# Deep Learning Basics

## Convolutional Neural Networks

Francis Steen and Xinyu You
Red Hen Lab
August 2019

# Content

- A Brief History of CNNs
- Why we need CNNs?
- The Structure of CNNs
  - Convolution
    - Convolution operation
    - Padding
    - Stride
  - Pooling
- Some typical CNNs
- Example: Dog or Cat?

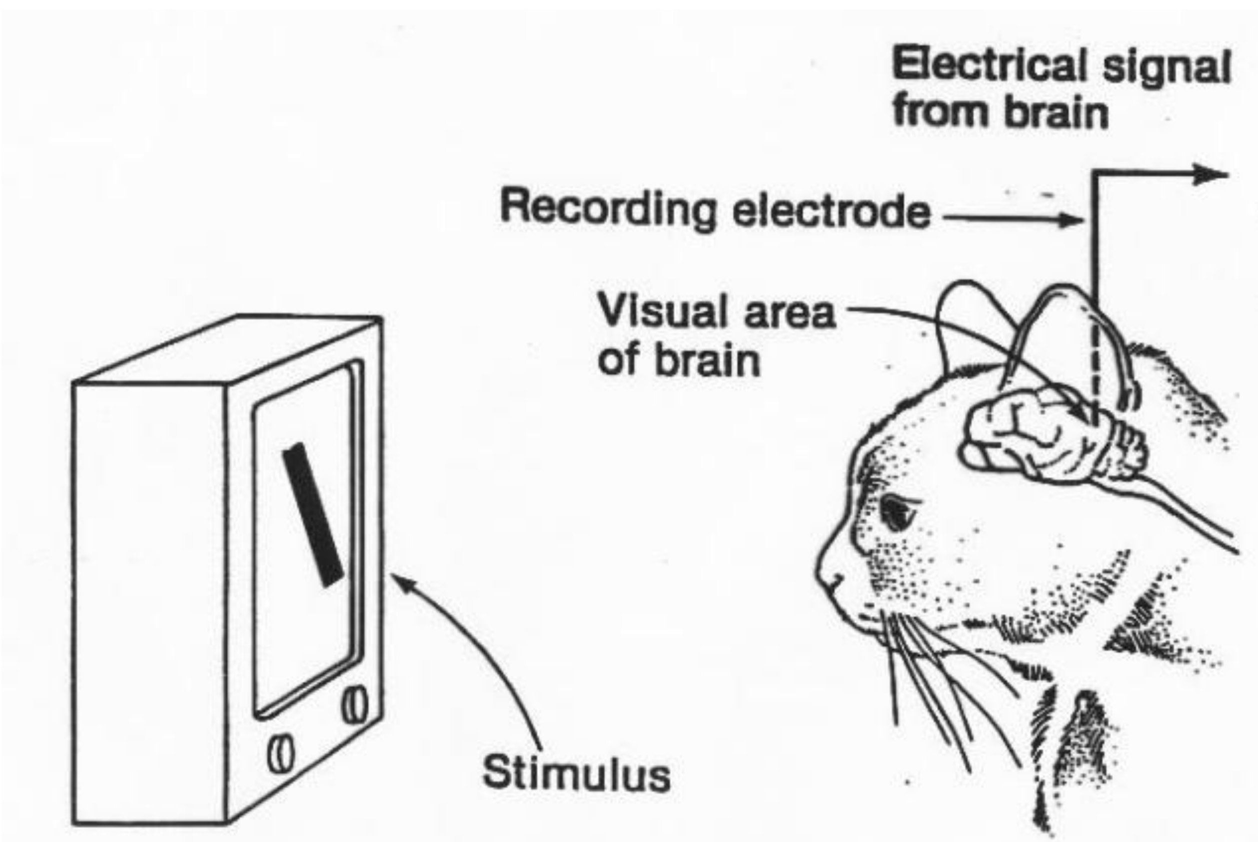
# A. A Brief History of CNNs

# Convolutional Neural Networks

- A convolutional neural network (CNN, or ConvNet) is a class of Feedforward Neural Network.

- It is put forward by the influence of biological Receptive Field mechanism.

# Receptive fields




- Work by Hubel and Wiesel in the 1950s and 1960s showed that cat and monkey visual cortices contain neurons that individually respond to small regions of the visual field.
- The receptive field is a portion of the sensory cortex that elicit neuronal responses when stimulated.

# First strong results

Acoustic Modeling using Deep Belief Networks, Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010

Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition, George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

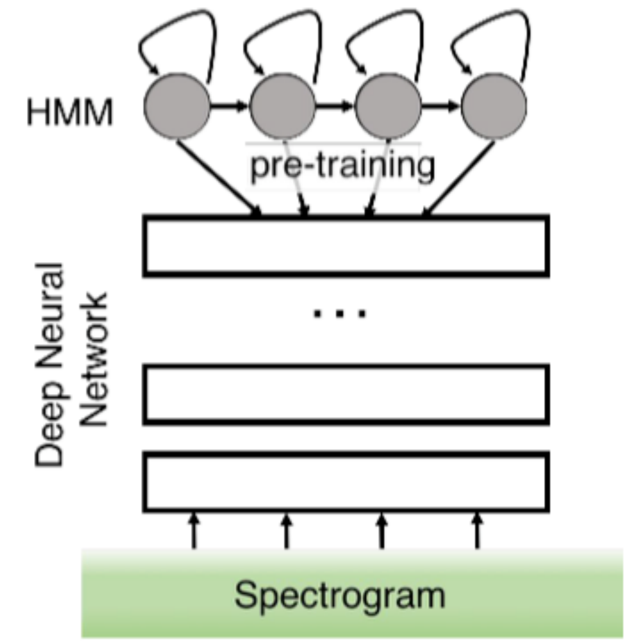
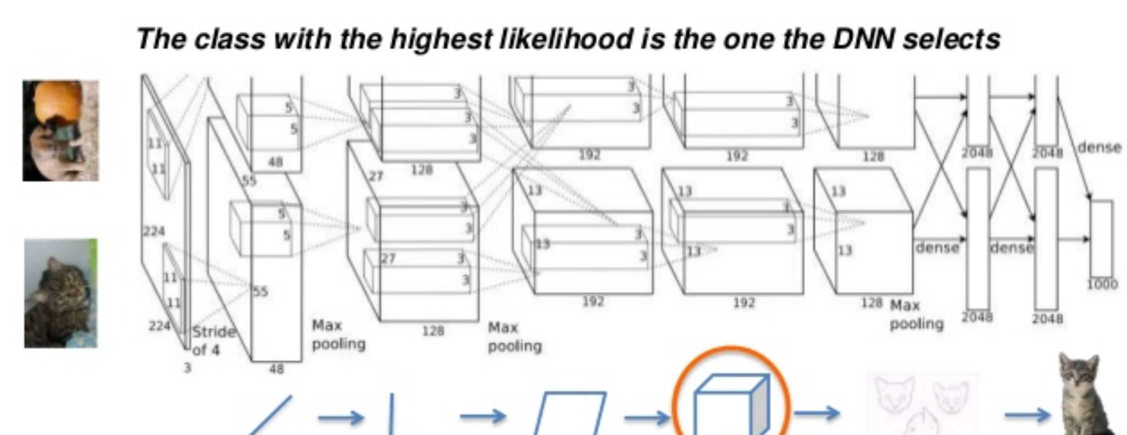



Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

ImageNet classification with deep convolutional neural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

AlexNet (Krizhevsky et al. 2012)




When AlexNet is processing an image, this is what is happening at each layer.

# B. Why do we need CNNs?

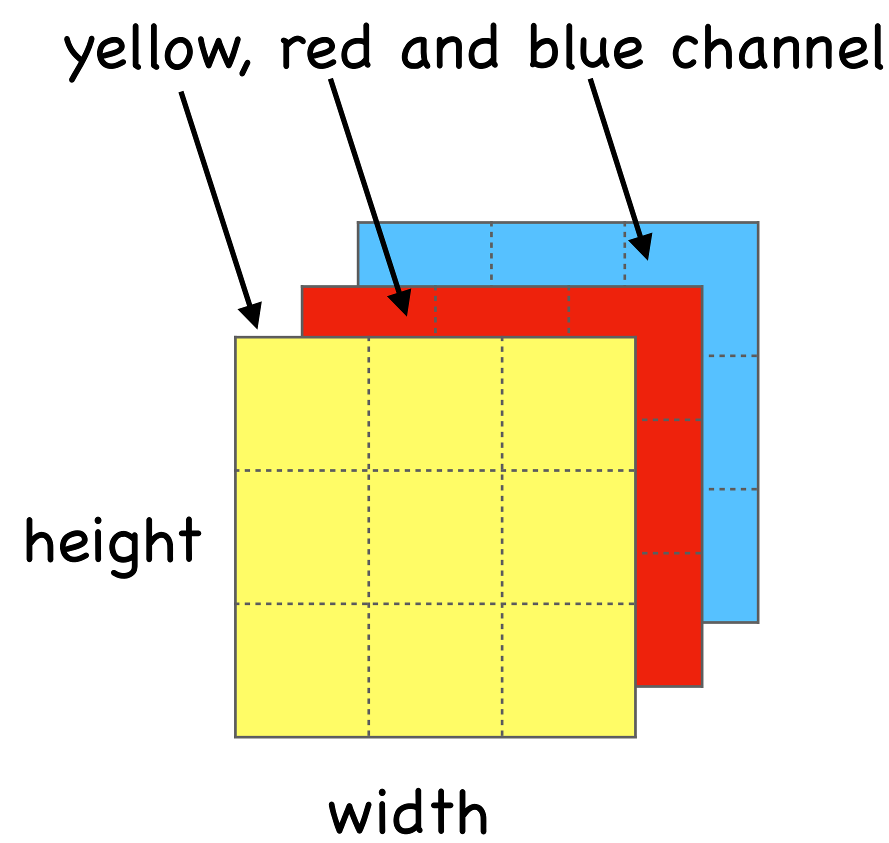
# Why do we need CNNs?

- The calculation of the fully connected layer is too large
- Location Information will get lost in forward neural network

# Combinatorial explosion

The calculation of the fully connected layer is too large

- Suppose there is a 32 weight × 32 height × 3 dimension image as input, and the output dimension is 28×28×6.
- So, 32×32×3=3072 , 28 × 28 × 6 = 4704.
- If we construct a forward neural network, one layer contains 3072 units, the next layer contains 4074 units. The two layers are connected to each other, and then the weight matrix is calculated, which is equal to 4074×3072≈**14 million**



# Losing location information

In a fully connected network, relative locations are lost

- The image is usually a three-dimensional shape in the direction of height, length, and channel.  However, when inputting to the full connection layer, we need to flatten the **3D** data into **1D** data.

- The image is a 3D shape that contains important spatial information. For example, spatially adjacent pixels are similar values, and each channel of the RBG has a close correlation, and there is no correlation between pixels that are far apart.

# Why we need CNNs

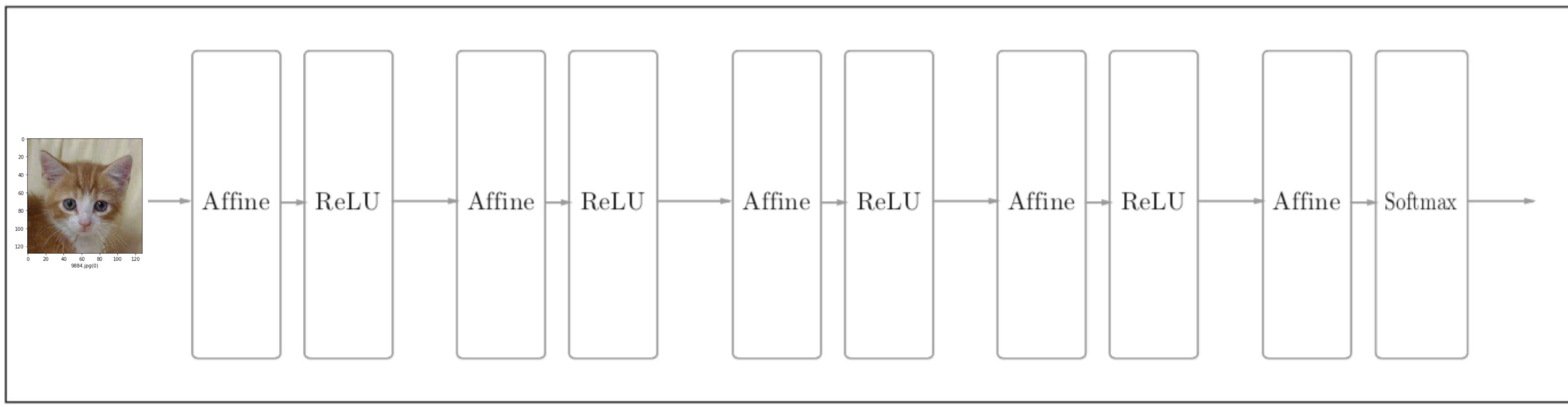
- The calculation of the fully connected (affine) layer is too large
- Location Information will lose in forward neural network

CNNs can solve those problems!

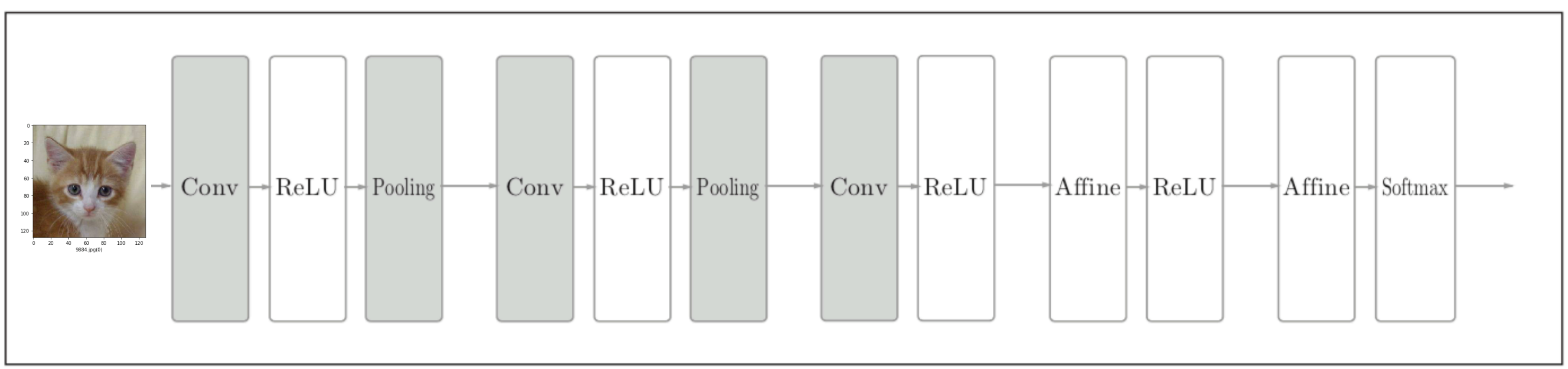
# C. The Structure of CNNs

# The Structure of CNNs

**DNNs**




**CNNs**

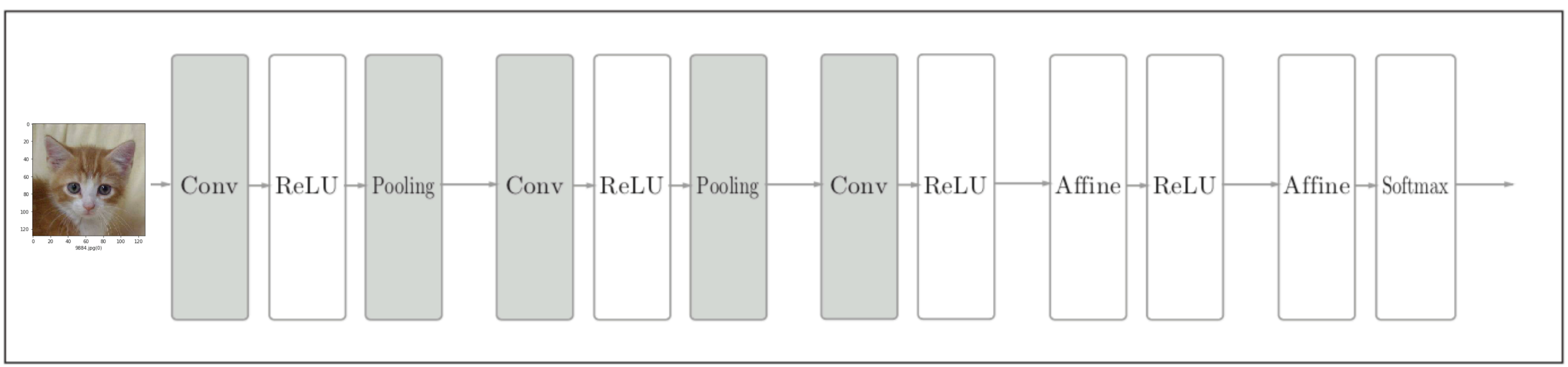


# The Structure of CNNs

- Two new layers:

  1. Convolutional Layer

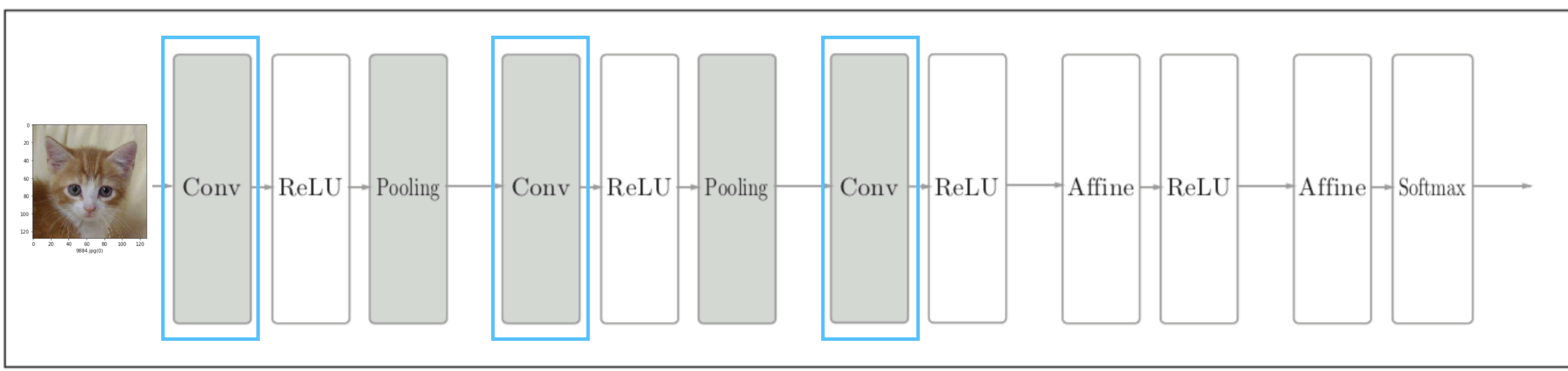
  2. Pooling Layer



# C1. Convolution

# Convolution

- Convolution is just another mathematical operation.



# Convolution operation

First, we have a 4x4 input data and 3x3 filter.

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

=

Input data
4 x 4

Filter
3 x 3

# Convolution operation

For input data, we apply the filter window.

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

=

Input data
4 x 4

Filter
3 x 3

# Convolution operation

**Take the product of two corresponding Numbers**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

=

-1*0 = 0, 25*0 = 0, 75*1 = 75

-2*0 = 0, 0*75= 0, 2*80 = 160

-2*0 = 0, 0*75= 0, 1*80 = 160

Input data
4 x 4

Filter
3 x 3

# Convolution operation

-1*0 = **0**, 25*0 = **0**, 75*1 = **75**

-2*0 = **0**, 0*75= **0**, 2*80 = **160**

-2*0 = **0**, 0*75= **0**, 1*80 = **160**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

=

| | | |
|---|---|---|
| 0 | 0 | 75 |
| 0 | 0 | 80 |
| 0 | 0 | 80 |

Input data
4 x 4

Filter
3 x 3

# Convolution operation

**Finally, sum them up**

$$\begin{bmatrix} 0 & 25 & 75 & 80 & 80 \\ 0 & 75 & 80 & 80 & 80 \\ 0 & 75 & 80 & 80 & 80 \\ 0 & 70 & 75 & 80 & 80 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 75 \\ 0 & 0 & 80 \\ 0 & 0 & 80 \end{bmatrix} \quad \Sigma$$

0+0+75+

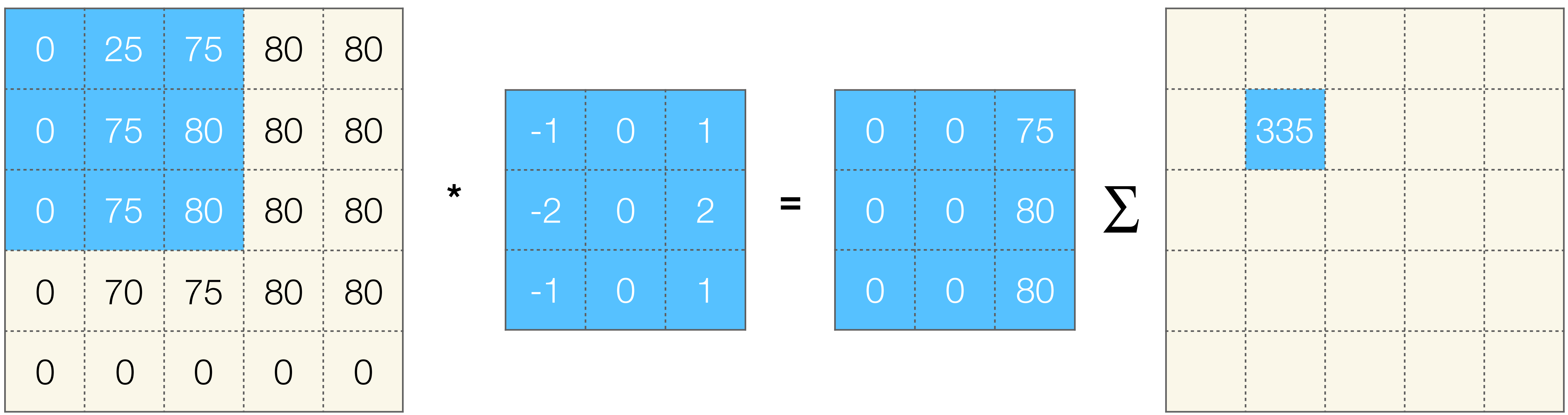
0+0+80+

0+0+80= **335**

Input data
4 x 4

Filter
3 x 3

# Convolution operation




Input data
4 x 4

Filter
3 x 3

# Convolution operation

This is the process of convolution operation, we give it a new sign $\circledast$

$$\begin{bmatrix}0&25&75&80&80\\0&75&80&80&80\\0&75&80&80&80\\0&70&75&80&80\\0&0&0&0&0\end{bmatrix} * \begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix} = \begin{bmatrix}0&0&75\\0&0&80\\0&0&80\end{bmatrix} \; \Sigma \; \begin{bmatrix}\;&\;&\;&\;&\;\\ &335&&&\\ &&&&\\ &&&&\\ &&&&\end{bmatrix}$$

$\circledast$

**Convolution Operation**

# Convolution operation

- For input data, the convolution operation slides the filter window at certain intervals to compute the output.
- Let’s do see how it works step by step.

# Convolution operation

For input data, the convolution operation slides the filter window at certain intervals to compute output.

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

$\circledast$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | | | |
| | | | | |
| | | | | |
| | | | | |

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| **0** | **25** | **75** | 80 | 80 |
| **0** | **75** | **80** | 80 | 80 |
| **0** | **75** | **80** | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

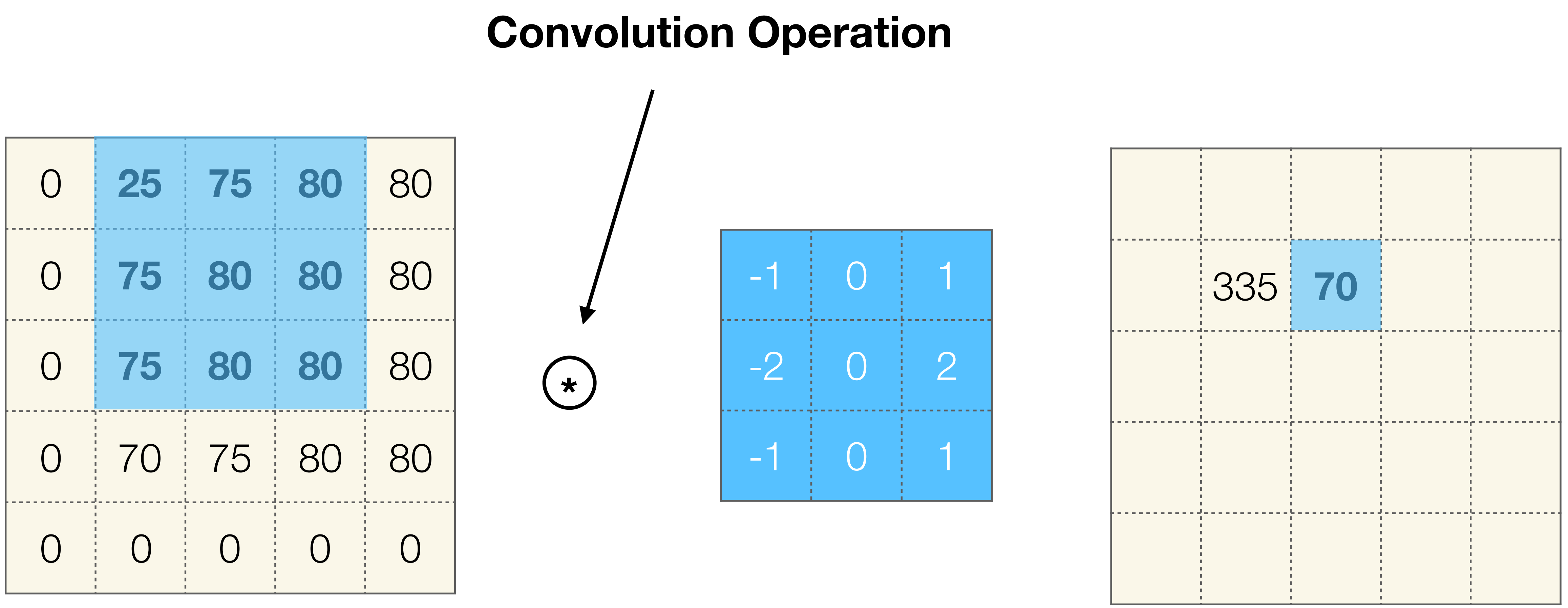
*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | **335** | | | |
| | | | | |
| | | | | |
| | | | | |

**75+160+80**

# Convolution operation




**-25 + 80+75*(-2) + 160 -75 +80**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | **75** | **80** | **80** |
| 0 | 75 | **80** | **80** | **80** |
| 0 | 75 | **80** | **80** | **80** |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | **5** | |
| | | | | |
| | | | | |
| | | | | |

**-75 + 80+80*(-2) + 160 -80 +80**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| **0** | **75** | **80** | 80 | 80 |
| **0** | **75** | **80** | 80 | 80 |
| **0** | **70** | **75** | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | 5 | |
| | **315** | | | |
| | | | | |
| | | | | |

**80 + 160 + 75**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | **75** | **80** | **80** | 80 |
| 0 | **75** | **80** | **80** | 80 |
| 0 | **70** | **75** | **80** | 80 |
| 0 | 0 | 0 | 0 | 0 |

$\circledast$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | 5 | |
| | 315 | **20** | | |
| | | | | |
| | | | | |

**-75 + 80 + 75*(-2) +160 - 75 + 80**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | **80** | **80** | **80** |
| 0 | 75 | **80** | **80** | **80** |
| 0 | 70 | **75** | **80** | **80** |
| 0 | 0 | 0 | 0 | 0 |

$\circledast$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | 5 | |
| | 315 | 20 | 5 | |
| | | | | |
| | | | | |

**-80 + 80 + 80*(-2) +160 - 75 + 80**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| **0** | **75** | **80** | 80 | 80 |
| **0** | **70** | **75** | 80 | 80 |
| **0** | **0** | **0** | 0 | 0 |

*

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | 5 | |
| | 315 | 20 | 5 | |
| | **230** | | | |
| | | | | |

**80 + 75*2**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | **75** | **80** | **80** | 80 |
| 0 | **70** | **75** | **80** | 80 |
| 0 | **0** | **0** | **0** | 0 |

$\circledast$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | 5 | |
| | 315 | 20 | 5 | |
| | 230 | **15** | | |
| | | | | |

**-75+80+75*(-2) +80*2**

# Convolution operation

**Convolution Operation**

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | **80** | **80** | **80** |
| 0 | 70 | **75** | **80** | **80** |
| 0 | 0 | **0** | **0** | **0** |

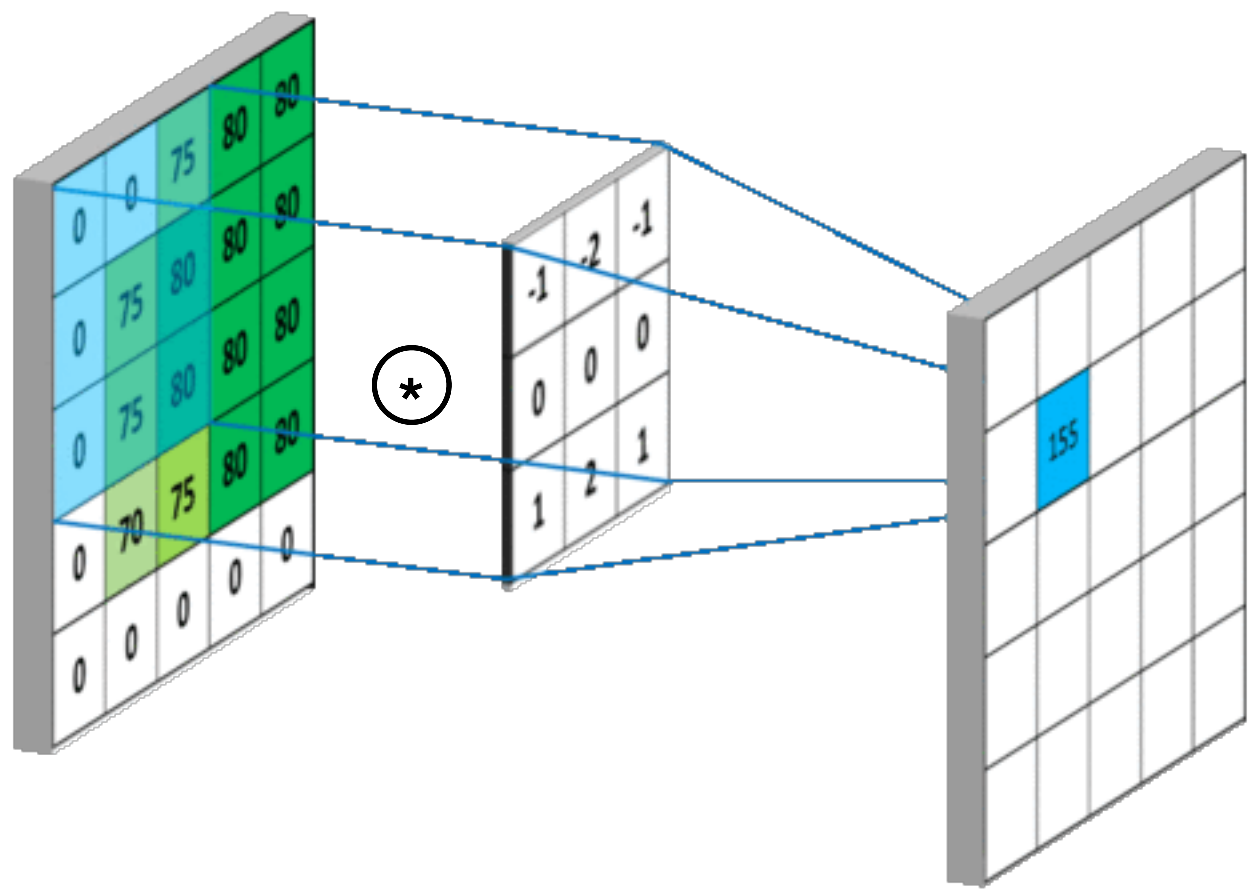
$\circledast$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | |
| | 335 | 70 | 5 | |
| | 315 | 20 | 5 | |
| | 230 | 15 | 10 | |
| | | | | |

**-80+80+75*(-2) +80*2**

# Convolution operation




Click to Play

**Convolution Operation**

# Bias

Add Bias to Each of element

$$\begin{bmatrix} 0 & 25 & 75 & 80 & 80 \\ 0 & 75 & 80 & 80 & 80 \\ 0 & 75 & 80 & 80 & 80 \\ 0 & 70 & 75 & 80 & 80 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \circledast \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 335 & 70 & 5 \\ 315 & 20 & 5 \\ 230 & 15 & 10 \end{bmatrix} + \boxed{5} \rightarrow \begin{bmatrix} 340 & 75 & 10 \\ 320 & 25 & 10 \\ 235 & 20 & 15 \end{bmatrix}$$

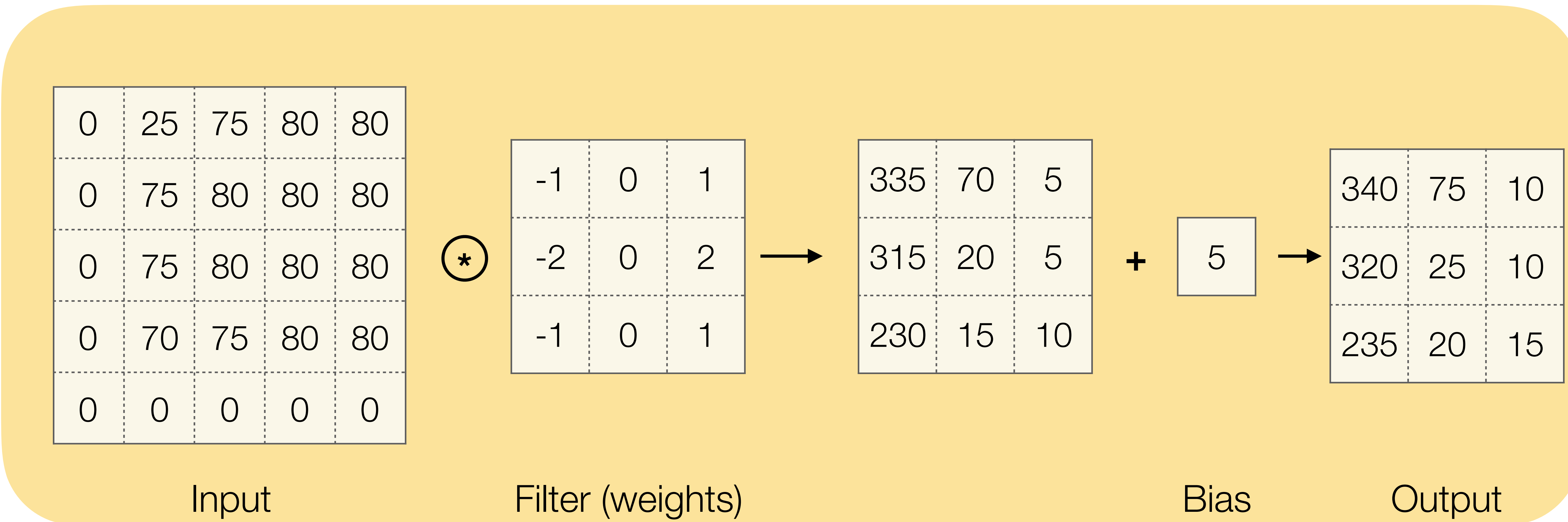
Input data
4 x 4

Filter
3 x 3

Bias

Output

# Convolutional Layer



# Why we need CNNs

- If each filter is 5 × 5, a filter has 25 parameters, plus the Bia parameter, then each filter has 26 parameters. There are a total of 6 filters, so the total number of parameters is **156**. Far less than the equivalent fully-collected layer, which has **14 million** parameters.

- When the input data is an image, the convolutional layer will be 3D. The data is received in the form of input data and is also output to the next layer in the form of 3D data. Therefore, in CNNs, it’s possible to correctly understand data having shapes such as images.

# Padding

## What’s the problem?

Only use one time
of this edge element 0 to calculate

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

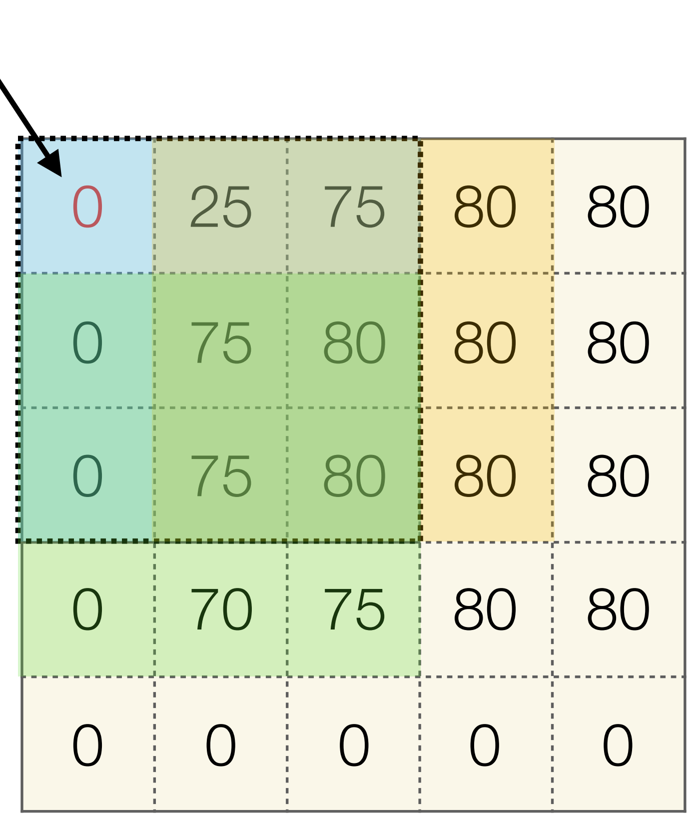
Use several times
of this central element 80 to calculate

| | | | | |
|---|---|---|---|---|
| 0 | 25 | 75 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 75 | 80 | 80 | 80 |
| 0 | 70 | 75 | 80 | 80 |
| 0 | 0 | 0 | 0 | 0 |

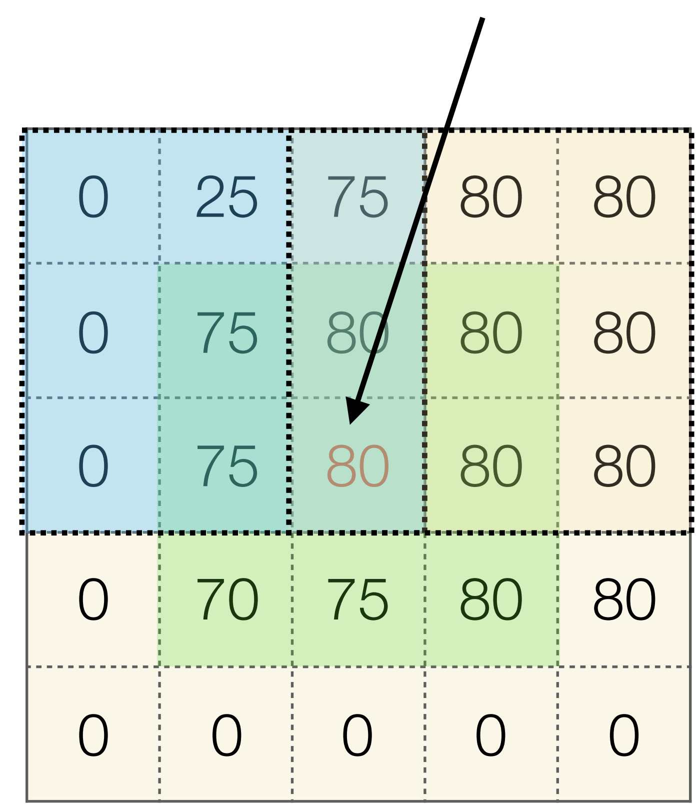
# Padding

We lose edge information in this way.
How to solve this problem?

Only use one time
of this edge element 0 to calculate




Use several times
of this central element 80 to calculate

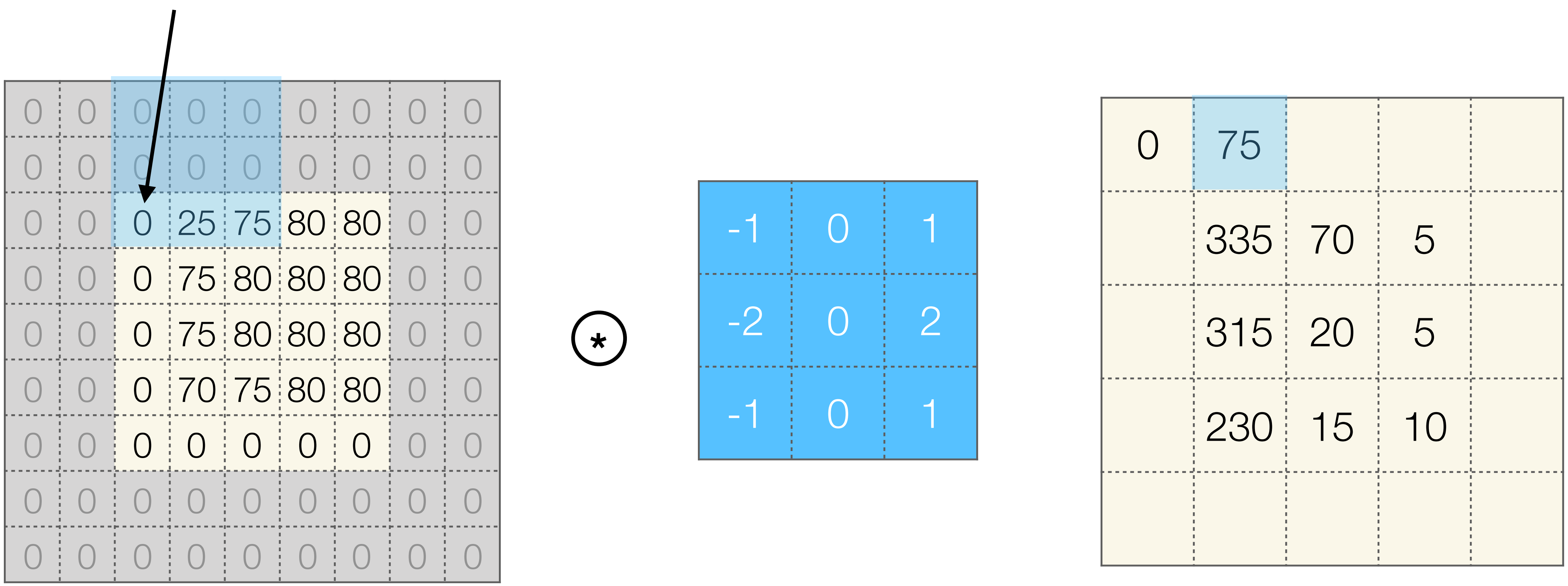


# Padding

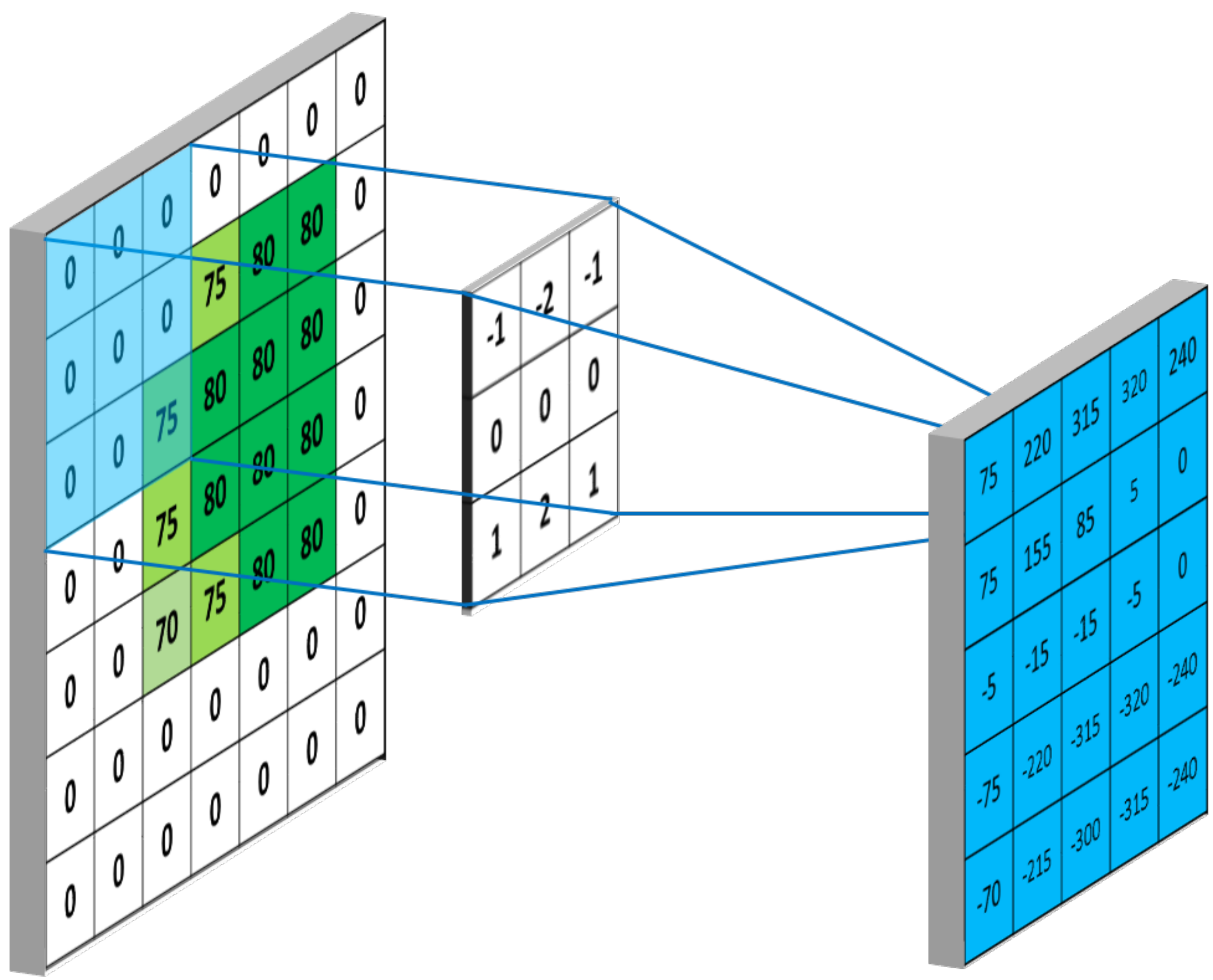
## Padding it with a margin of 2!

This means padding around with a margin of 2 pixel.

Now, we use several times
of this edge element 0 to calculate too.



# Padding


0 0 0 0 0 0 0
0 0 0 75 80 80 0
0 0 75 80 80 80 0
0 0 75 80 80 80 0
0 0 70 75 80 80 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
-1 -2 -1
0 0 0
1 2 1
75 220 315 320 240
75 155 85 5 0
-5 -15 -15 -5 0
-75 -220 -315 -320 -240
-70 -215 -300 -315 -240

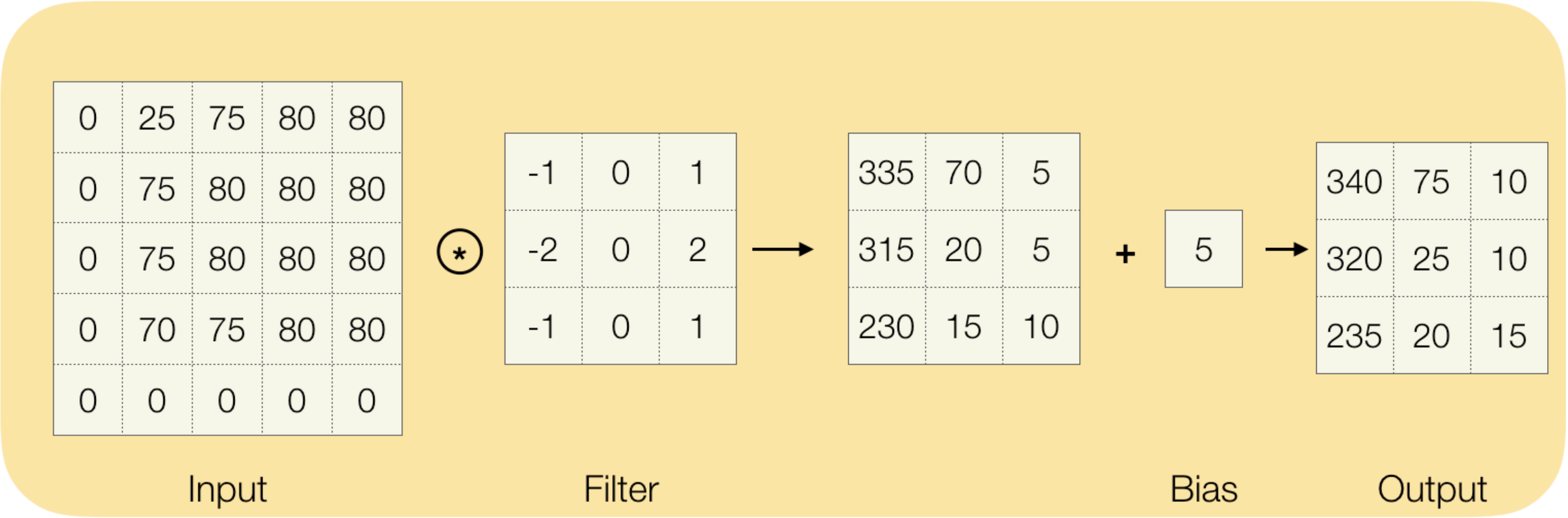
# Stride

Stride is the amount by which the filter is moved as it passes over the image.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 25 | 75 | 80 | 80 | 0 | 0 |
| 0 | 0 | 0 | 75 | 80 | 80 | 80 | 0 | 0 |
| 0 | 0 | 0 | 75 | 80 | 80 | 80 | 0 | 0 |
| 0 | 0 | 0 | 70 | 75 | 80 | 80 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Stride = 2

# Convolutional Layer




```
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu',
input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))
```

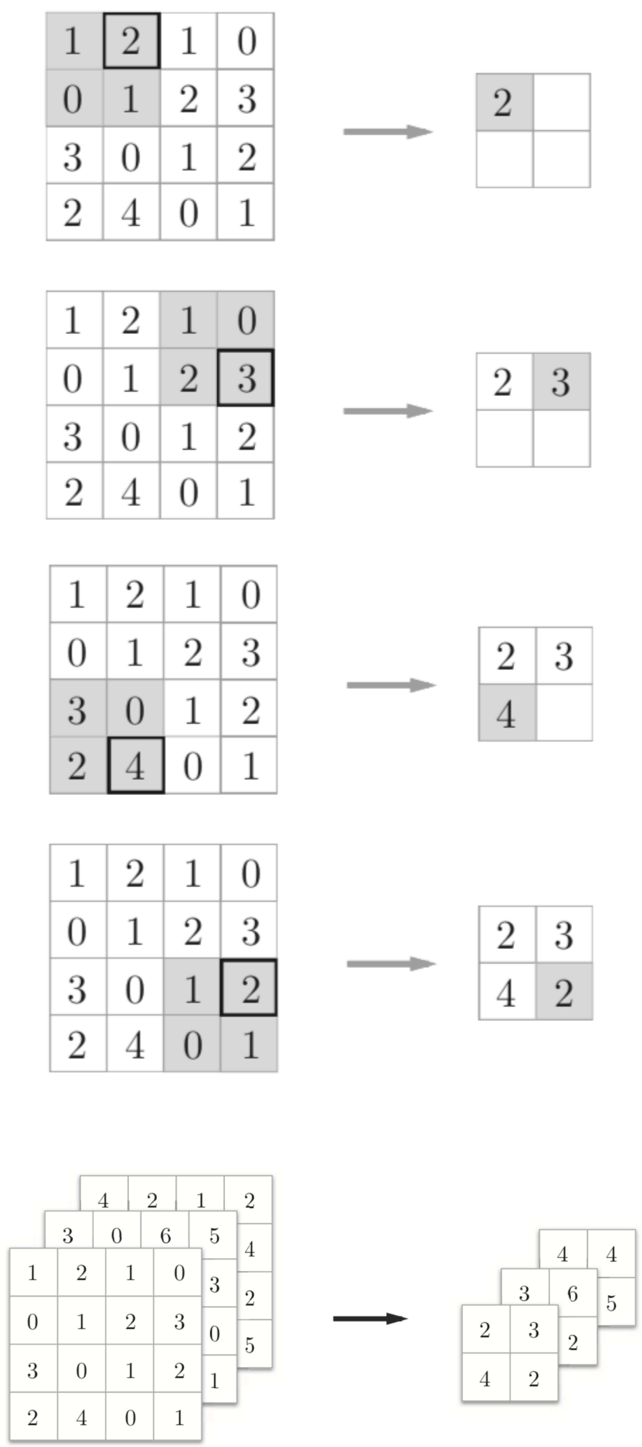
# C2. Pooling

# Pooling

- It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture

- Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting

# Max Pooling




- The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.

- The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations.

- The depth dimension remains unchanged.

# Code

```
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu',
input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

# Some typical CNNs

# Some typical CNNs

- LeNet
- AlexNet
- GoogleLeNet
- VGGNet
- ResNet

# LeNet

- The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990’s. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.

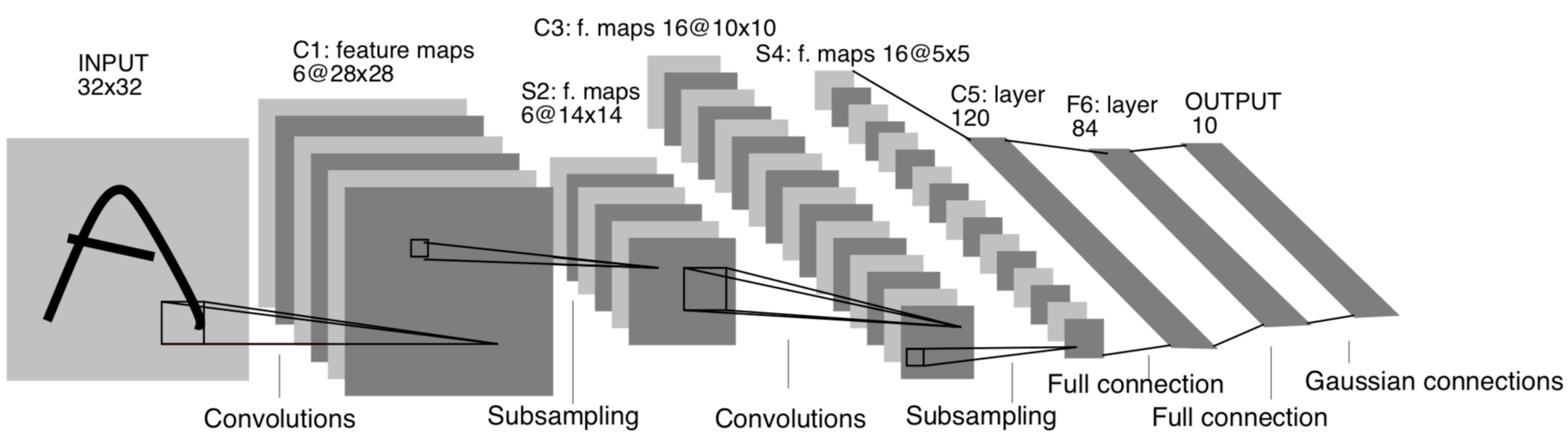


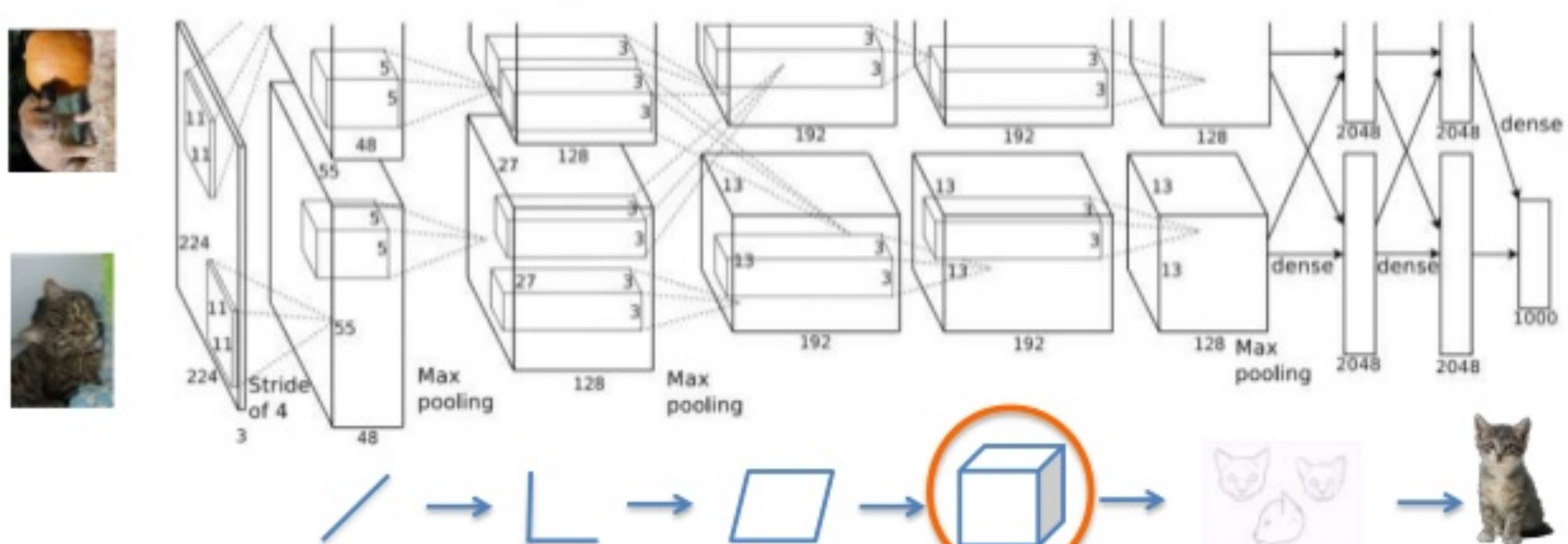

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

# AlexNet

- 2012 ILSVRC winner （top 5 error of 16% compared to runner-up with 26% error）

AlexNet (Krizhevsky et al. 2012)

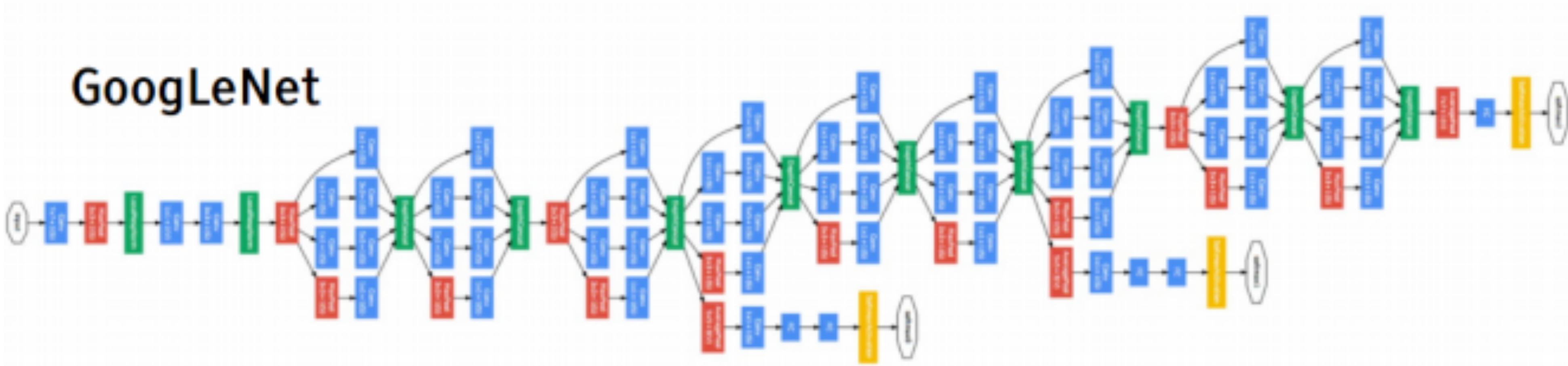
***The class with the highest likelihood is the one the DNN selects***




When AlexNet is processing an image, this is what is happening at each layer.

# GoogLeNet

- The ILSVRC 2014 winner
- Provide an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M)



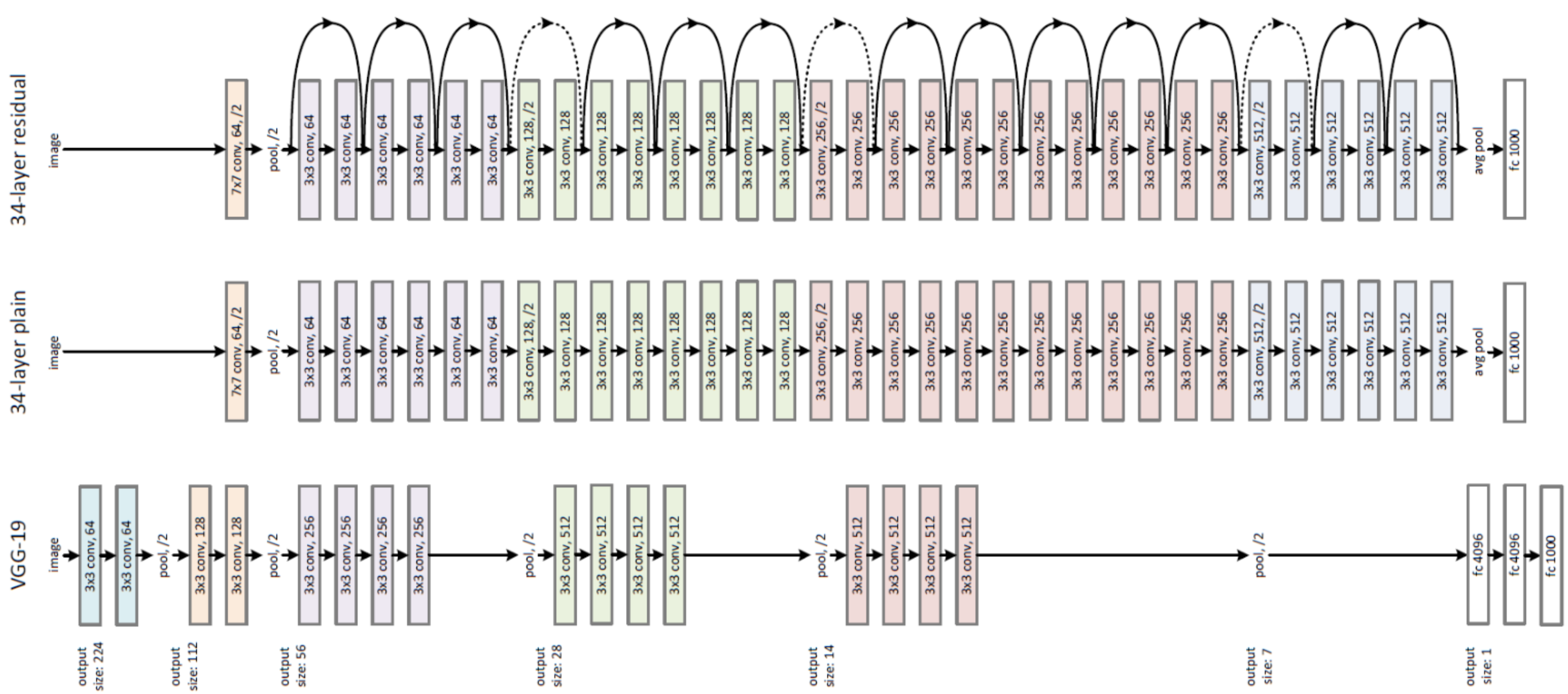
# VGGNet

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

- It shows that the depth of the network is a critical component for good performance.
- A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M).

# ResNet

- The winner of ILSVRC 2015
- It features special skip connections and a heavy use of batch normalization.



# Exercise: dog or cat?

# Exercise: dog or cat?

https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification

Thanks!