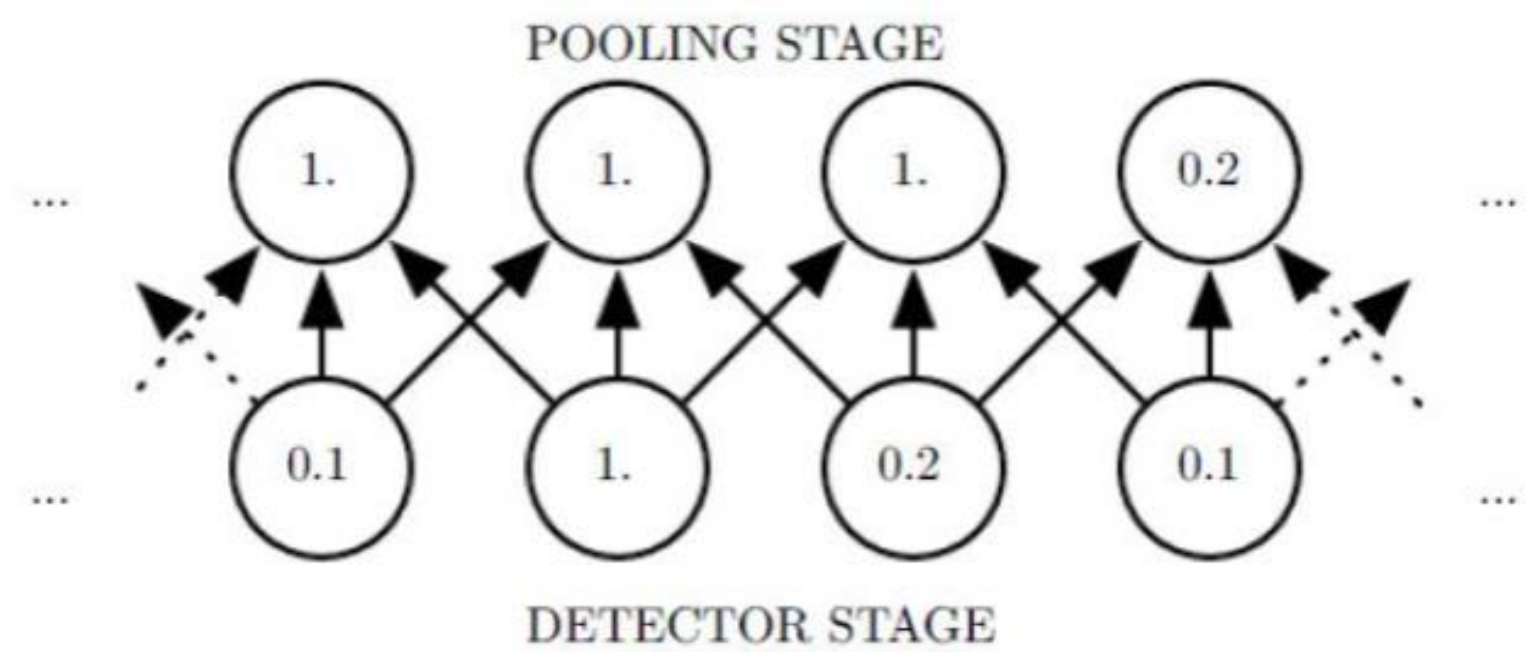
Pooling

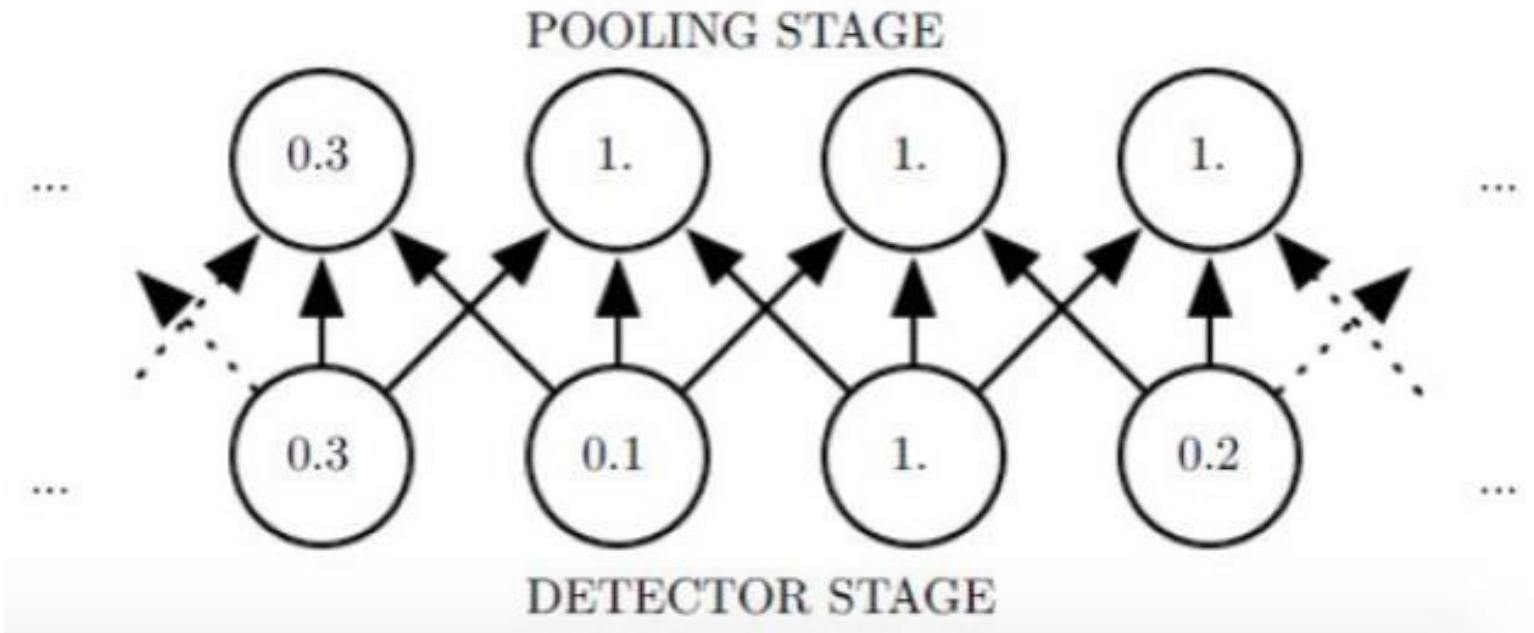
- Summarizing the input (i.e., output the max of the input)



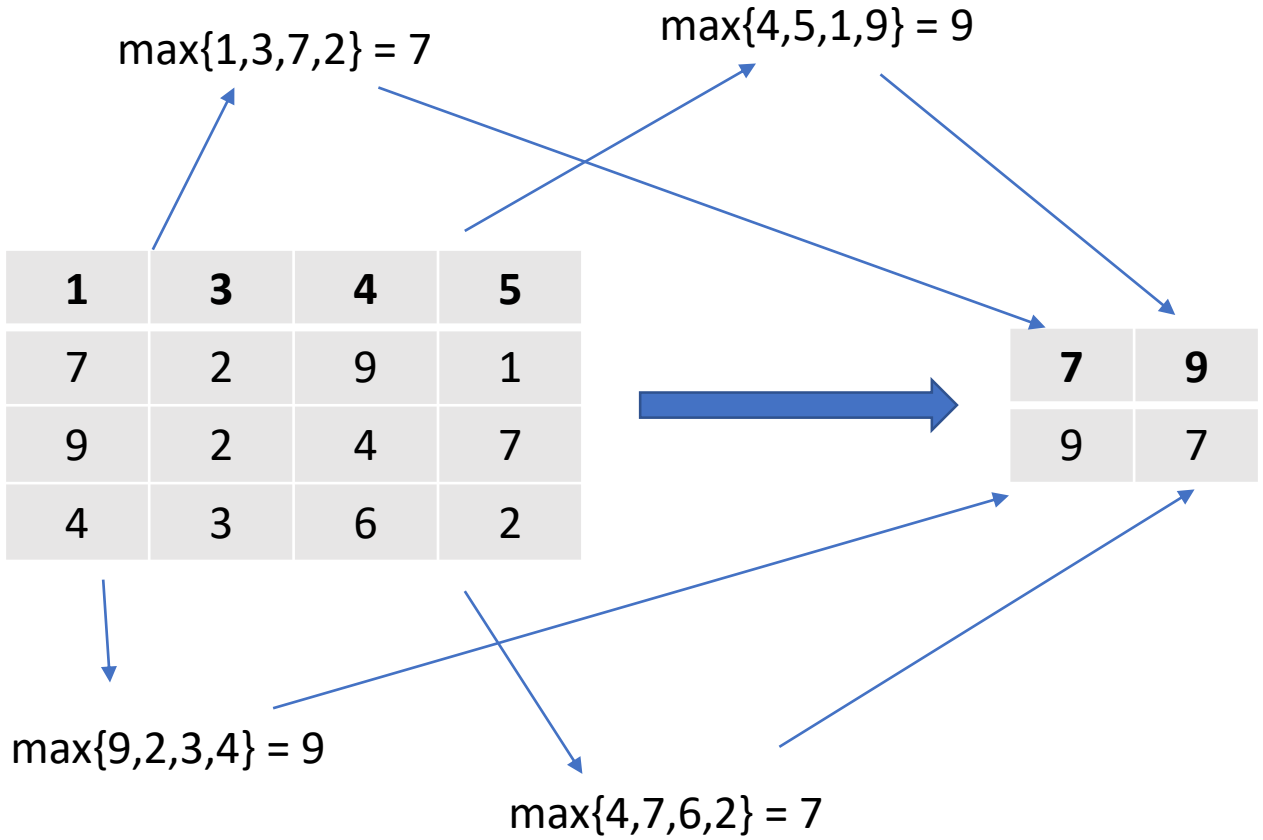
Advantage

- Induce invariance





Example: Max-pooling



Motivation from neuroscience

- David Hubel and Torsten Wiesel studied early visual system in human brain (V1 or primary visual cortex), and won Nobel prize for this

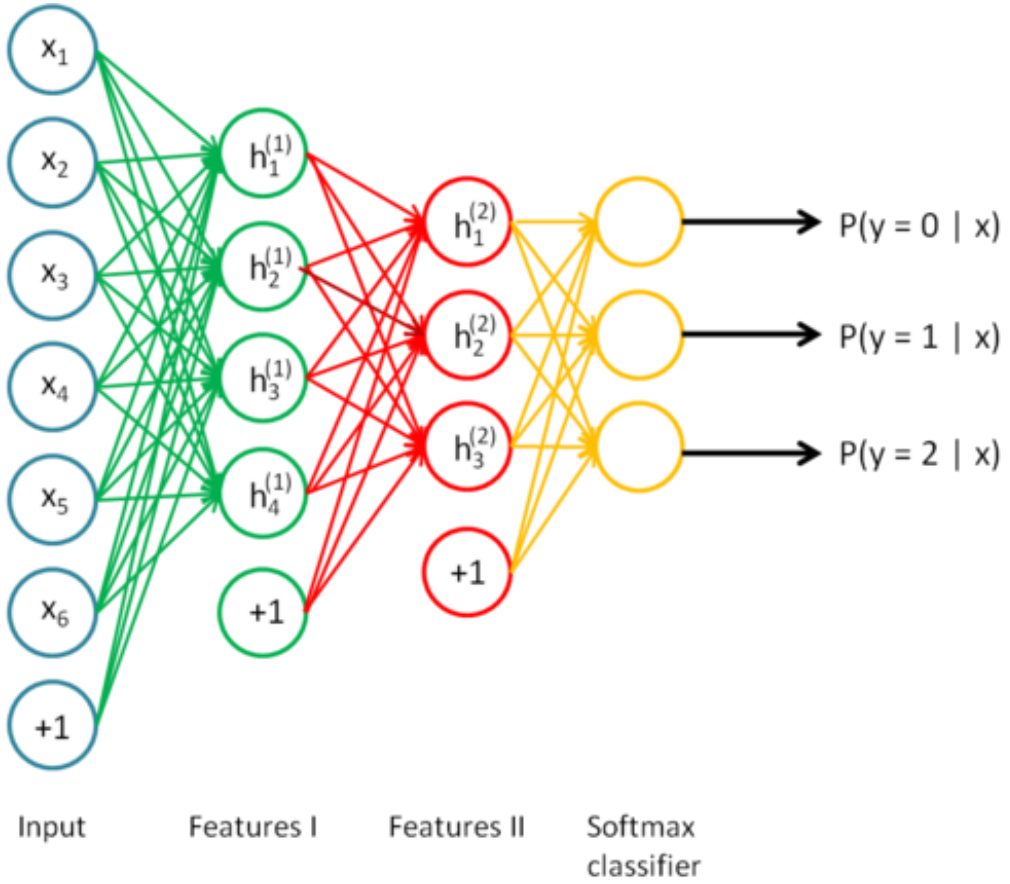
- V1 properties
  - 2D spatial arrangement
  - Simple cells: inspire convolution layers
  - Complex cells: inspire pooling layers

Softmax

Softmax

- Recall that logistic regression produces a decimal between 0 and 1.0. For example, a logistic regression output of 0.8 from an email classifier suggests an 80% chance of an email being spam and a 20% chance of it being not spam. Clearly, the sum of the probabilities of an email being either spam or not spam is 1.0.
- **Softmax** extends this idea into a multi-class world. That is, Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would.

Softmax


x_1
x_2
x_3
x_4
x_5
x_6
+1
$h_1^{(1)}$
$h_2^{(1)}$
$h_3^{(1)}$
$h_4^{(1)}$
+1
$h_1^{(2)}$
$h_2^{(2)}$
$h_3^{(2)}$
+1
P(y = 0 | x)
P(y = 1 | x)
P(y = 2 | x)
Input
Features I
Features II
Softmax classifier

Example: LeNet

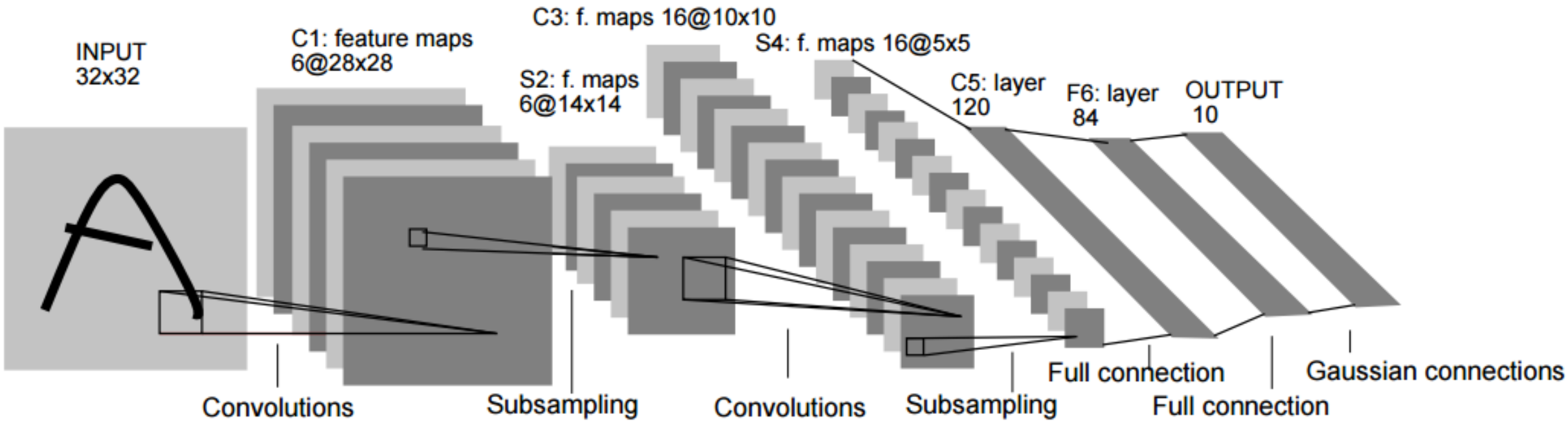
LeNet-5

- Proposed in *"Gradient-based learning applied to document recognition" , by Yann LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner, in Proceedings of the IEEE, 1998*

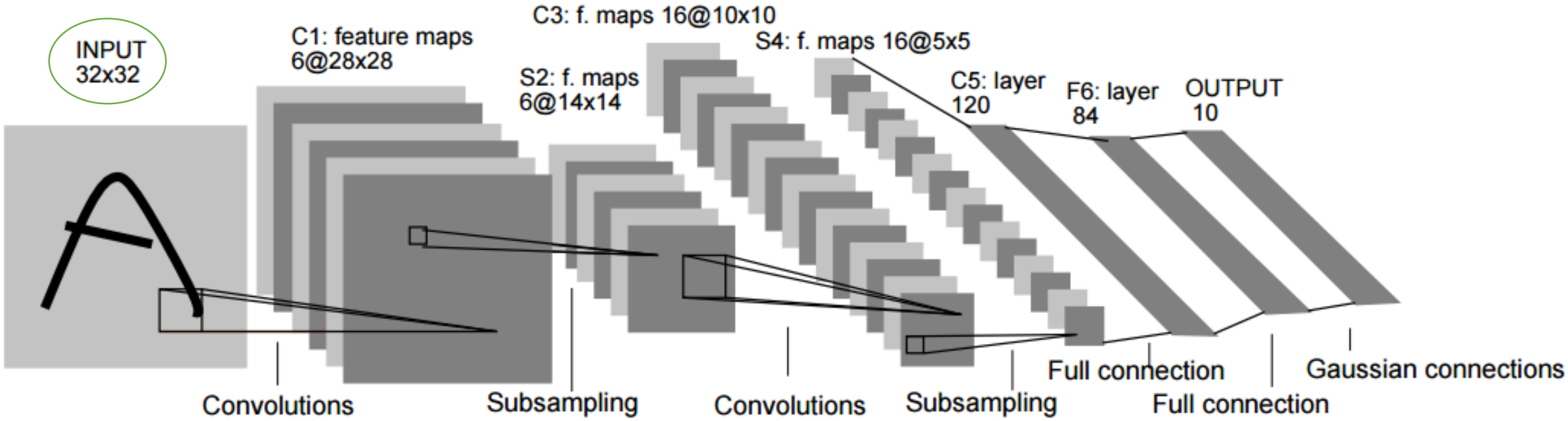
- Apply convolution on 2D images (MNIST) and use backpropagation

- Structure: 2 convolutional layers (with pooling) + 3 fully connected layers
  - Input size: 32x32x1
  - Convolution kernel size: 5x5
  - Pooling: 2x2

LeNet-5

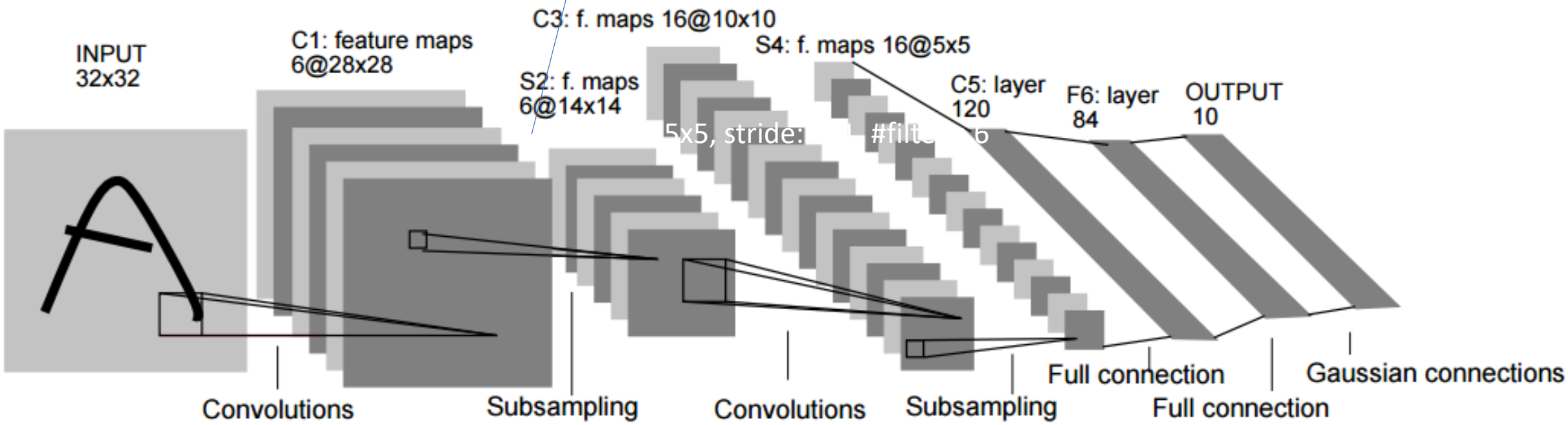

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

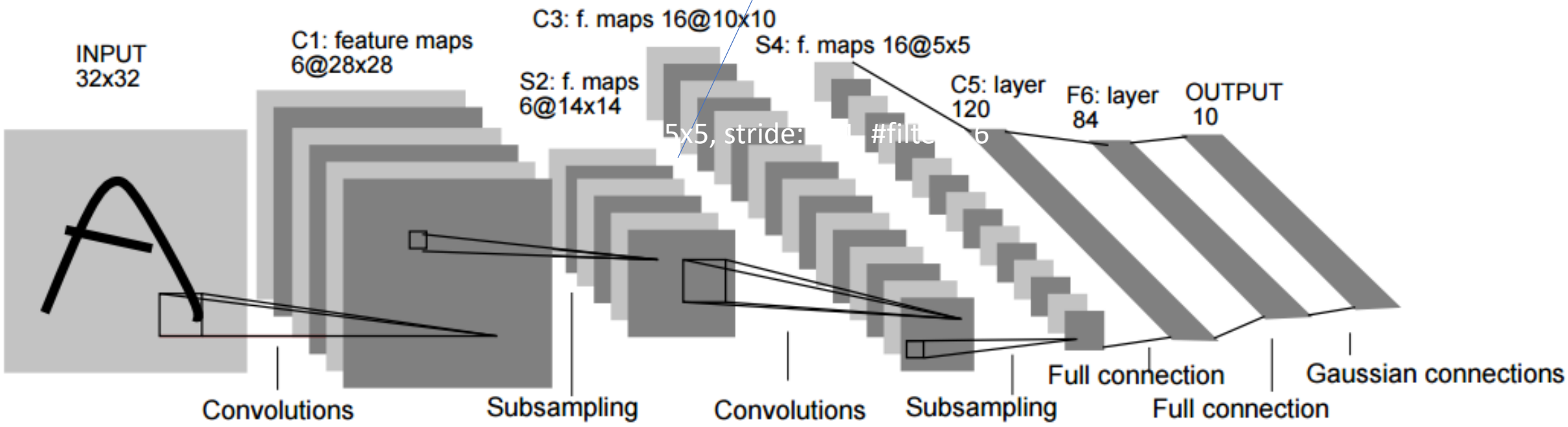
LeNet-5

Pooling: 2x2, stride: 2

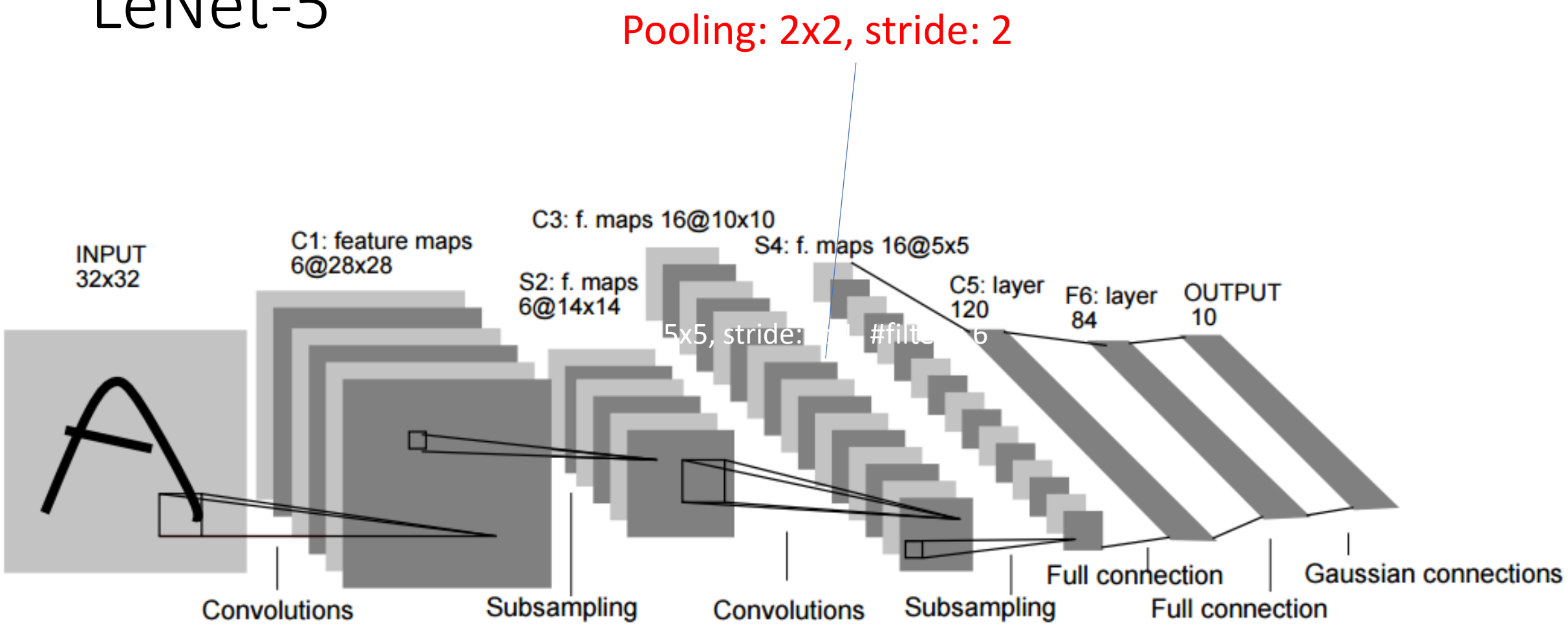


LeNet-5

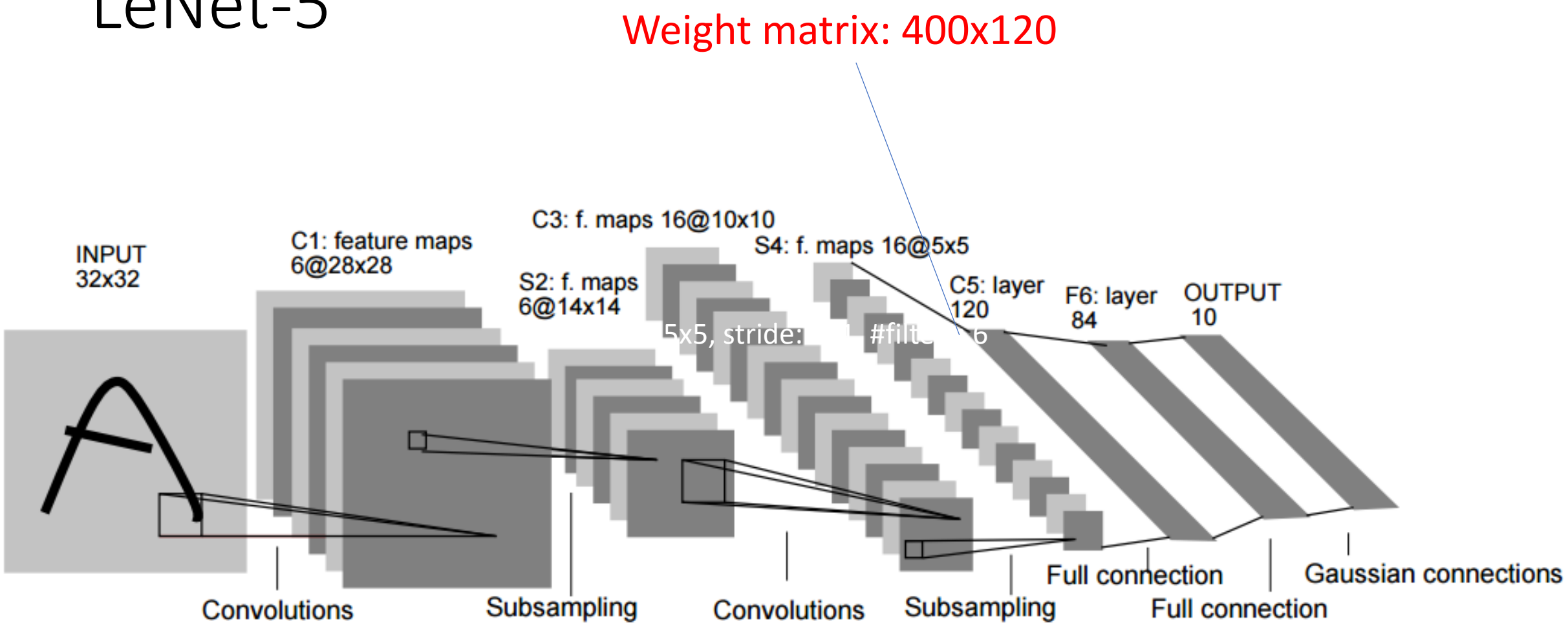
Filter: 5x5x6, stride: 1x1,
#filters: 16



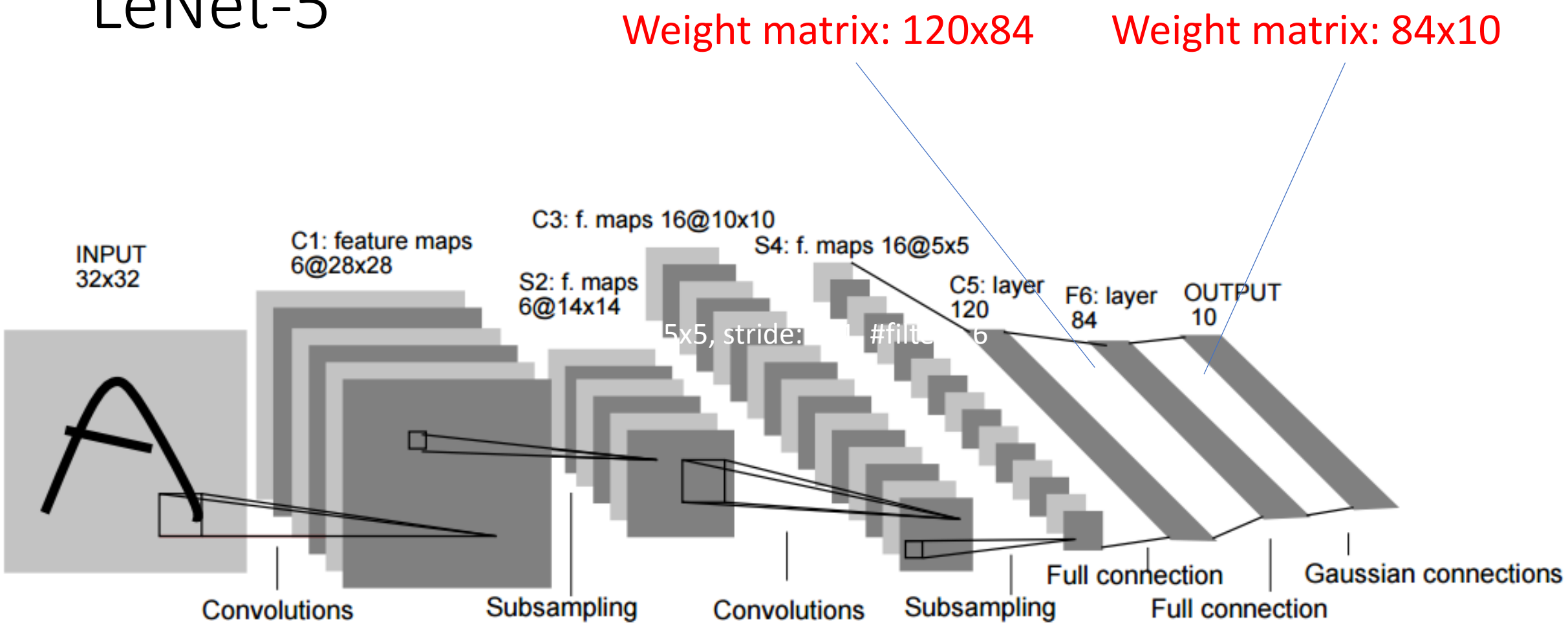
LeNet-5


Pooling: 2x2, stride: 2
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

LeNet-5


Weight matrix: 400x120
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
5x5, stride:
#filt
6
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

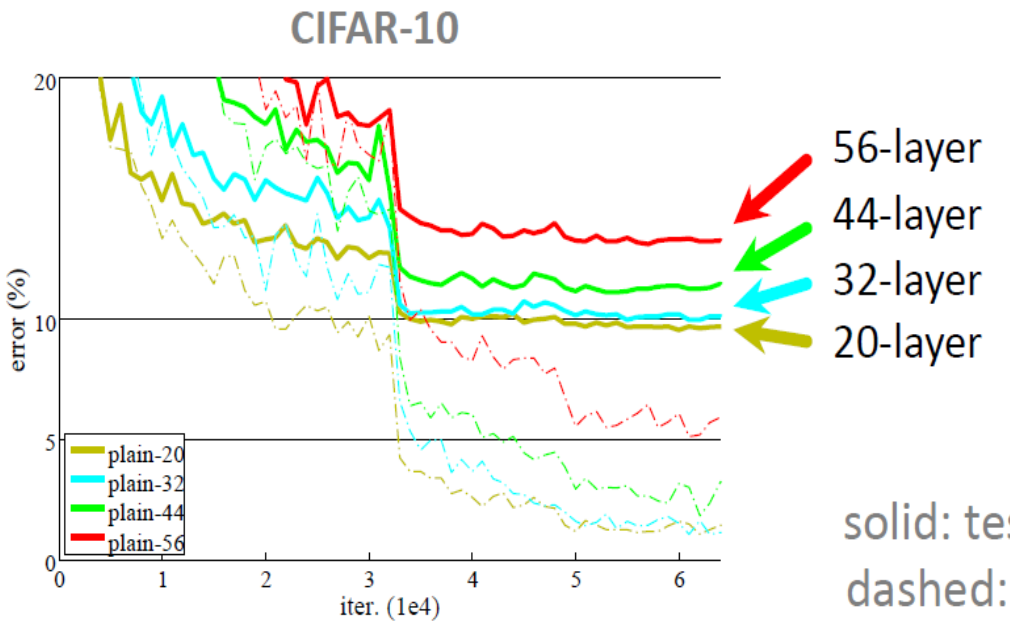
LeNet-5

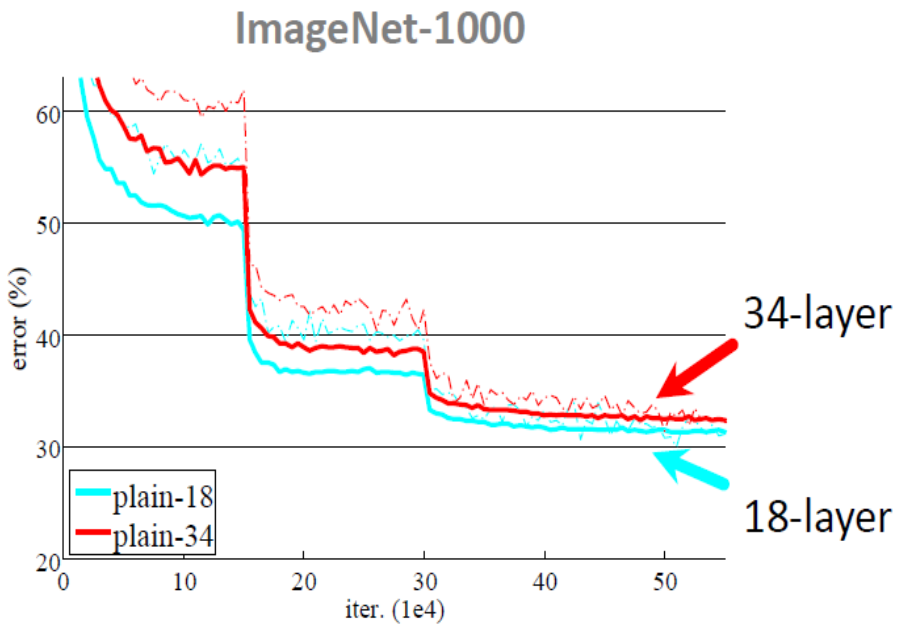

Weight matrix: 120x84
Weight matrix: 84x10
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
5x5, stride:
#filt
6
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

Example: ResNet

Plain Network

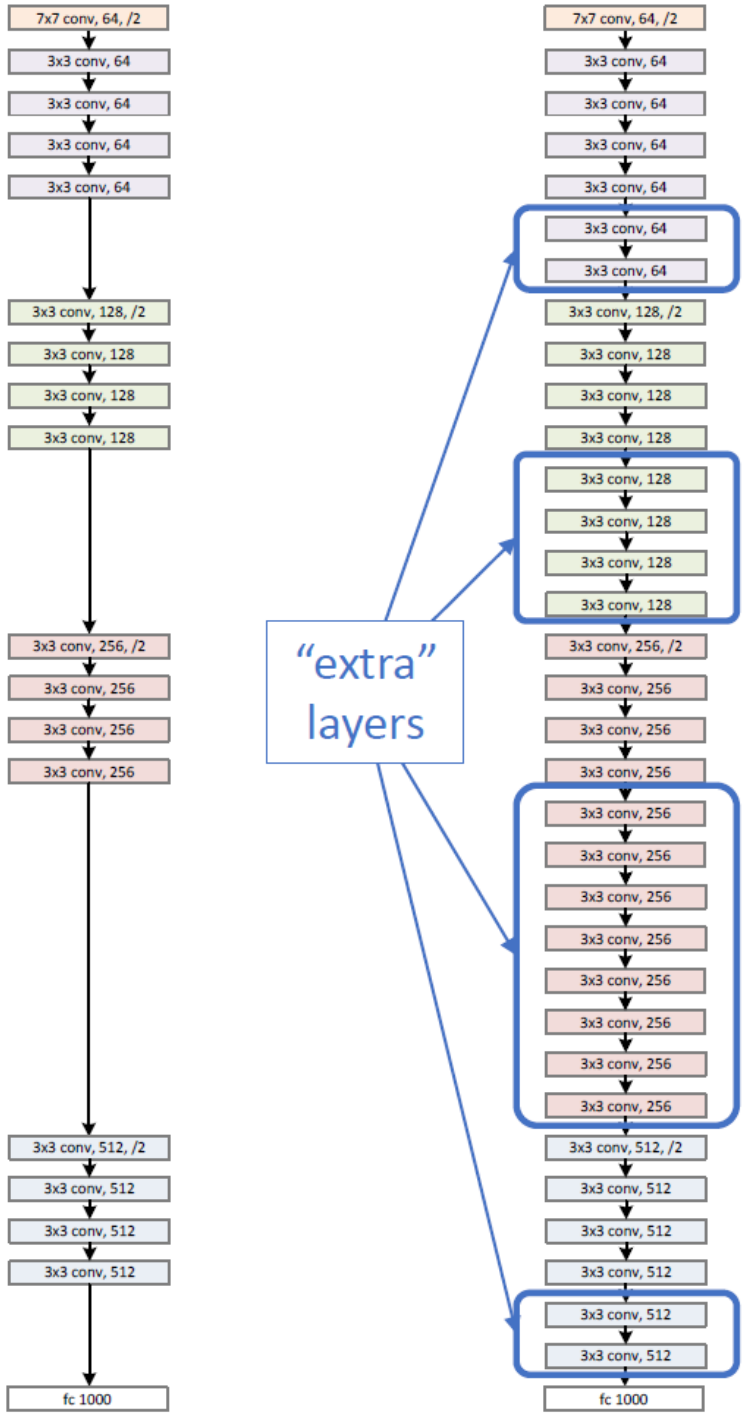
- “Overly deep” plain nets have higher training error
- A general phenomenon, observed in many datasets





Residual Network

- Naïve solution
  - If extra layers are an identity mapping, then a training errors does not increase

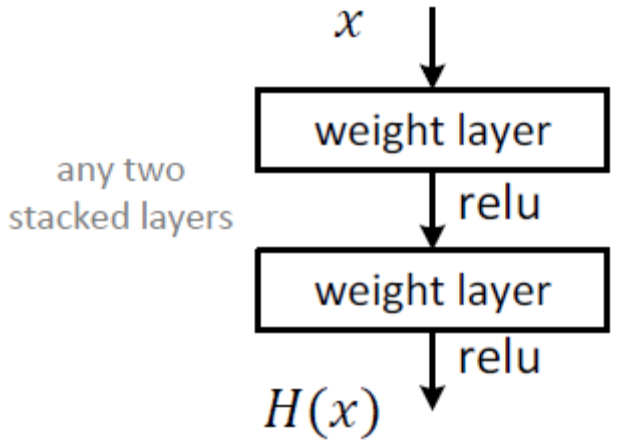


Residual Network

- Deeper networks also maintain the tendency of results
  - Features in same level will be almost same
  - An amount of changes is fixed
  - Adding layers makes smaller differences
  - Optimal mappings are closer to an identity

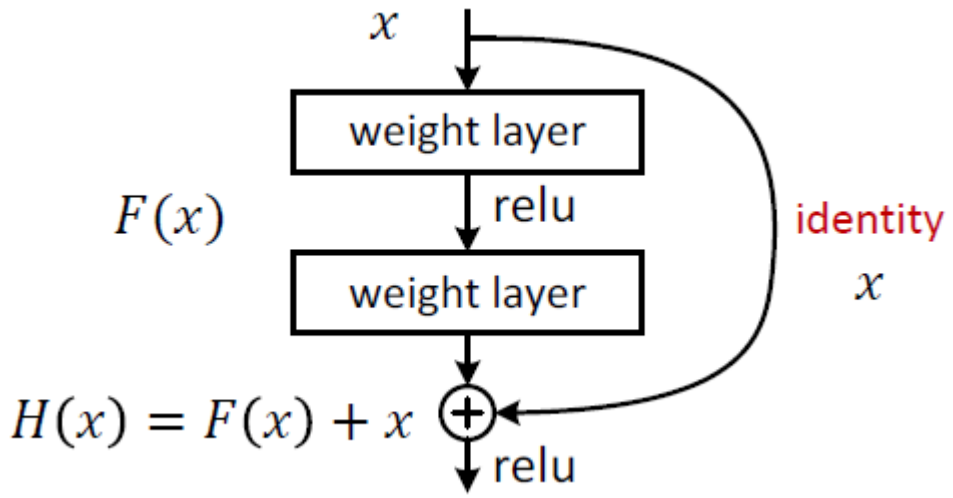
Residual Network

- Plain block
  - Difficult to make identity mapping because of multiple non-linear layers



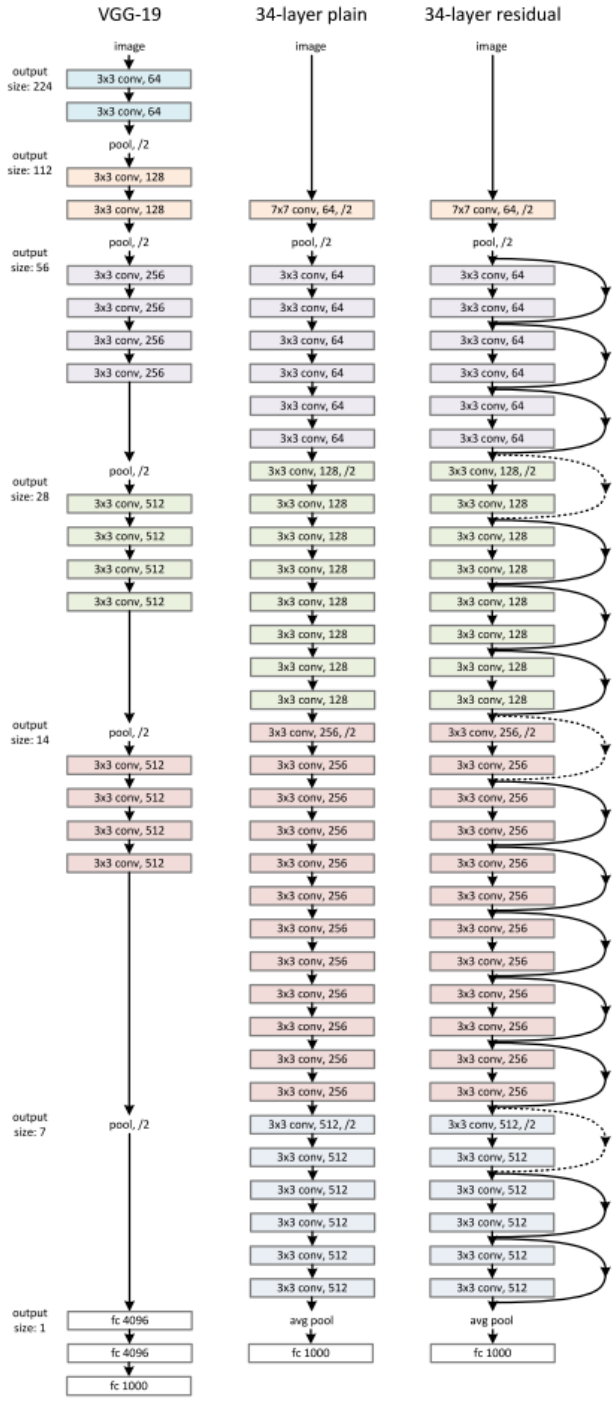
Residual Network

- Residual block
  - If identity were optimal, easy to set weights as 0
  - If optimal mapping is closer to identity, easier to find small fluctuations
- -> Appropriate for treating perturbation as keeping a base information



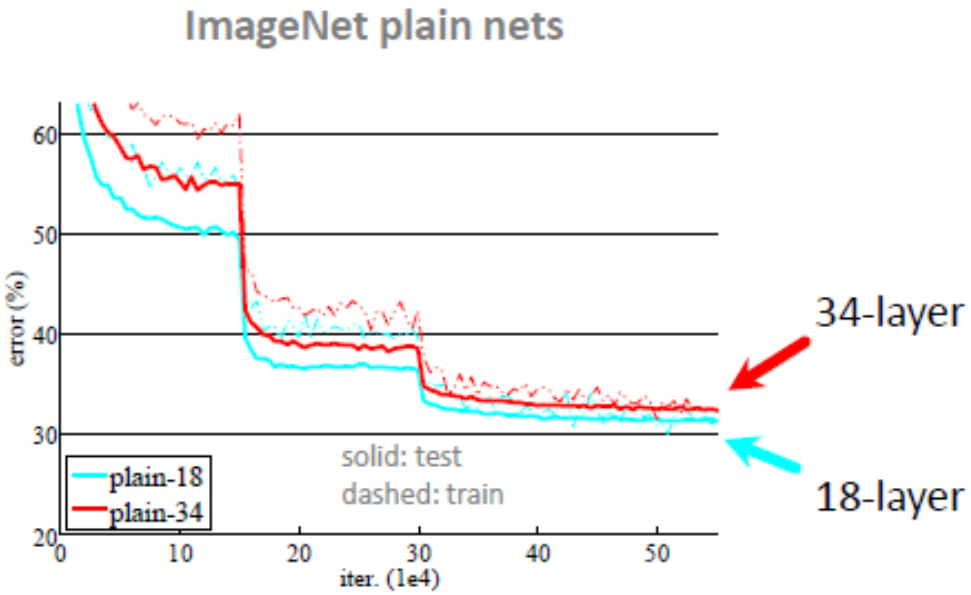
Network Design

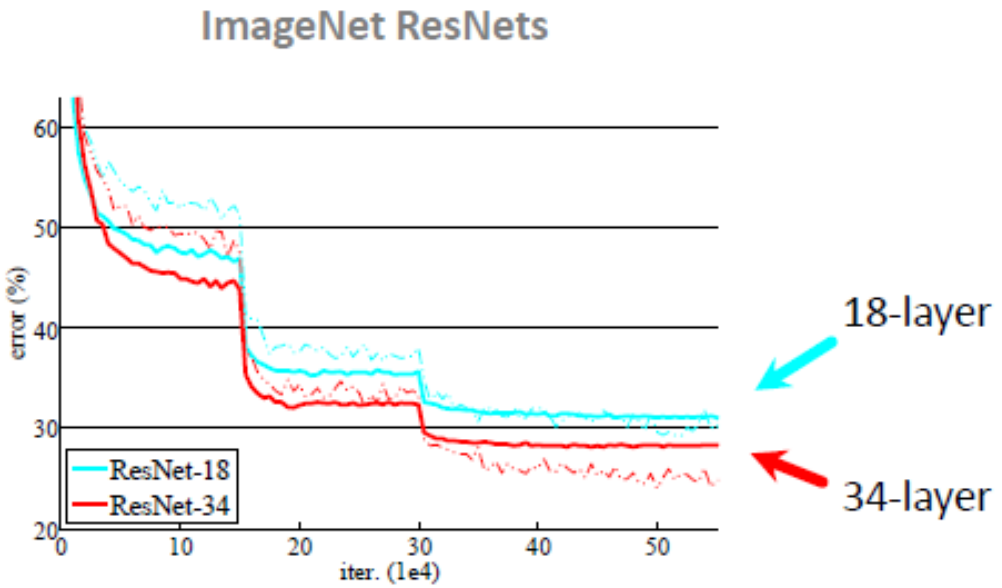
- Basic design (VGG-style)
  - All 3x3 conv (almost)
  - Spatial size/2 => #filters x2
  - Batch normalization
  - Simple design, just deep
- Other remarks
  - No max pooling (almost)
  - No hidden fc
  - No dropout



Results

- Deep Resnets can be trained without difficulties
- Deeper ResNets have lower training error, and also lower test error



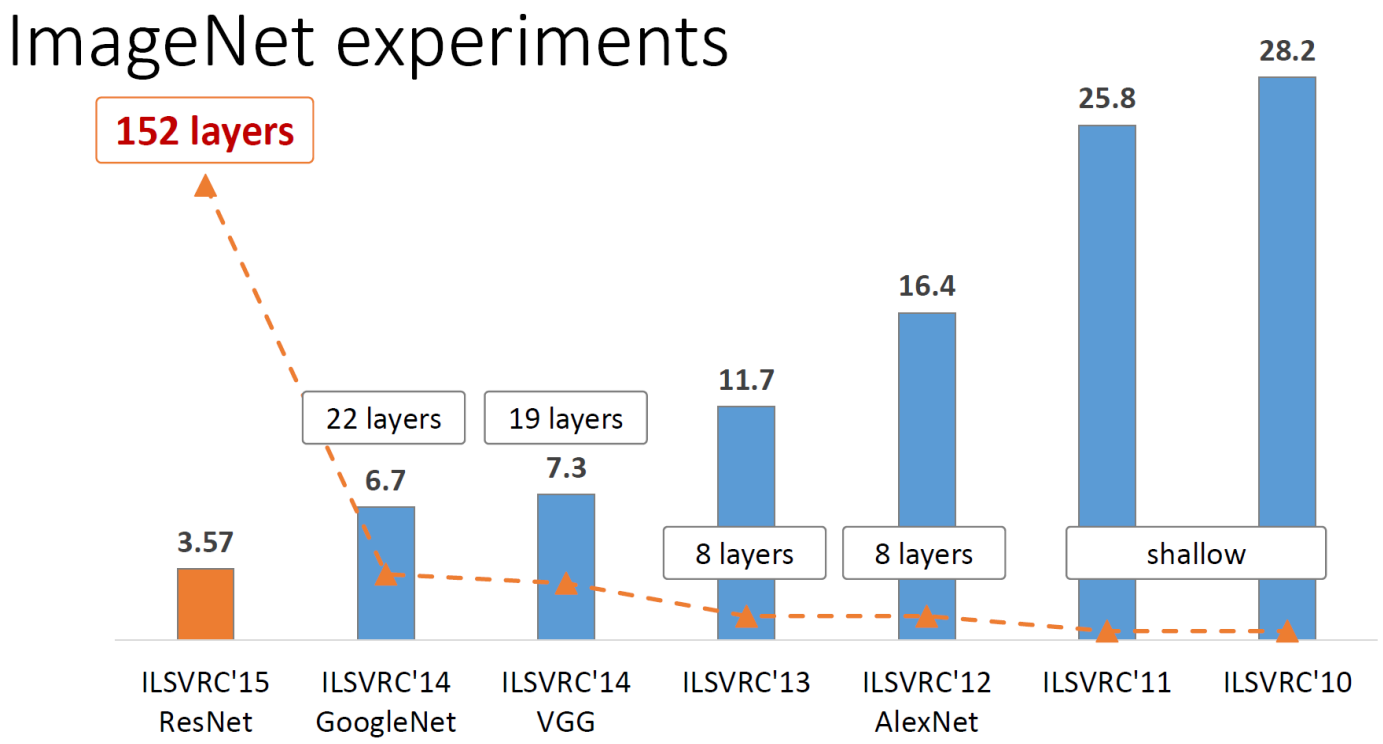


Results

- 1st places in all five main tracks in “ILSVRC & COCO 2015 Competitions”
  - ImageNet Classification
  - ImageNet Detection
  - ImageNet Localization
  - COCO Detection
  - COCO Segmentation

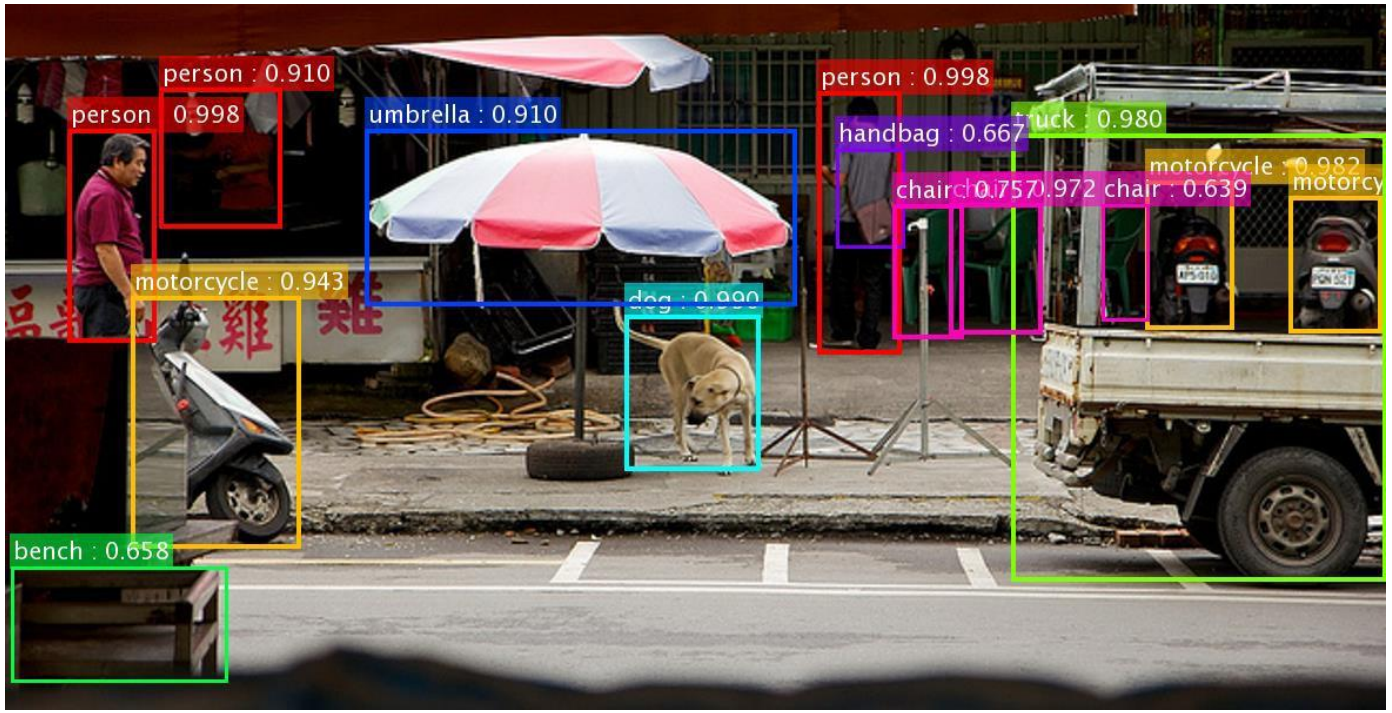
Quantitative Results

- ImageNet Classification



Qualitative Result

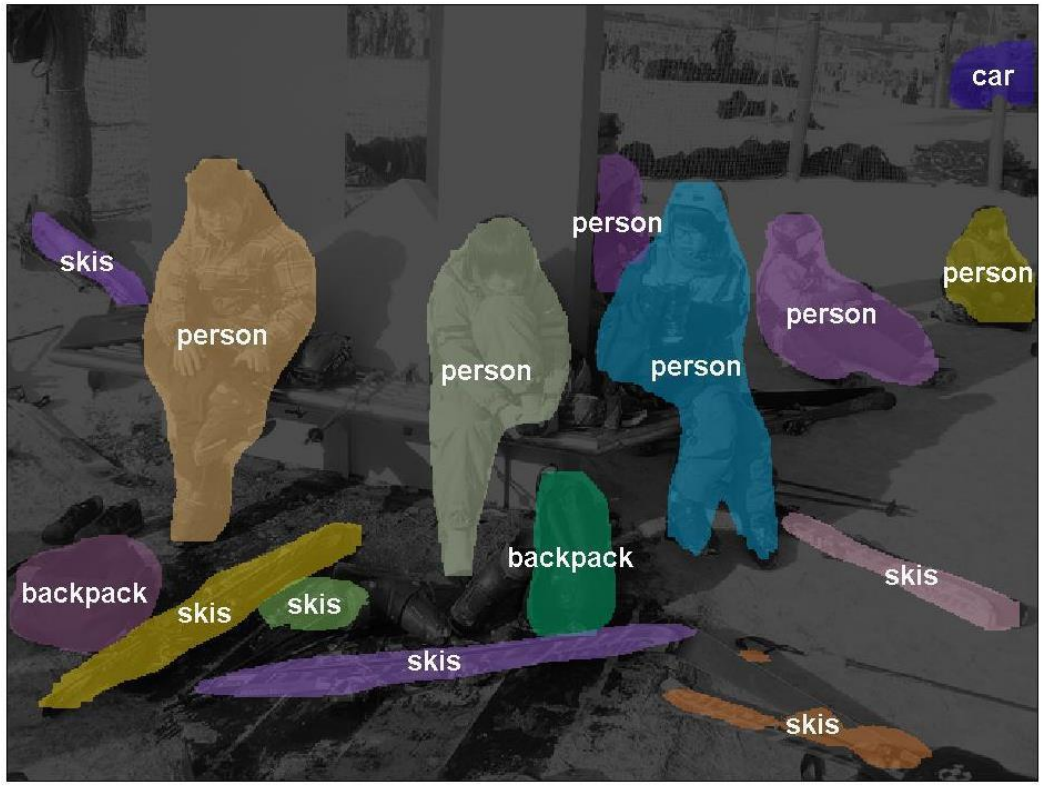
- Object detection
  - Faster R-CNN + ResNet



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.
Jifeng Dai, Kaiming He, & Jian Sun. “Instance-aware Semantic Segmentation via Multi-task Network Cascades”. arXiv 2015.

Qualitative Results

- Instance Segmentation



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.