CSE7850/CX4803 Machine Learning in Computational Biology

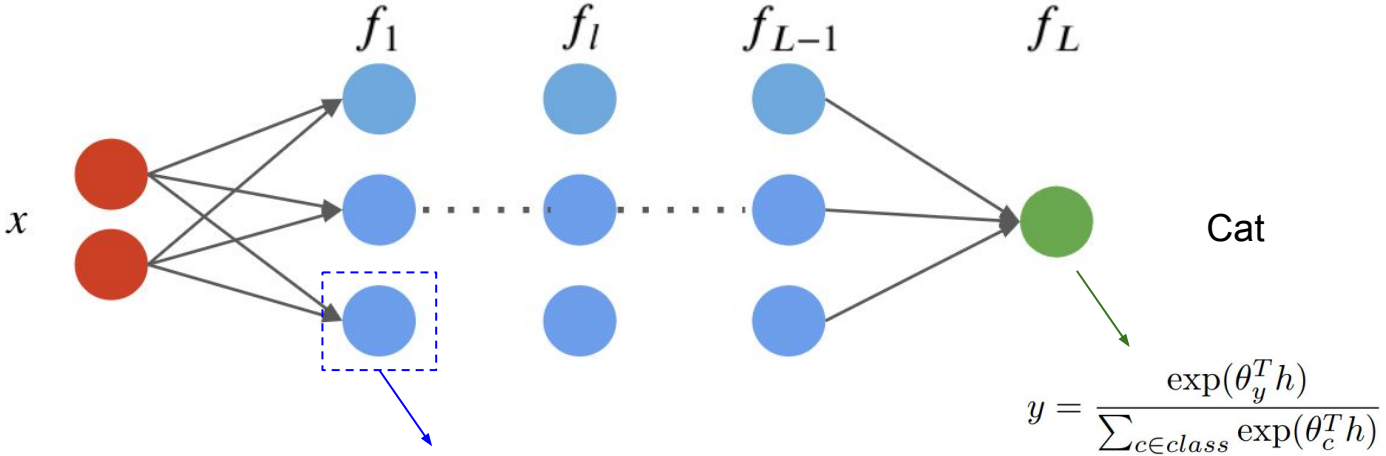
Lecture 8: Deep Learning

Yunan Luo

Acknowledgements: Some of the slides were adapted from lectures of Matt Gormley and Volodymyr Kuleshov

Recap: neural network

A (fully connected) neural network is a computational model that consists of a composition of multiple neural network layers



$$f(x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)}$$

https://github.com/kuleshov/cornell-cs5785-2021-applied-ml/blob/main/notebooks/lecture19-deep-learning.ipynb

Deep Learning

A class of machine learning methods that emphasizes:

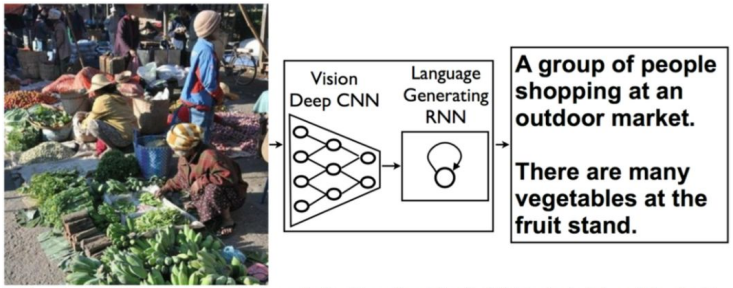
- **Deep** neural networks
- **Large** unstructured datasets, especially images, text, and audio
- Modern computational resources, like **GPU**s

Expressivity of Deep Models

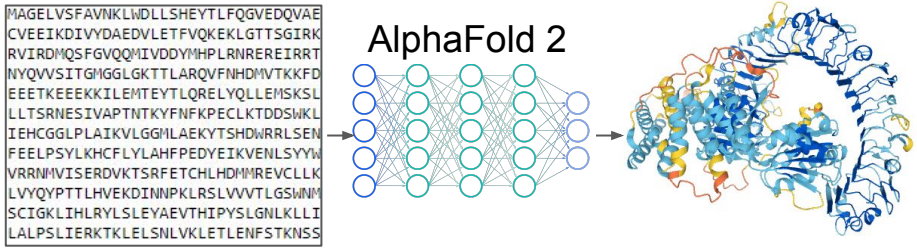
Deep neural networks are very powerful models in part because they can represent well complex mappings between x and y.

- Shallow neural networks can represent any function, but need very large hidden layers.
- Deep networks can represent very complex $\mathcal{X} \rightarrow \mathcal{Y}$ mappings with fewer parameters.

In practice, deep neural networks can learn very complex mappings such as $\text{image} \rightarrow \text{text description}$ that other algorithms cannot.



Ref: Learning CNN-LSTM Architectures for Image Caption Generation: https://cs224d.stanford.edu/reports/msoh.pdf

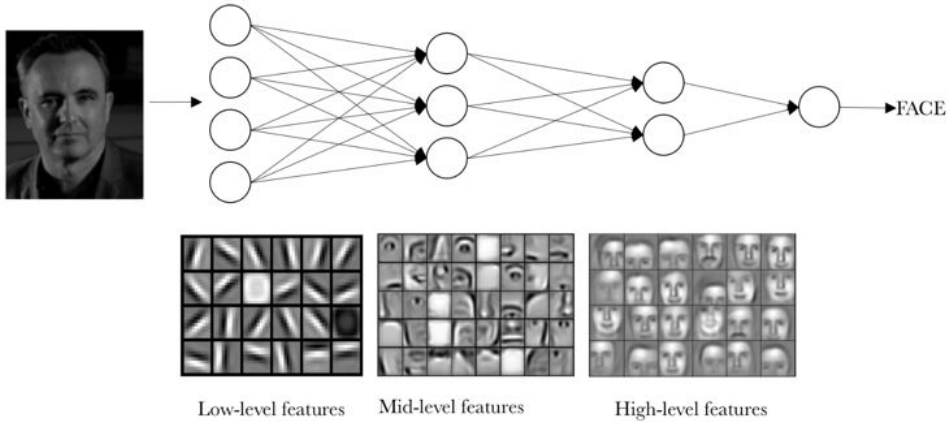


Representation Learning

How does a deep neural network use its representational power?

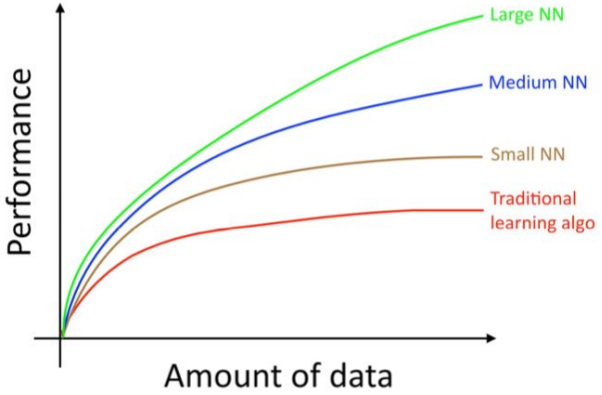
- The sequential layers can learn to represent data at an increasing level of abstraction
- This can also be interpreted as learning: each layer maps input to a more abstract feature space that is *learned* from data.

This is an example of representations that a deep neural network learns from data.

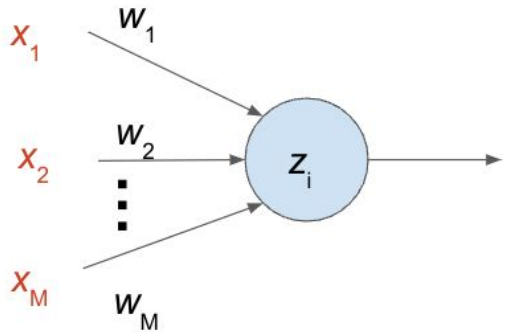


Scaling to Large Datasets

- Deep neural nets also scale to very large datasets
    - Classical algorithms like linear regression saturate after a certain dataset size. Deep learning models typically keep improving as we add more data.

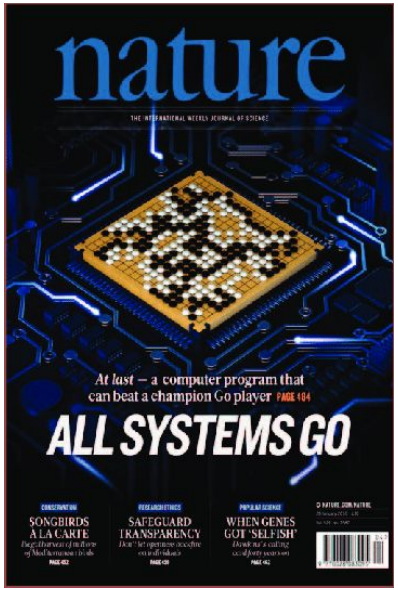


- Deep learning datasets benefit from large datasets because they can easily use specialized computational hardware.
    - Neural networks mostly implement linear algebra, and benefit from hardware acceleration.
    - Today, they are trained on graphical processing units (GPUs), or even more specialized hardware.

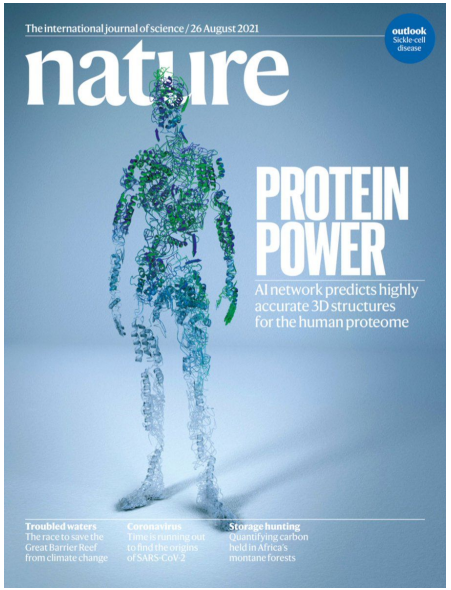


$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma\left(\sum_i w_i \cdot x_i\right) = \sigma(\boldsymbol{w}^T \boldsymbol{x})$$

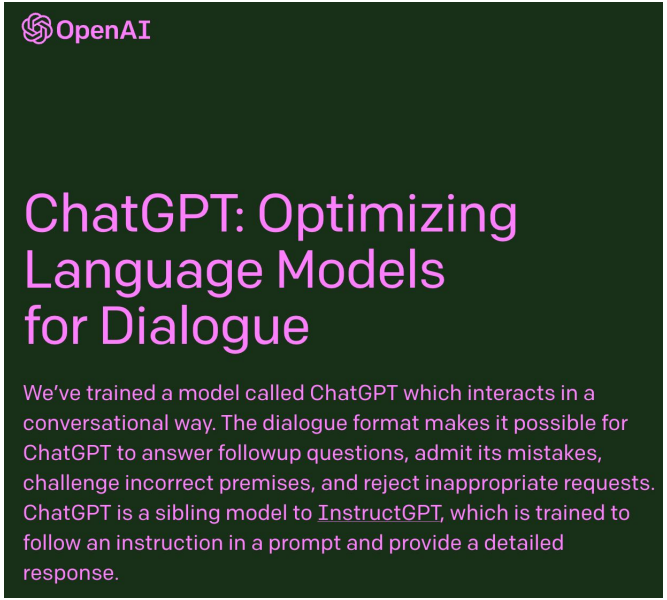
Successes of Deep Learning



Alpha Go (2016)



Alpha Fold (2018-2021)



ChatGPT (2022)

Successes of Deep Learning

Translation

Speech recognition

Successes of Deep Learning

A beautiful field of vibrant flowers atmospheric, hyper realistic, 8k, epic composition, cinematic, octane render, artstation landscape vista photography by Carr Clifton & Galen Rowell, 16K resolution, Landscape veduta photo by Dustin Lefevre & tdraw, 8k resolution, detailed landscape painting by Ivan Shishkin, DeviantArt, Flickr, rendered in Enscape, Miyazaki, Nausicaa Ghibli, Breath of The Wild, 4k detailed post processing, artstation, rendering by octane, unreal engine —ar 16:9

Text-to-image generative models
(Stable diffusion, DALL-E, Imagen, ...)

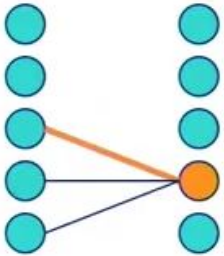
Image from: https://prompthero.com/prompt/722b1199390

Pros and Cons of Deep Neural Networks

- **Pros**: Deep neural nets (DNNs) are among the most powerful ML models.
  - DNNs set a new state-of-the-art in many application areas.
  - They can be combined with many algorithms (supervised, unsupervised), and significantly improve them.
  - Many specialized software and hardware platforms for DNNs.

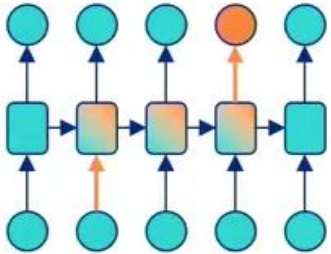
- **Cons**: A number of technical challenges:
  - DNNs can be slow and hard to train and require a lot of data
  - Deep neural networks are prone to overfitting.
  - Requires a massive amount of data

Deep Learning Models and Implementations



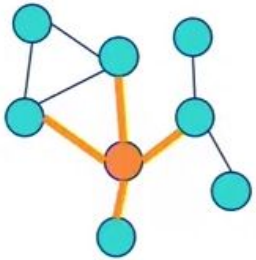
Convolutional Networks
(e.g. computer vision)

- data in regular grid
- information flow to local neighbours
- AlphaFold 1



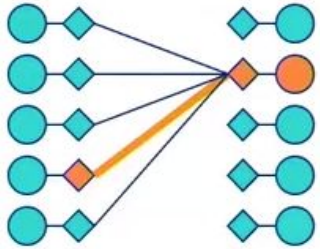
Recurrent Networks
(e.g. language)

- data in ordered sequence
- information flow sequentially



Graph Networks (e.g. recommender systems or molecules)

- data in fixed graph structure
- information flow along fixed edges



Attention Module (e.g. language)

- data in unordered set
- information flow dynamically controlled by the network (via keys and queries)

PyTorch TensorFlow Keras Google colab

Image from John Jumper's talk "Highly accurate protein structure prediction with AlphaFold"

An example of Deep Learning Models:

Convolutional Neural Network (CNN)

Background: Computer Vision

Image Classification

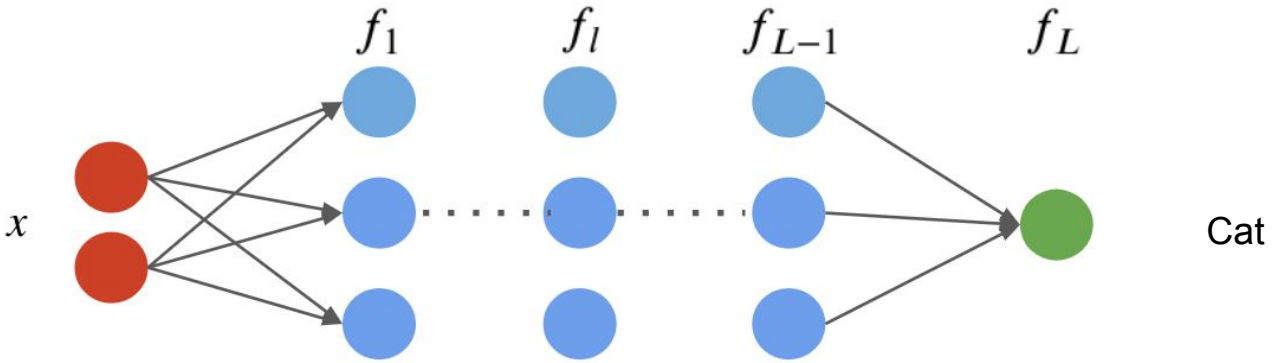


Image Classification

- ImageNet LSVRC-2011 contest:
  - **Dataset**: 1.2 million labeled images, 1000 classes
  - **Task**: Given a new image, label it with the correct class
  - **Multiclass** classification problem
- • Examples from http://image-net.org/

IMAGENET

SEARCH

14,197,122 images, 21841 synsets indexed

Home | Explore
About | Download

Not logged in. Login | Signup

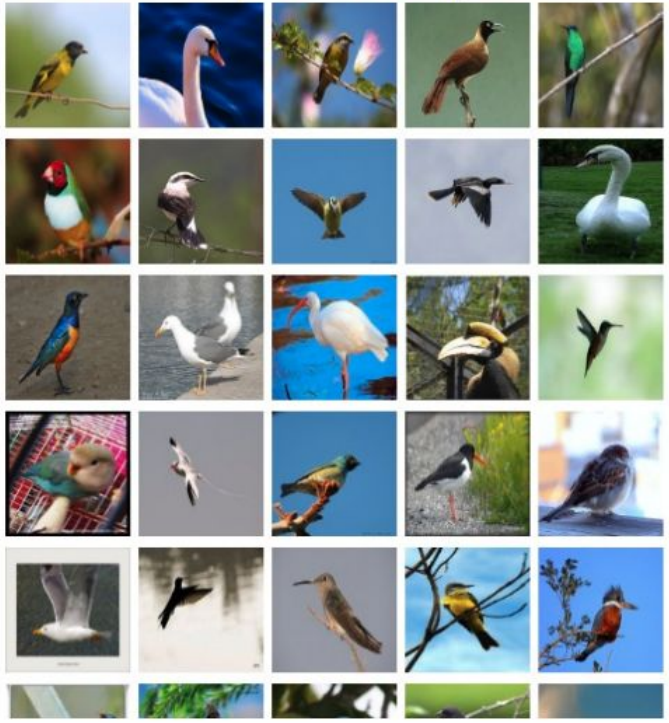
Bird

Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings

2126 pictures | 92.85% Popularity Percentile | Wordnet IDs

- marine animal, marine creature, sea animal, sea creature (1)
- scavenger (1)
- biped (0)
- predator, predatory animal (1)
- larva (49)
- acrodont (0)
- feeder (0)
- stunt (0)
- chordate (3087)
  - tunicate, urochordate, urochord (6)
  - cephalochordate (1)
  - vertebrate, craniate (3077)
    - mammal, mammalian (1169)
    - bird (871)
      - dickeybird, dickey-bird, dickybird, dicky-bird (0)
      - cock (1)
      - hen (0)
      - nester (0)
      - night bird (1)
      - bird of passage (0)
      - protoavis (0)
      - archaeopteryx, archeopteryx, Archaeopteryx lithographi
      - Sinornis (0)
      - Ibero-mesornis (0)
      - archaeornis (0)
      - ratite, ratite bird, flightless bird (10)
      - carinate, carinate bird, flying bird (0)
      - passerine, passeriform bird (279)
      - nonpasserine bird (0)
      - bird of prey, raptor, raptorial bird (80)
      - gallinaceous bird, gallinacean (114)

Treemap Visualization | Images of the Synset | Downloads



IMAGENET

SEARCH

14,197,122 images, 21841 synsets indexed

Home | Explore
About | Download

Not logged in. Login I Signup

German iris, Iris kochii

Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica

469 pictures | 49.6% Popularity Percentile | Wordnet IDs

- halophyte (0)
- succulent (39)
- cultivar (0)
- cultivated plant (0)
- weed (54)
- evergreen, evergreen plant (0)
- deciduous plant (0)
- vine (272)
- creeper (0)
- woody plant, ligneous plant (1868)
- geophyte (0)
- desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
- mesophyte, mesophytic plant (0)
- aquatic plant, water plant, hydrophyte, hydrophytic plant (11
- tuberous plant (0)
- bulbous plant (179)
  - iridaceous plant (27)
    - iris, flag, fleur-de-lis, sword lily (19)
      - bearded iris (4)
        - Florentine iris, orris, Iris germanica florentina, Iris
        - German iris, Iris germanica (0)
        - German iris, Iris kochii (0)
        - Dalmatian iris, Iris pallida (0)
      - beardless iris (4)
      - bulbous iris (0)
      - dwarf iris, Iris cristata (0)
      - stinking iris, gladdon, gladdon iris, stinking gladwyn,
      - Persian iris, Iris persica (0)
      - yellow iris, yellow flag, yellow water flag, Iris pseuda
      - dwarf iris, vernal iris, Iris verna (0)
      - blue flag, Iris versicolor (0)

Treemap Visualization | Images of the Synset | Downloads

IMAGENET

SEARCH

14,197,122 images, 21841 synsets indexed

Home | Explore
About | Download

Not logged in. Login I Signup

Court, courtyard

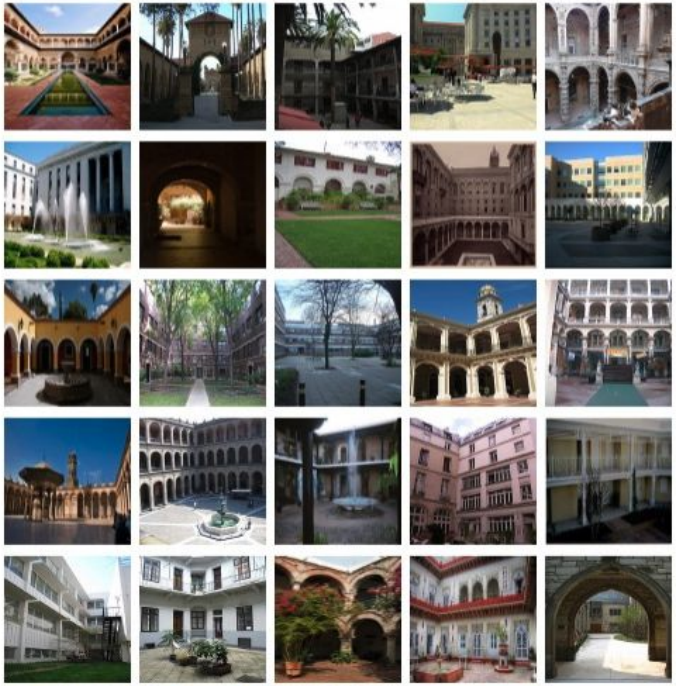
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"

165 pictures | 92.61% Popularity Percentile | Wordnet IDs

Numbers in brackets: (the number of synsets in the subtree).

- ImageNet 2011 Fall Release (32326)
  - plant, flora, plant life (4486)
  - geological formation, formation (175)
  - natural object (1112)
  - sport, athletics (176)
  - artifact, artefact (10504)
    - instrumentality, instrumentation (5494)
    - structure, construction (1405)
      - airdock, hangar, repair shed (0)
      - altar (1)
      - arcade, colonnade (1)
      - arch (31)
      - area (344)
        - aisle (0)
        - auditorium (1)
        - baggage claim (0)
        - box (1)
        - breakfast area, breakfast nook (0)
        - bullpen (0)
        - chancel, sanctuary, bema (0)
        - choir (0)
        - corner, nook (2)
        - court, courtyard (6)
          - atrium (0)
          - bailey (0)
          - cloister (0)
          - food court (0)
          - forecourt (0)
          - parvis (0)

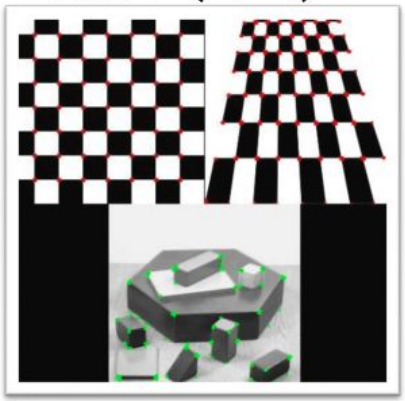
Treemap Visualization | Images of the Synset | Downloads



Feature Engineering for Image Classification

Edge detection (Canny)

Corner Detection (Harris)



Figures from http://opencv.org

Scale Invariant Feature Transform (SIFT)

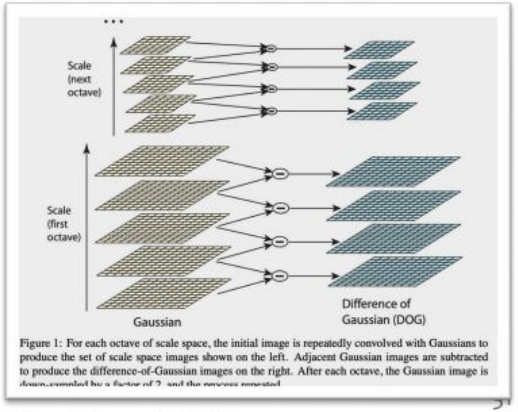
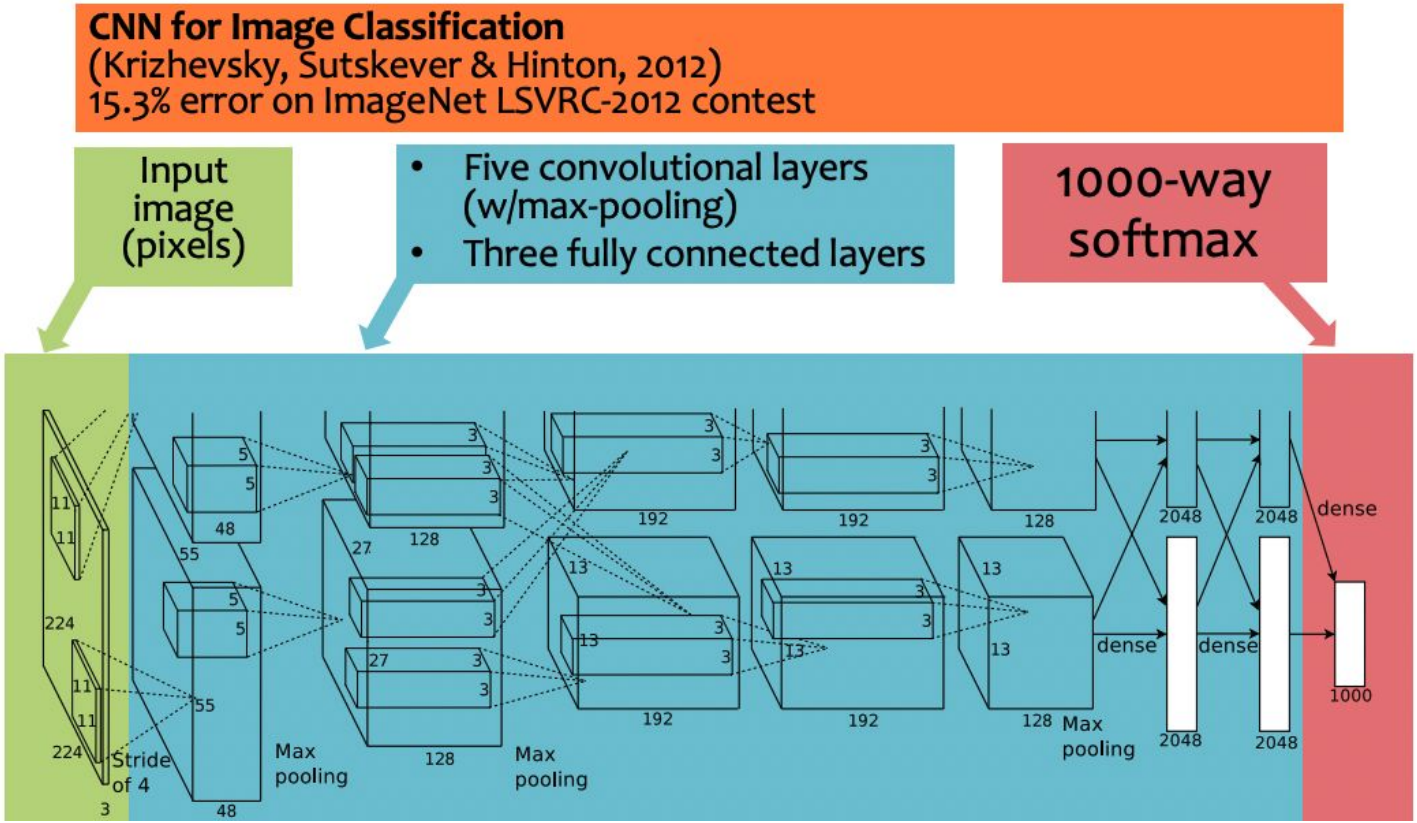


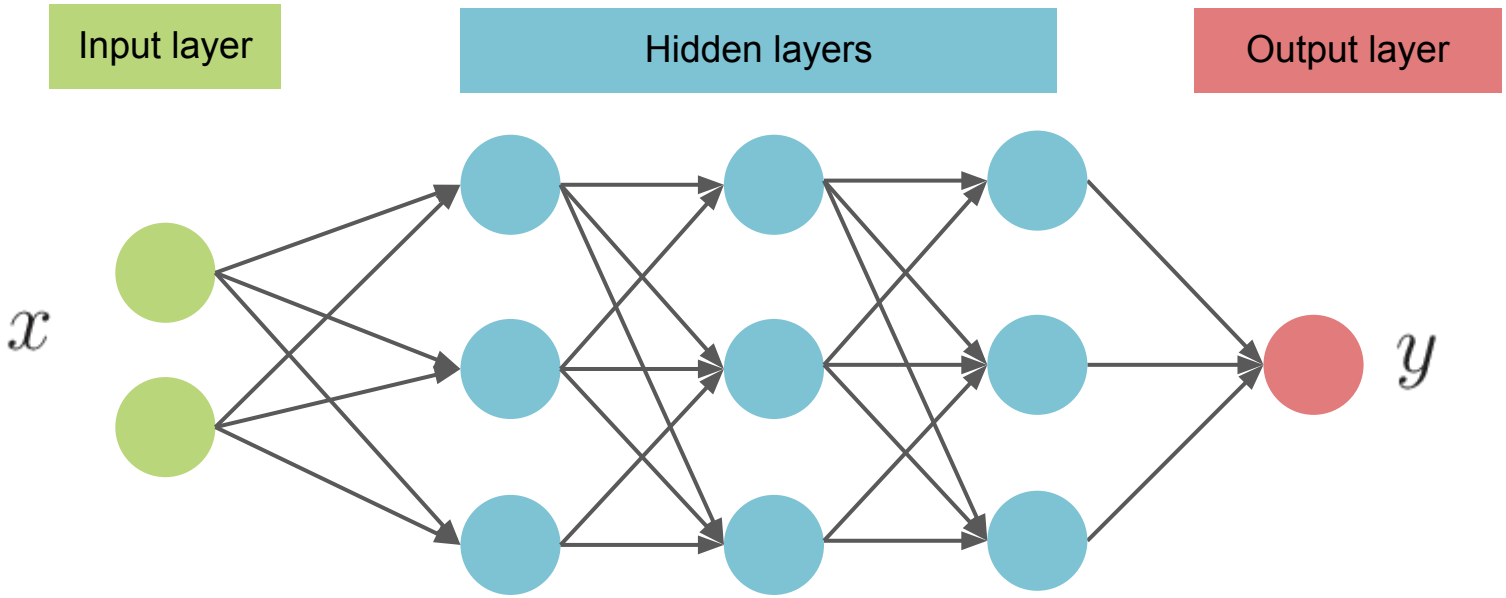
Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is

Figure from Lowe (1999) and Lowe (2004)

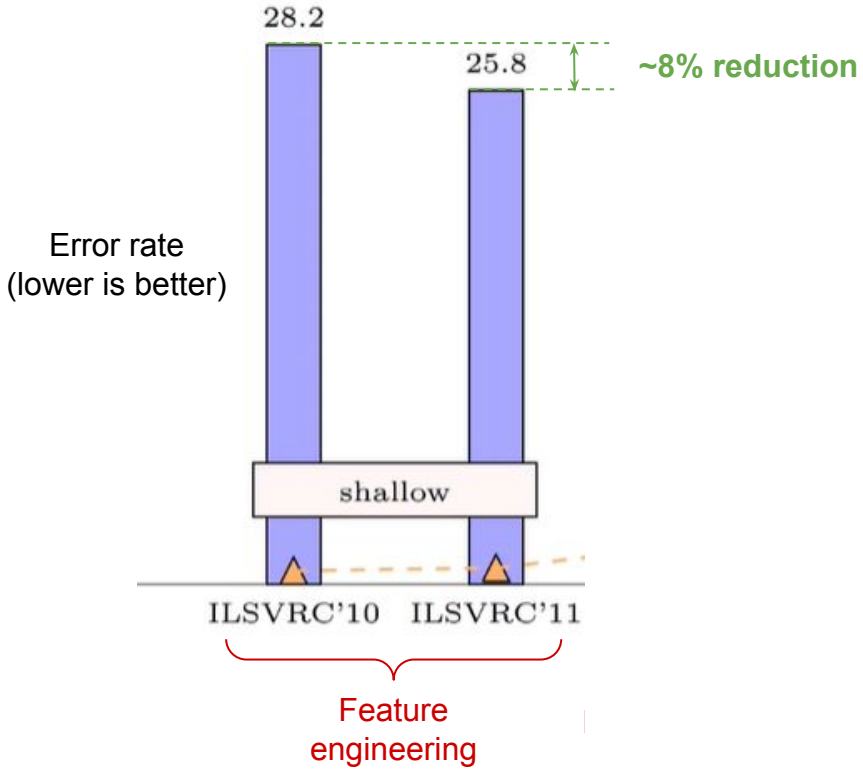
Deep Learning for Image Classification (2012)



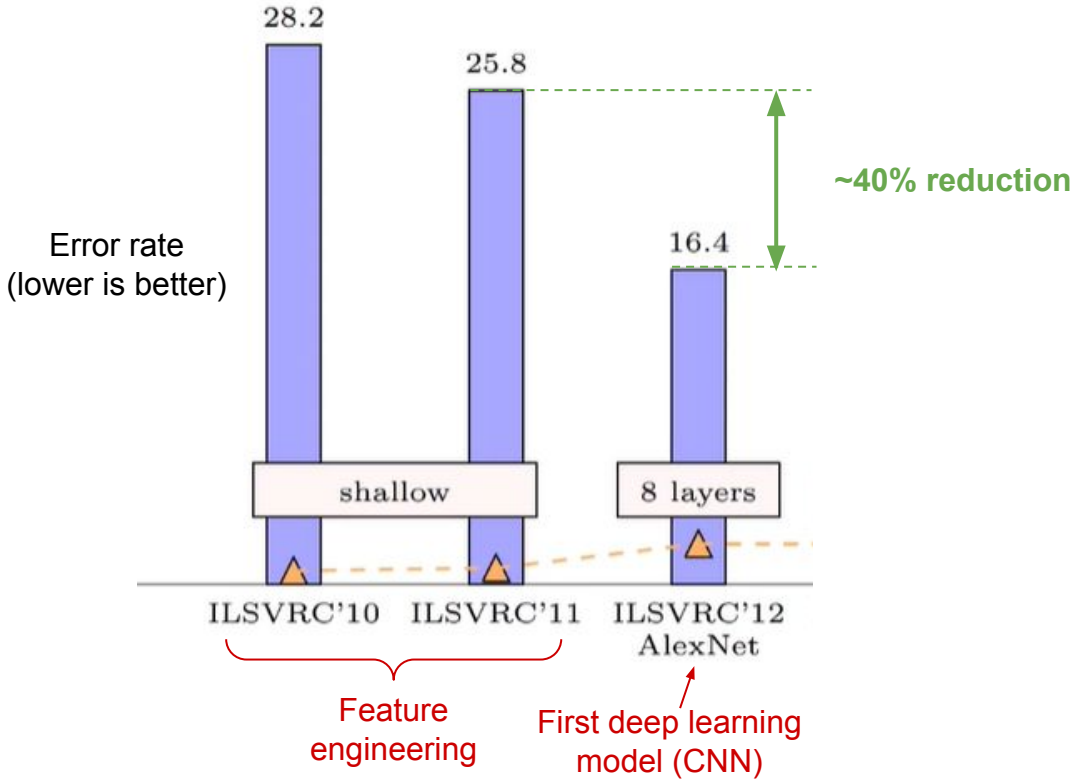
Same idea!



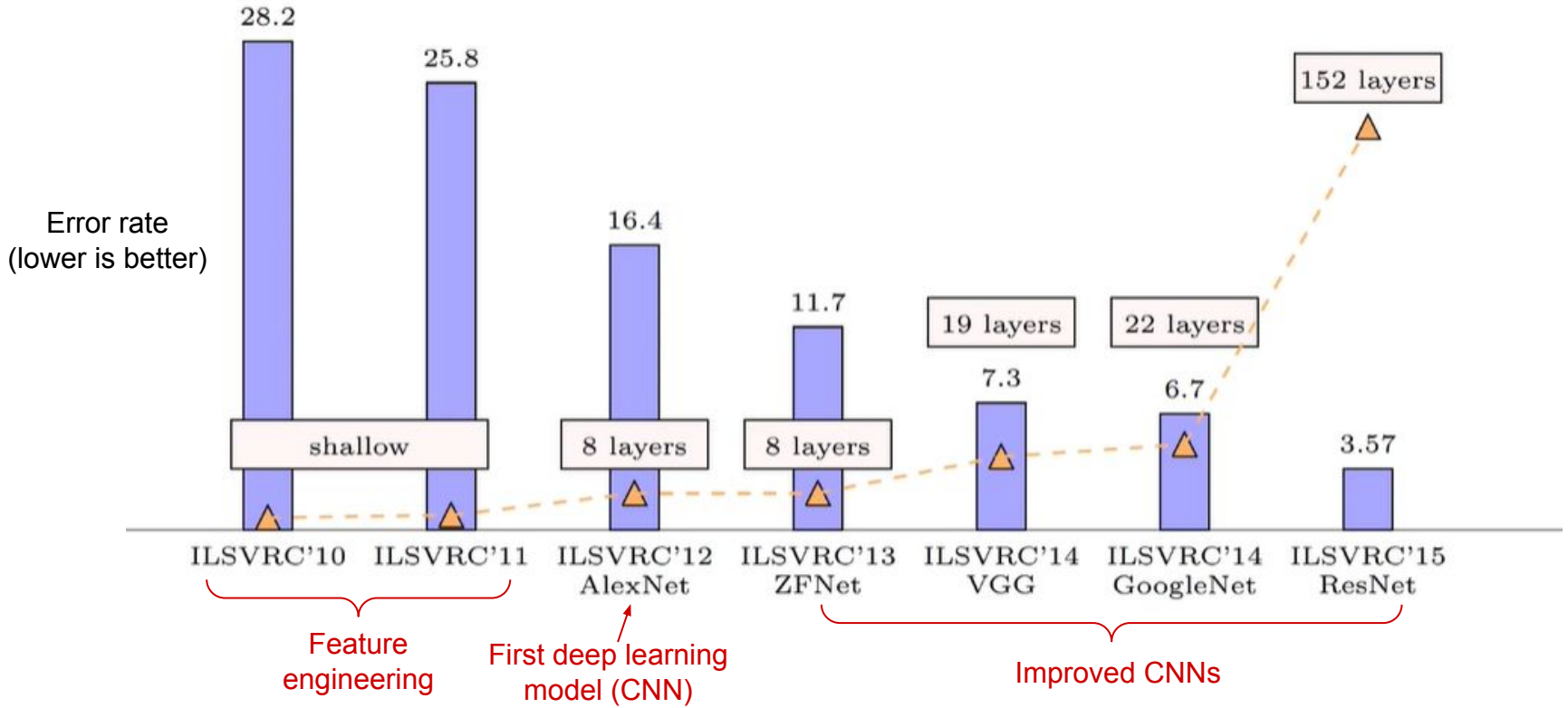
How Deep Learning Performed on the ImageNet Contest?



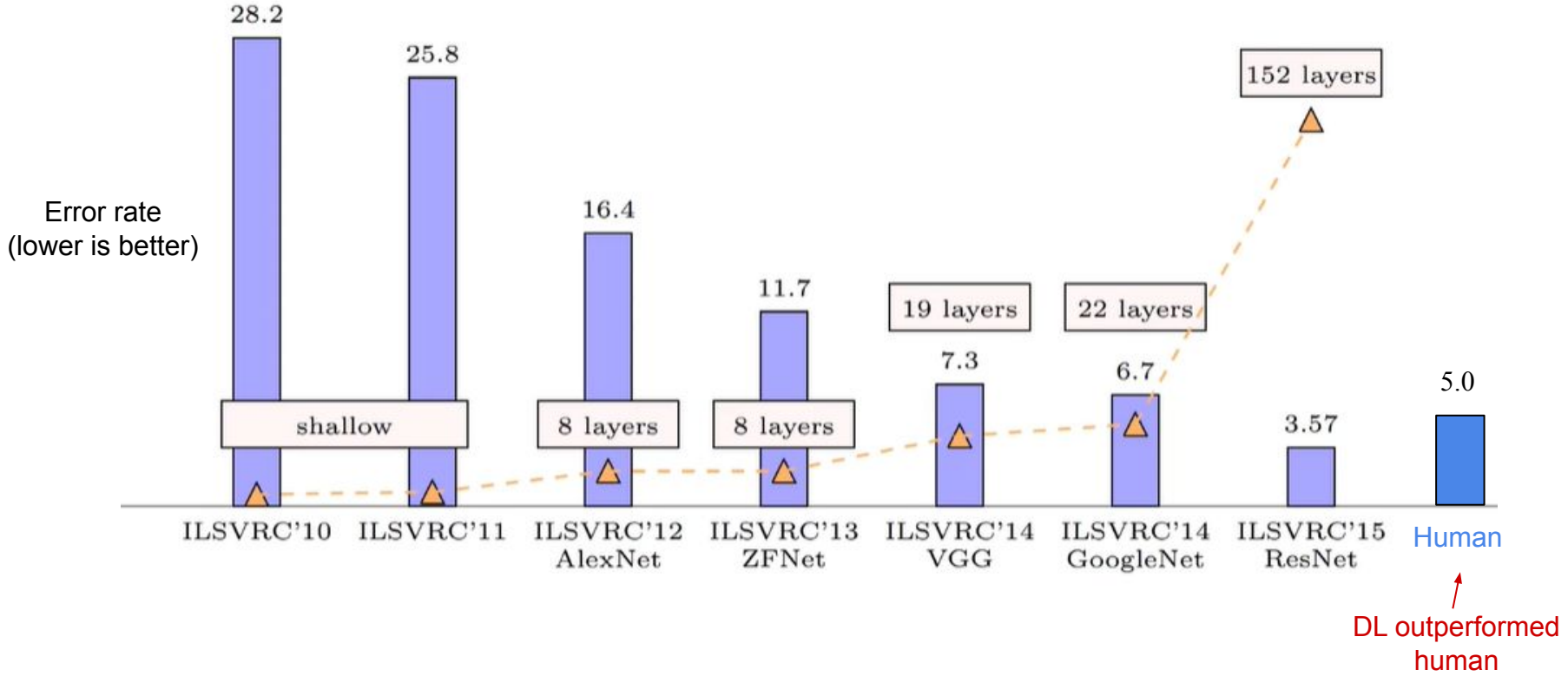
How Deep Learning Performed on the ImageNet Contest?



How Deep Learning Performed on the ImageNet Contest?



How Deep Learning Performed on the ImageNet Contest?

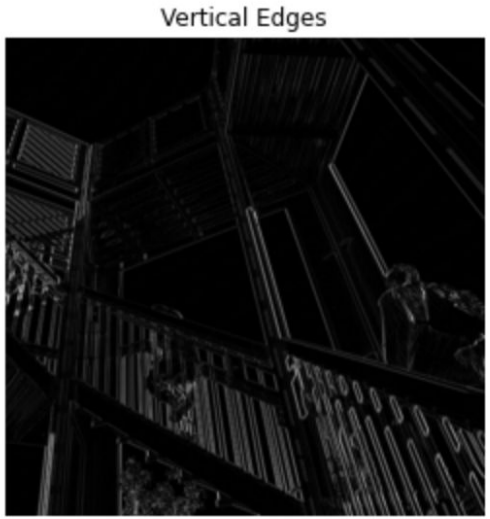


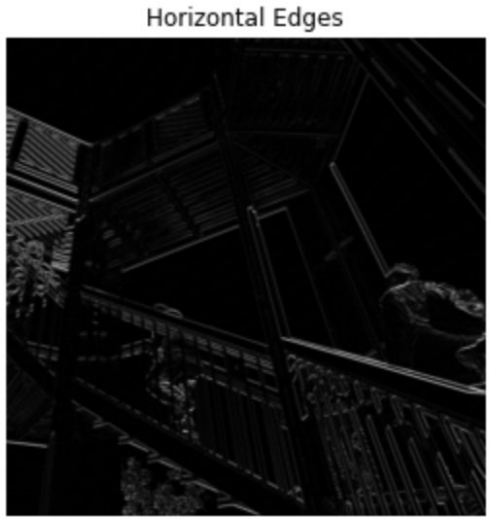
Convolutional Neural Networks

- Our goal will be to introduce a modern and very powerful deep neural network architecture: the convolutional neural network (CNN).
    - CNNs are state-of-the-art across image processing tasks
    - They are loosely motivated by the structure of the visual cortex
    - They are based on two operations: **convolution** and **pooling**

Motivation: Edge Detection

- Consider the problem of detecting edges in an image.

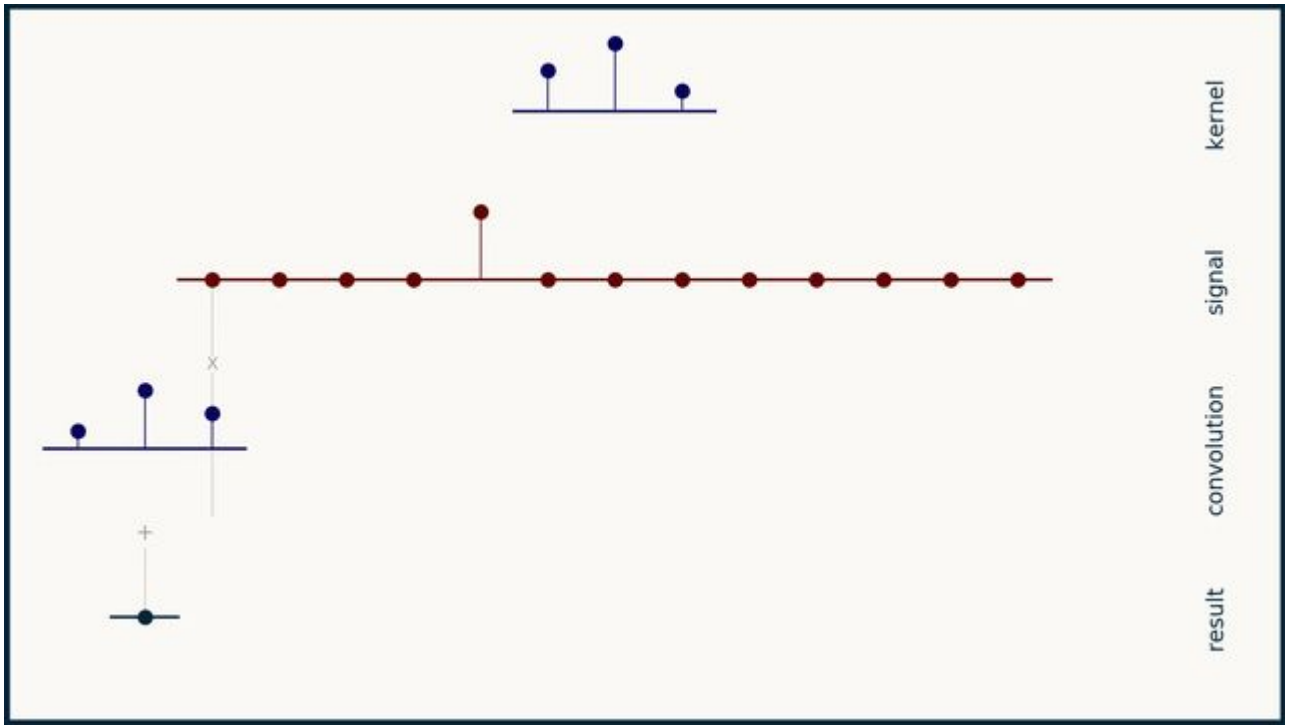




- This can be useful for detecting or classifying objects in a scene.

Convolution: Intuition

- Here, we convolve a 1x3 *filter* with a 1x13 *signal* to obtain an *activation map*.



- Here, we convolve a 1x3 filter with a 1x13 signal to obtain an activation map.

Convolution: definition

Let $f \in \mathbb{R}^p$ and $x \in \mathbb{R}^d$ be two vectors, called the *filter* (or *kernel*) and the *signal* respectively. Typically, $p < d$.

In deep learning, a convolution $(f * x) \in \mathbb{R}^d$ is typically defined as

$$(f * x)[i] \triangleq \underbrace{\sum_{j=1}^{p} f[j]x[i+j]}_{\text{dot product of } f \text{ with part of } g} \tag{1}$$

where $x[j] = 0$ when $j \leq 0$ or $j > d$.

- We may think of the convolution

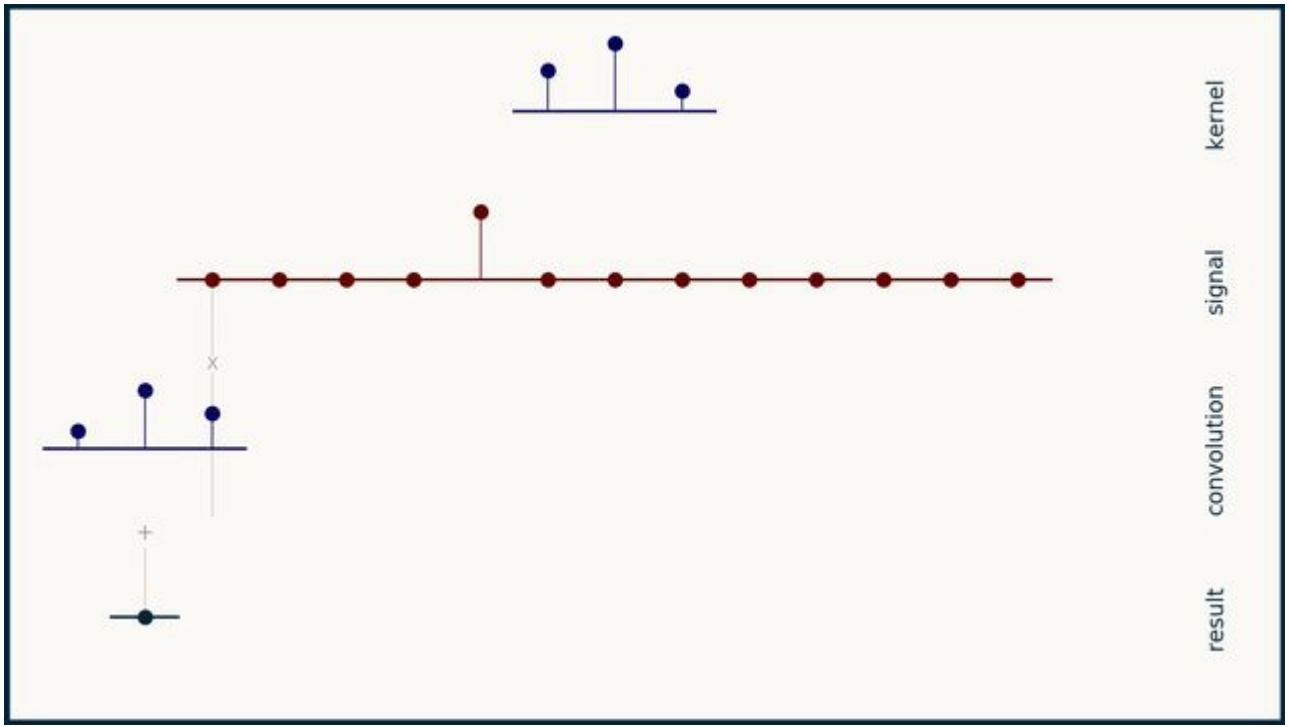
  $$\sum_{j=1}^{p} f[j]x[i+j]$$

  as a "local" dot product of `f` with `x[i:i+p]` (the subsequence of x that starts at position i).

- $(f * x)[i]$ is large when f is similar to x at i and small otherwise. It measures the extent to which f is "found" in x at i.

Convolution: Intuition

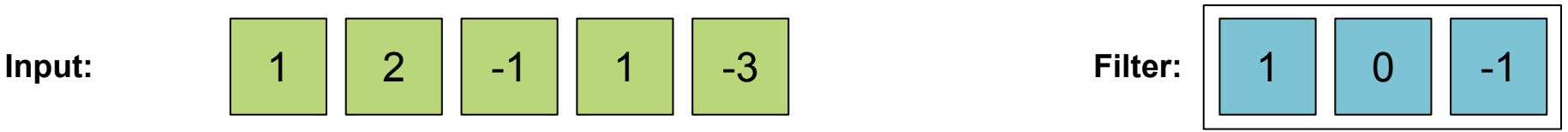
- Here, we convolve a 1x3 *filter* with a 1x13 *signal* to obtain an *activation map*.



- Notice how the output has large values at locations where the dot product between the kernel and the signal is large.

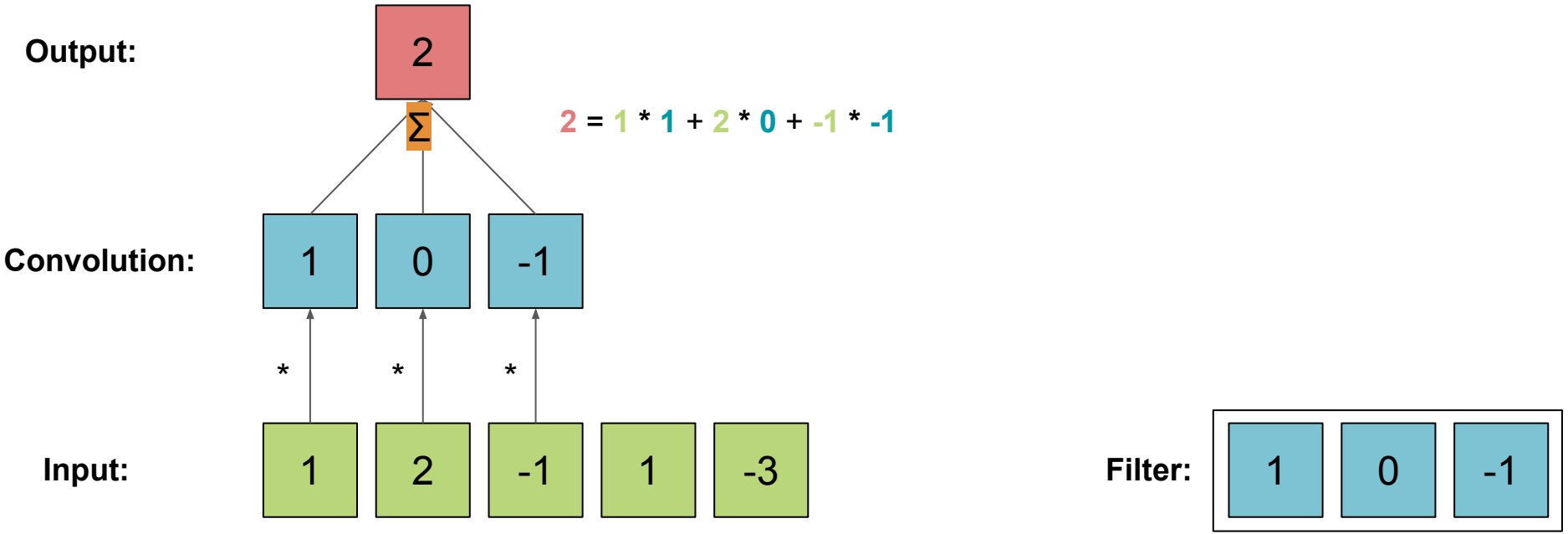
An example of 1D convolution

$$(f * x)[i] \triangleq \sum_{j=1}^{p} f[j]x[i+j]$$



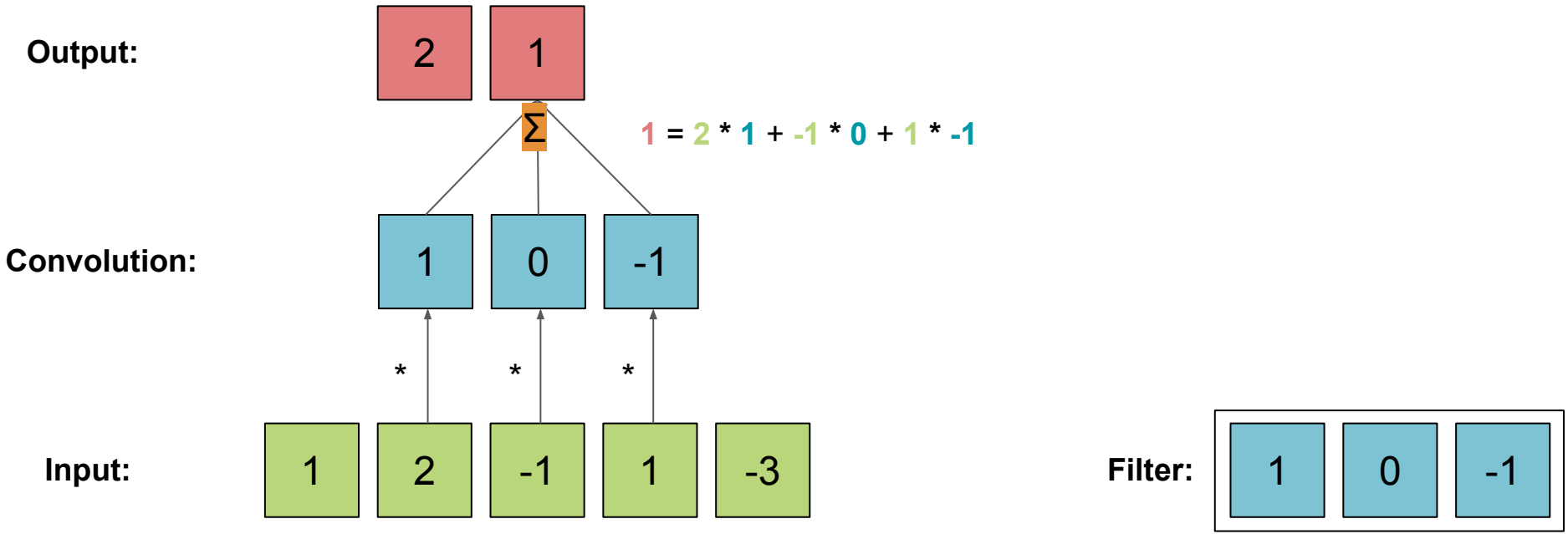
An example of 1D convolution

$$(f * x)[i] \triangleq \sum_{j=1}^{p} f[j]x[i+j]$$



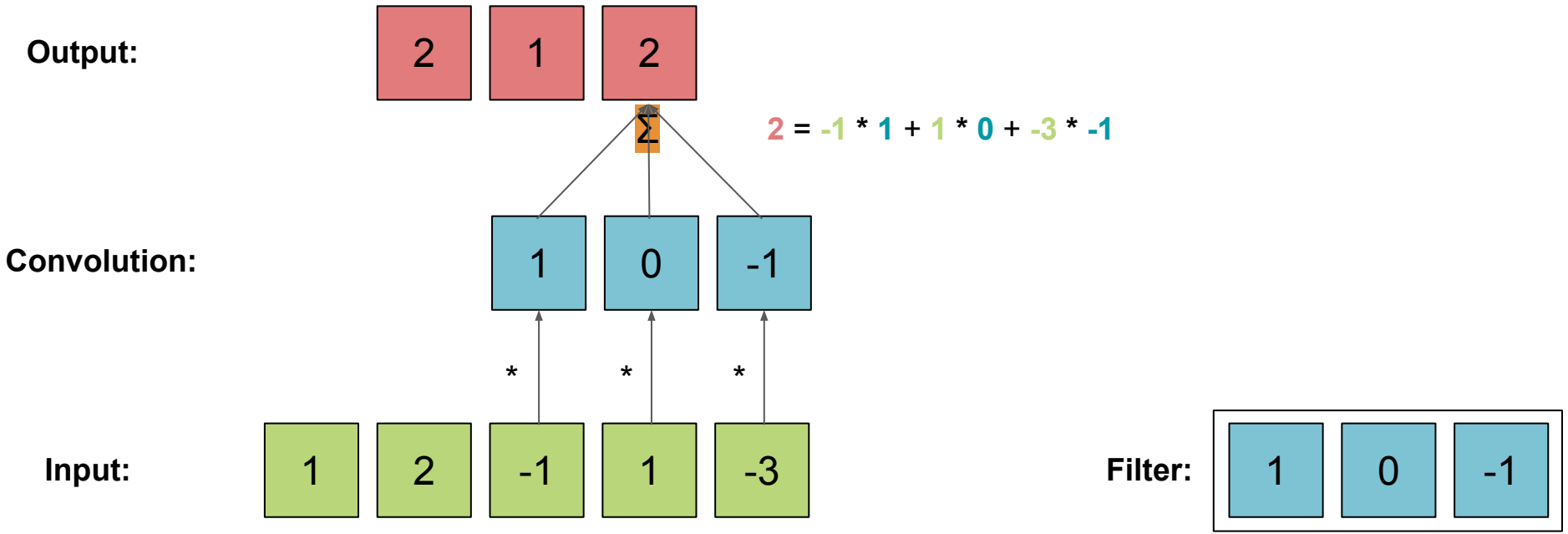
An example of 1D convolution

$$(f * x)[i] \triangleq \sum_{j=1}^{p} f[j]x[i+j]$$

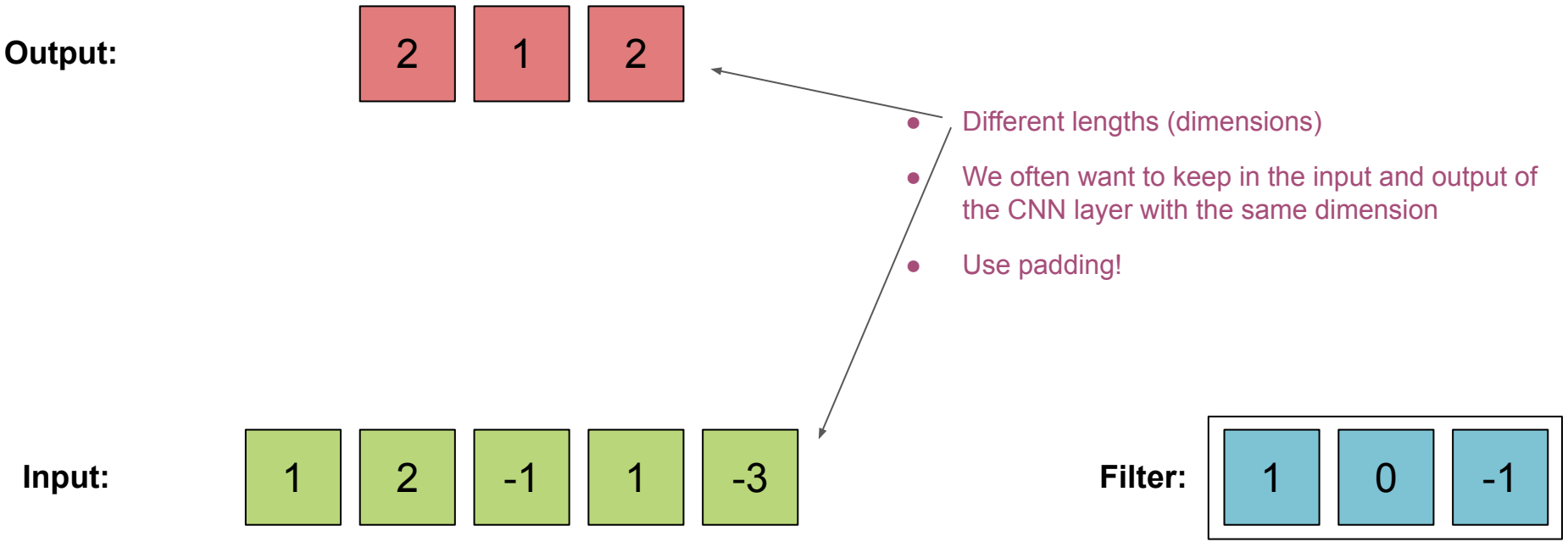


An example of 1D convolution

$$(f * x)[i] \triangleq \sum_{j=1}^{p} f[j]x[i+j]$$

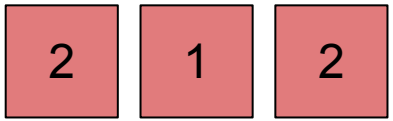


An example of 1D convolution



An example of 1D convolution

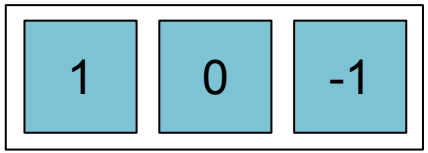
Output:



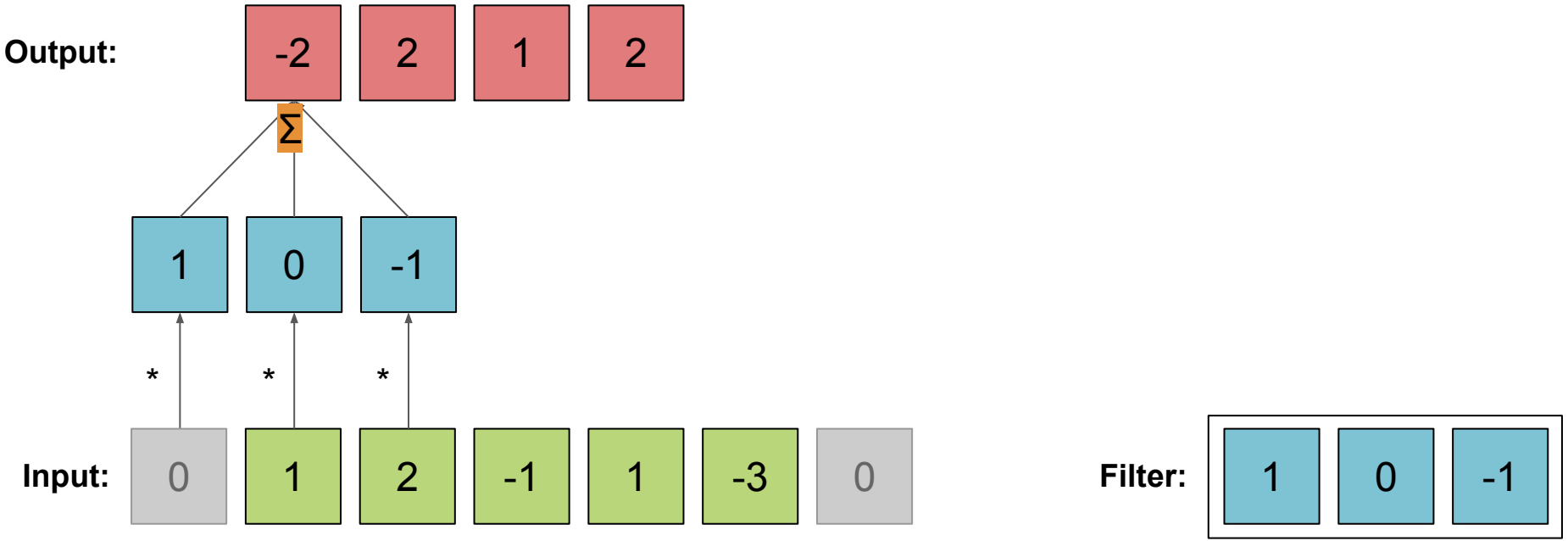
- Different lengths (dimensions)
- We often want to keep in the input and output of the CNN layer with the same dimension
- **Use padding!**

Input:

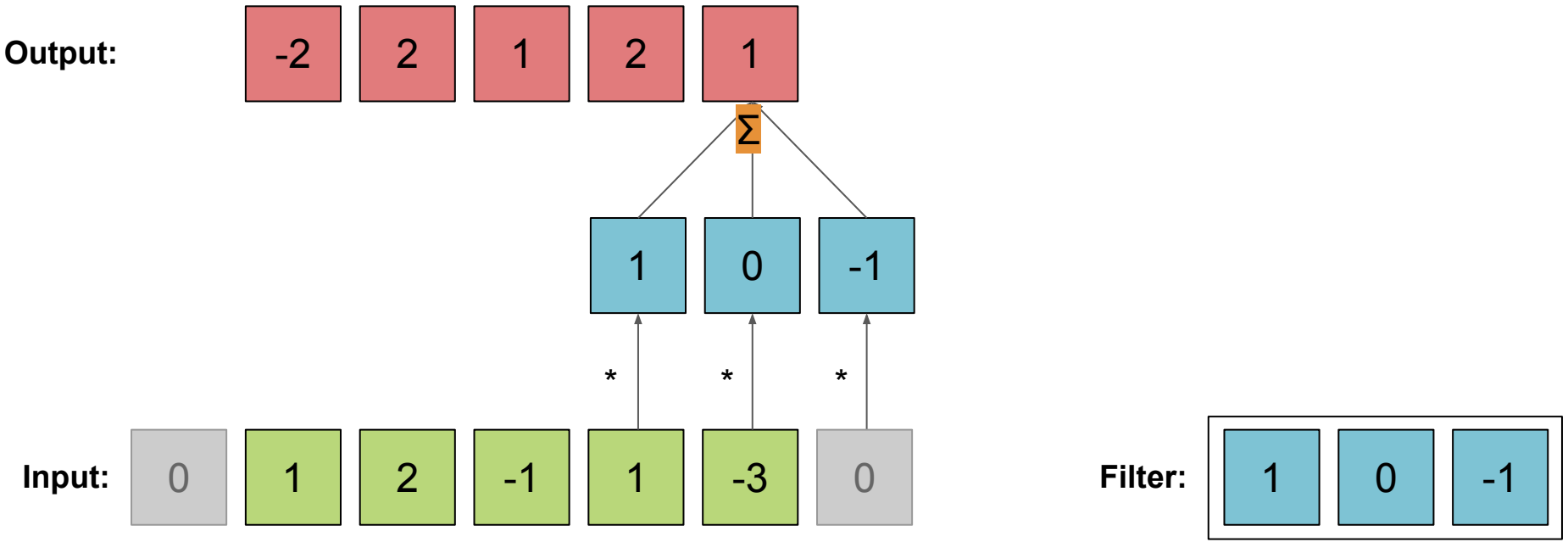
Filter:



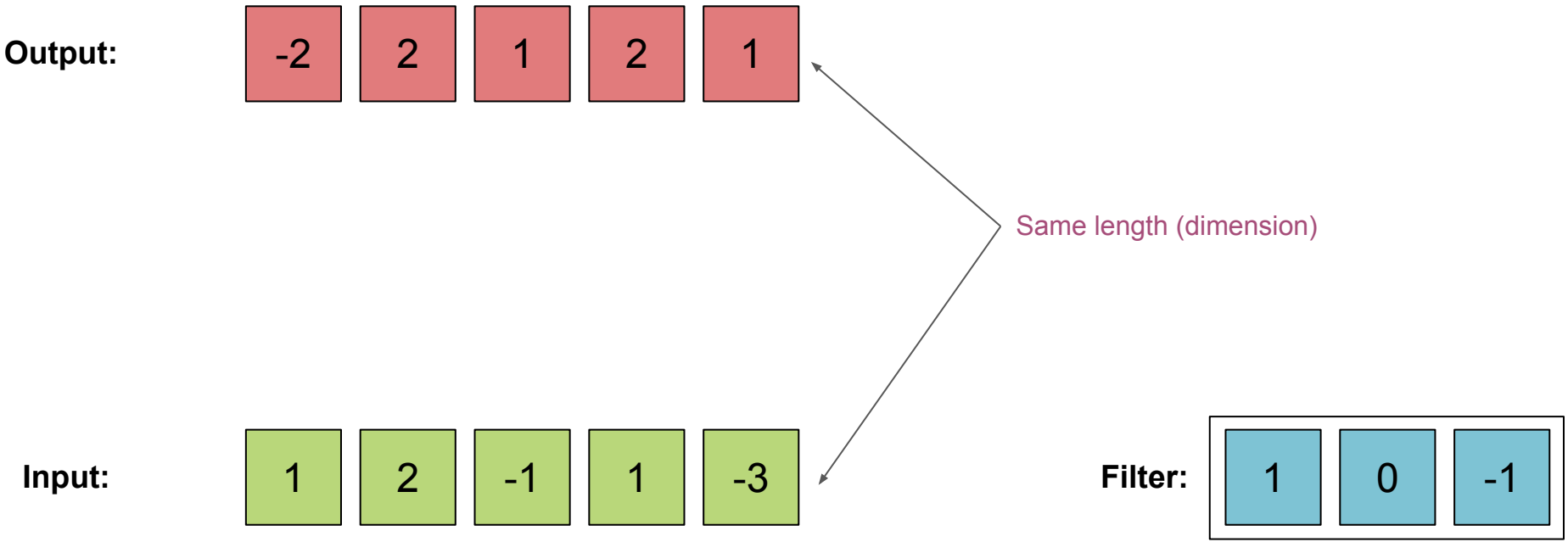
An example of 1D convolution



An example of 1D convolution



An example of 1D convolution



Convolution on 2D data

x

f_1 f_l f_{L-1} f_L

Cat

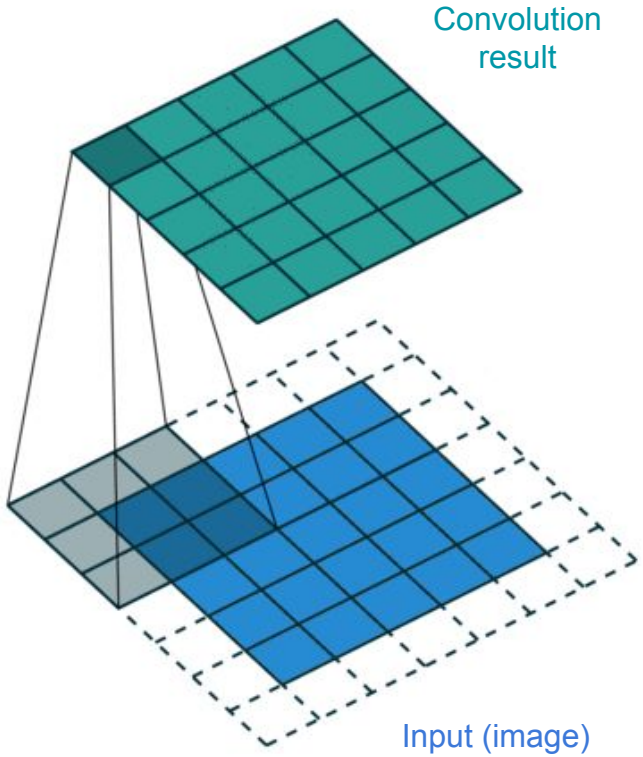
Need to represent an image in two dimensions

2D Convolutional Layers

In this example, f is a 3x3 array (gray area). x is a 5x5 array (blue area). The result of the convolution is the top green area.

In the context of neural networks, f are learnable weights, x are inputs from previous layers.

- The signal x can be an image.
- The filter f can be a "feature" of the image, like an edge or a more complex object.



2D Convolutional Layers

Input image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

Filter (kernel)

0	0	0
0	1	1
0	1	0

Output

2D Convolutional Layers

Input image

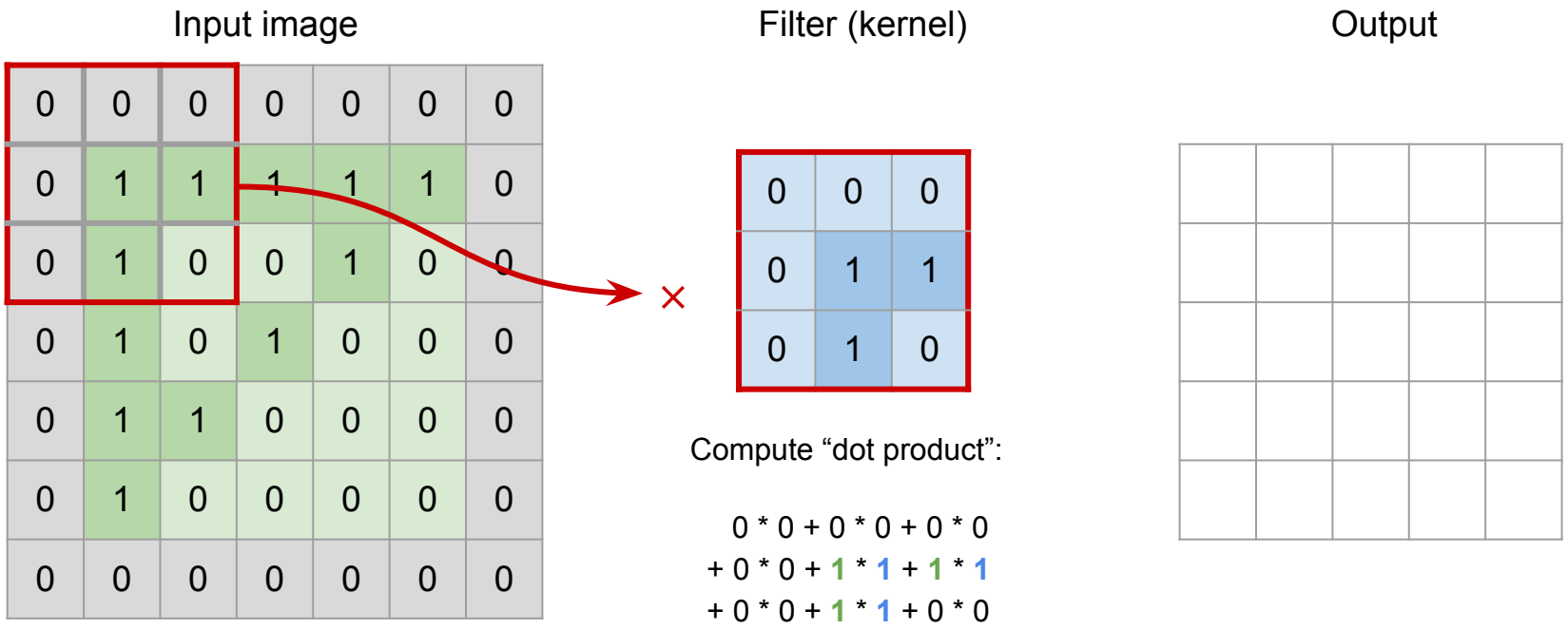
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Filter (kernel)

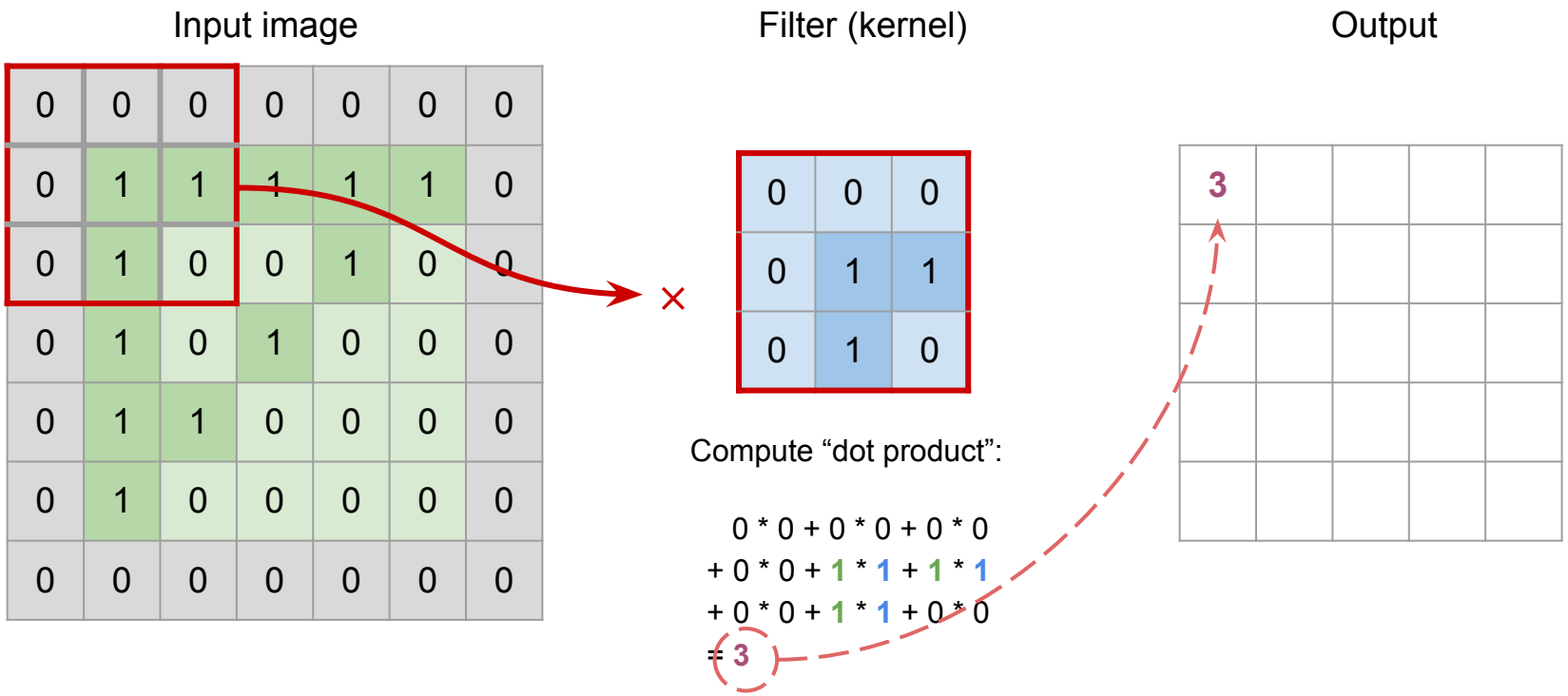
0	0	0
0	1	1
0	1	0

Output

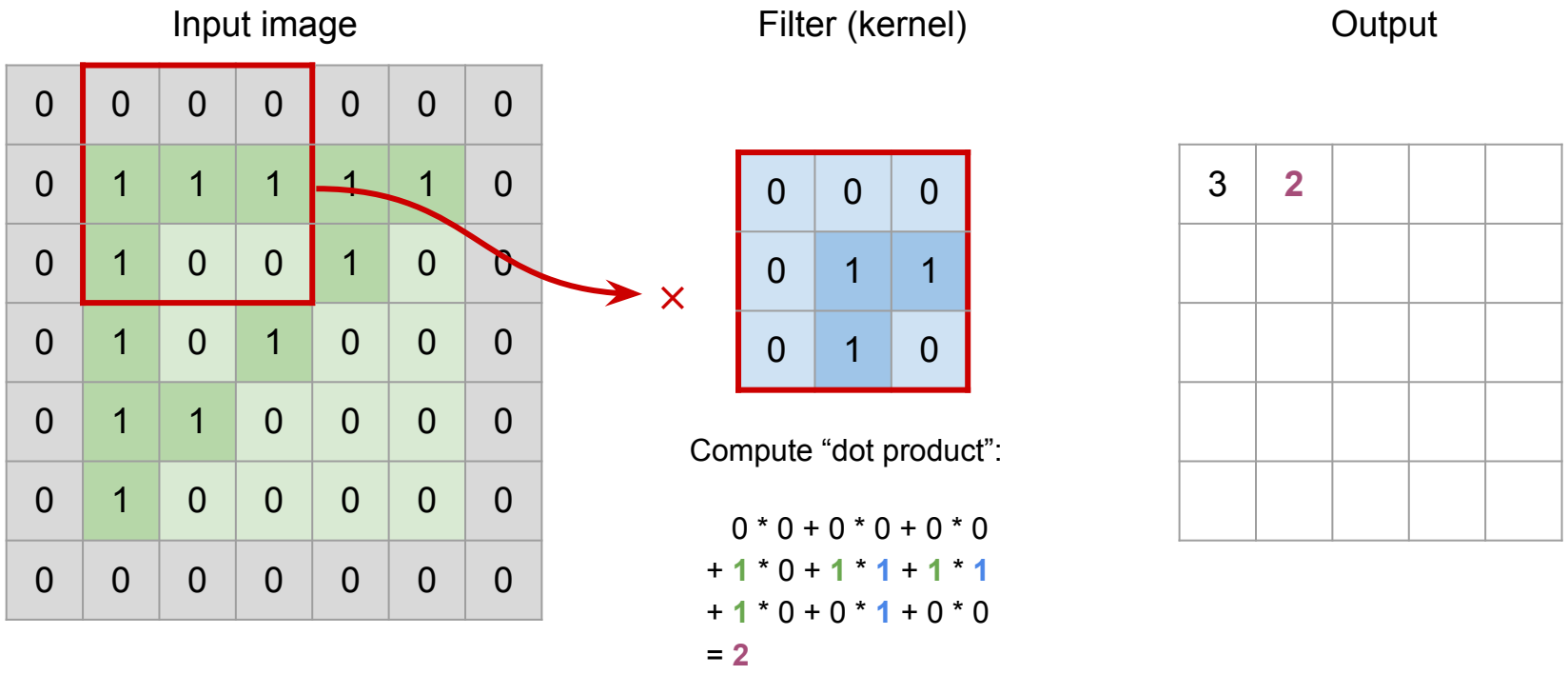
2D Convolutional Layers



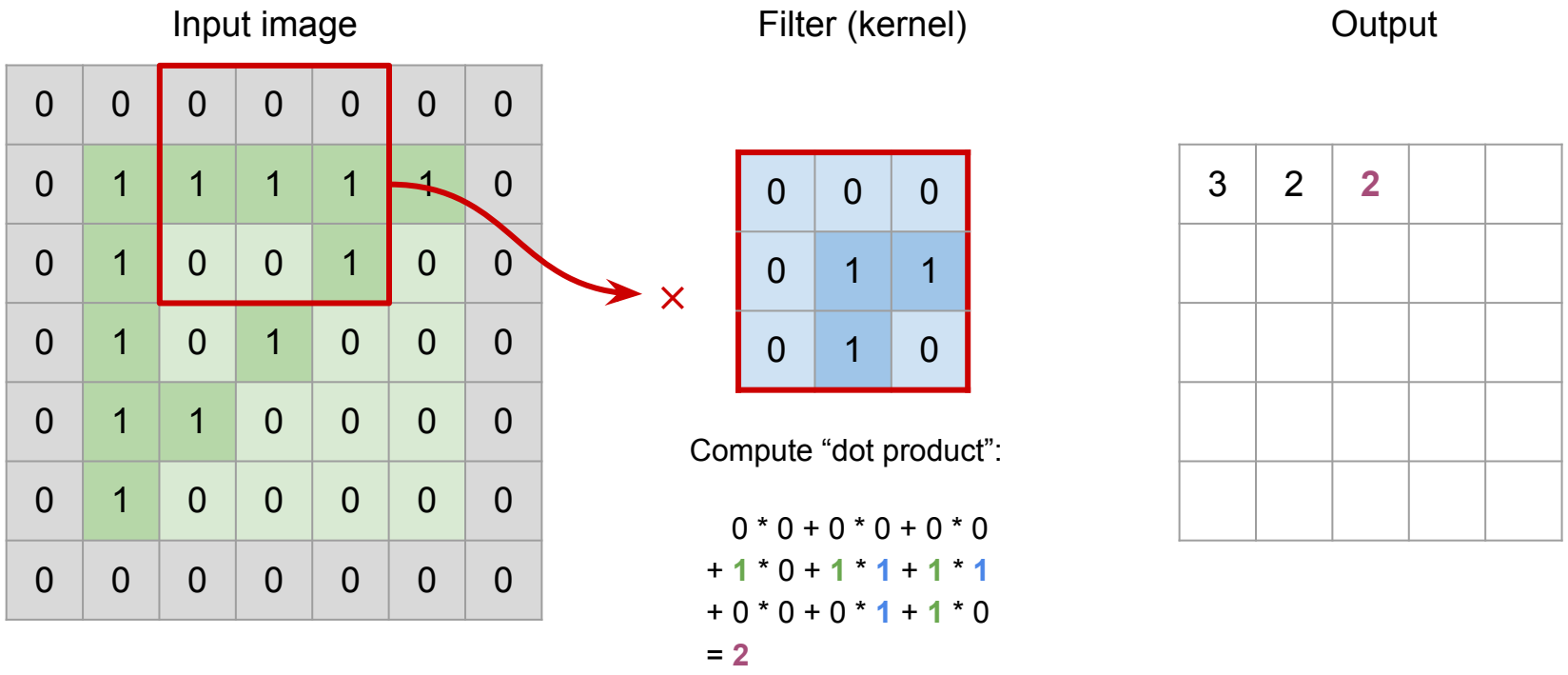
2D Convolutional Layers



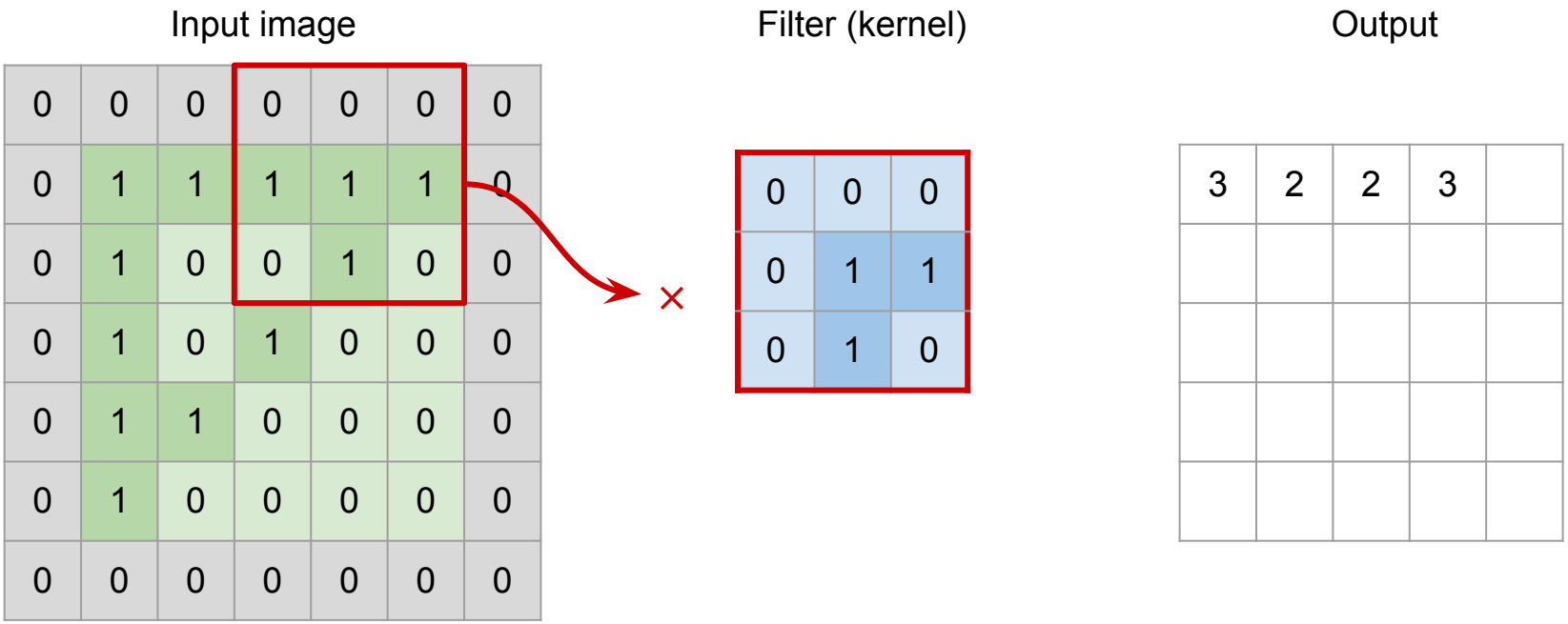
2D Convolutional Layers



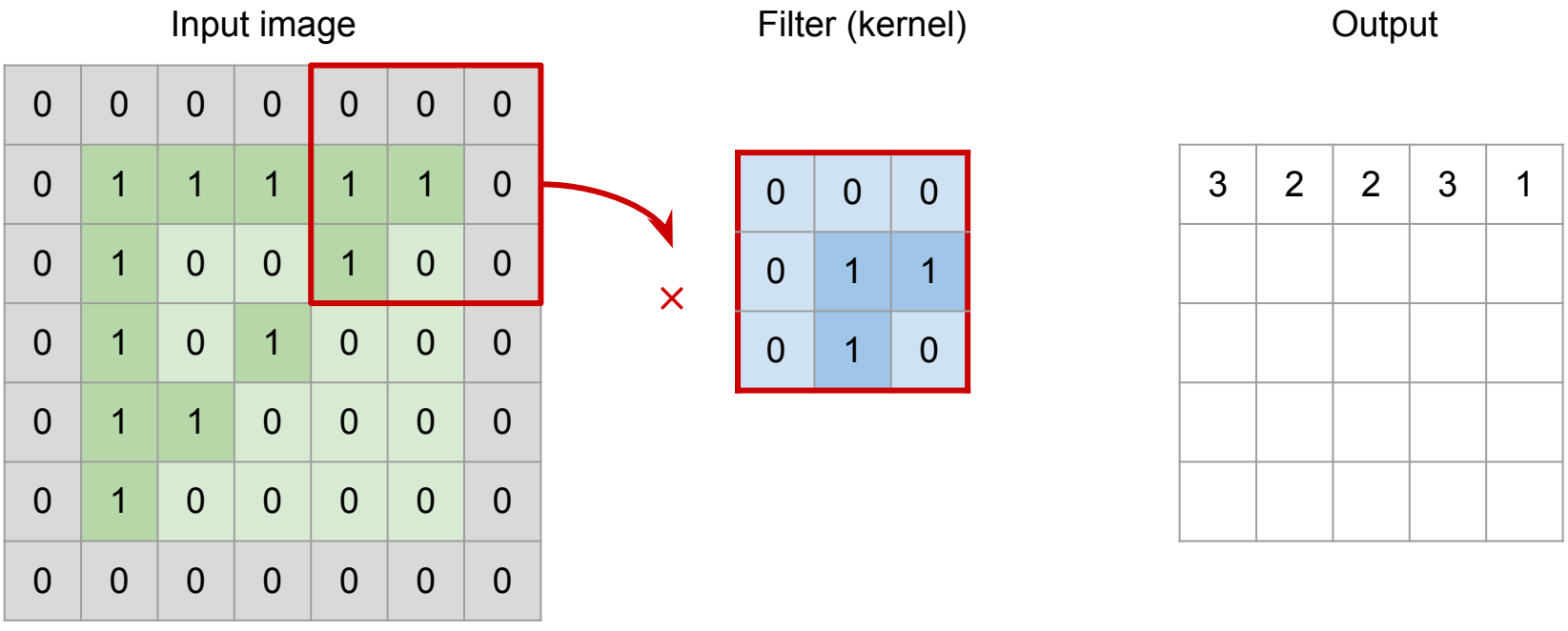
2D Convolutional Layers



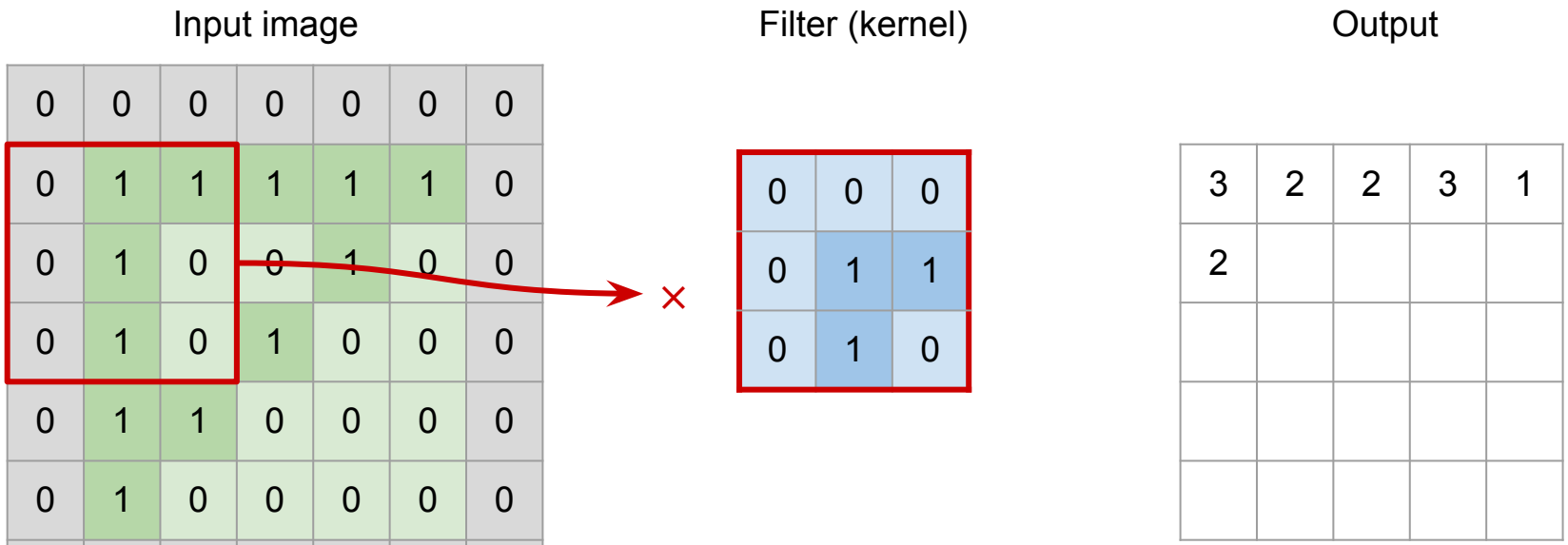
2D Convolutional Layers



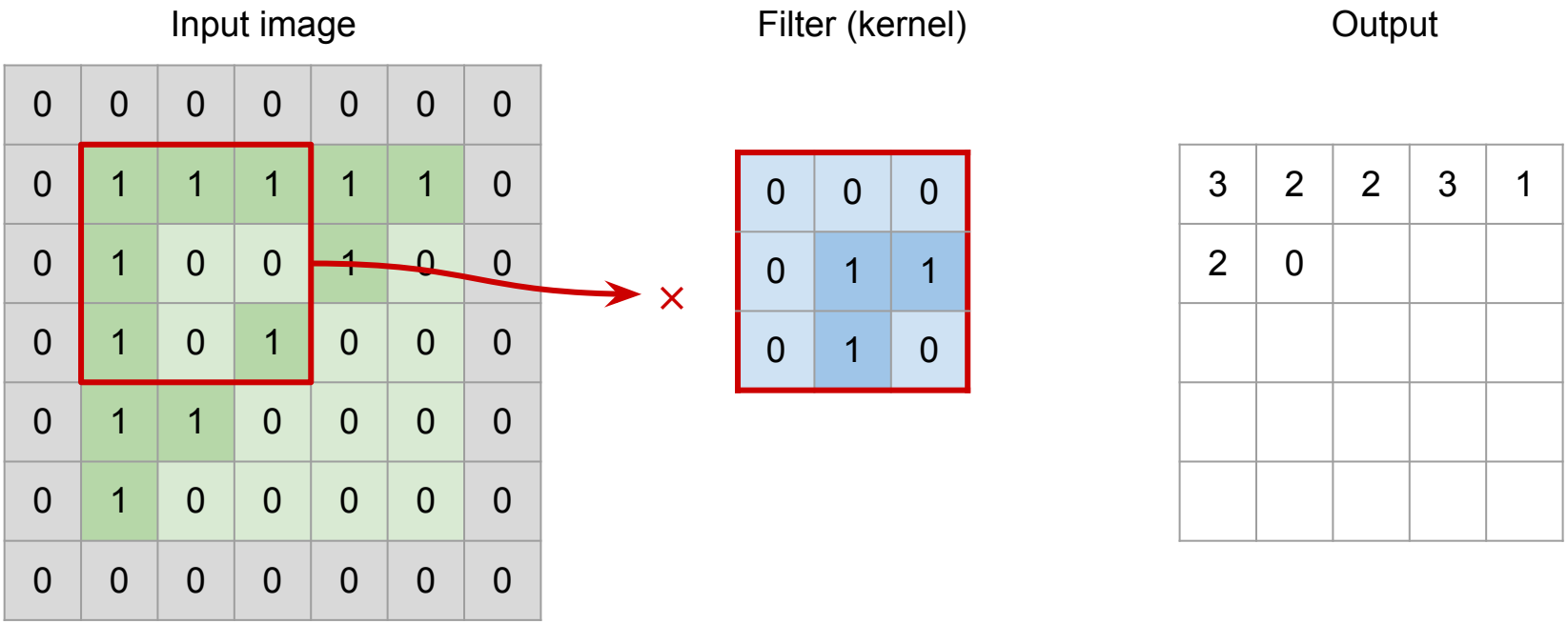
2D Convolutional Layers



2D Convolutional Layers



2D Convolutional Layers



2D Convolutional Layers

Input image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

×

Filter (kernel)

0	0	0
0	1	1
0	1	0

Output

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

2D Convolutional Layers

Input image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

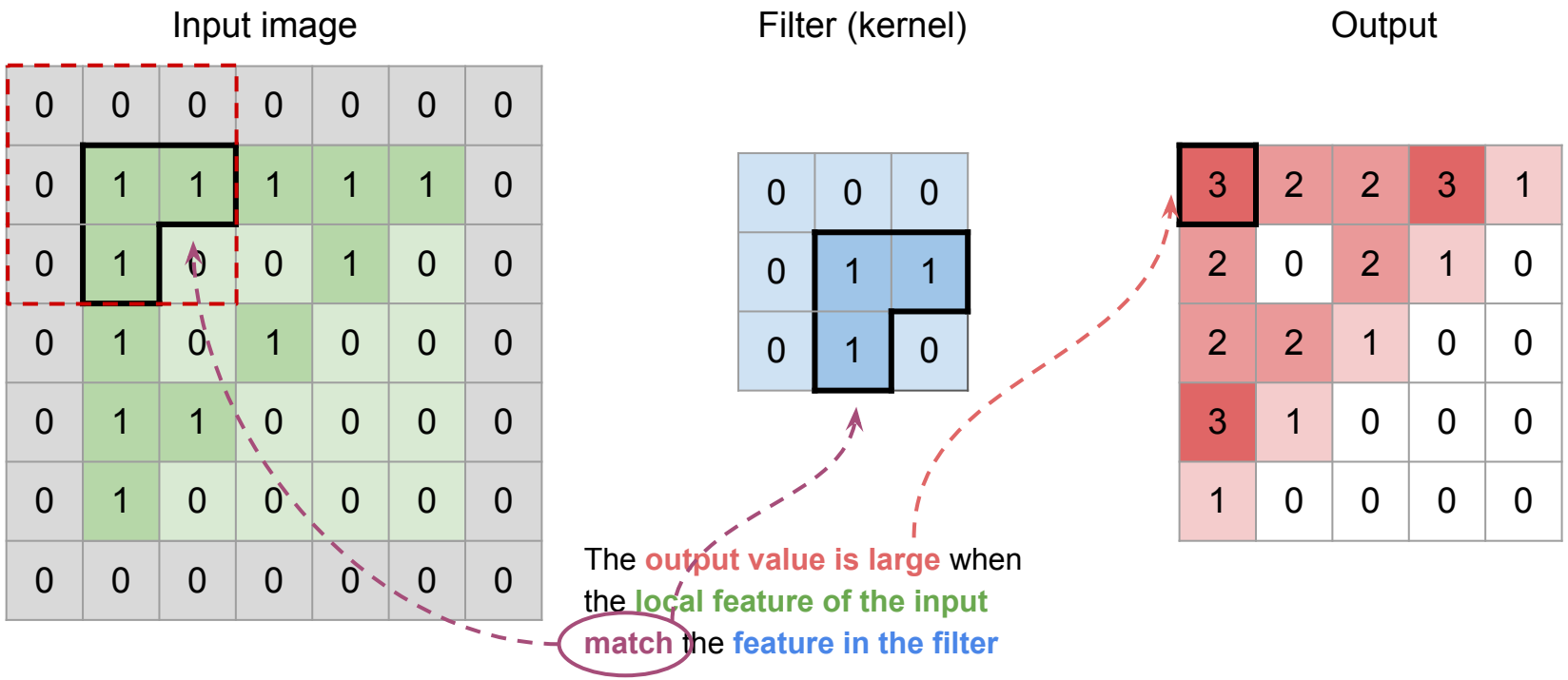
Filter (kernel)

0	0	0
0	1	1
0	1	0

Output

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

2D Convolutional Layers



2D Convolutional Layers

Input image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Filter (kernel)

0	0	0
0	1	1
0	1	0

Output

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

The **output value is large** when the **local feature of the input match** the **feature in the filter**

2D Convolutional Layers

Input image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Filter (kernel)

0	0	0
0	1	1
0	1	0

Output

3	2	2	3	1
2	0	2	1	0
2	2	1	0	0
3	1	0	0	0
1	0	0	0	0

The **output value is large** when the **local feature of the input match** the **feature in the filter**

2D Convolutional Layers

Input image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

Filter (kernel)

0	0	0
0	1	1
0	1	0

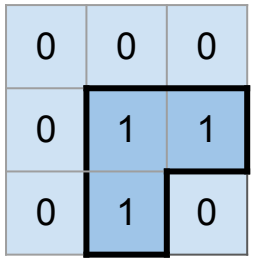
We can multiple filters to capture different patterns in the input

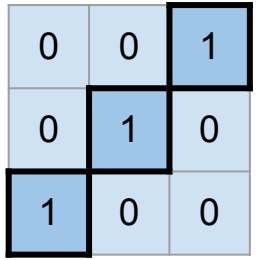
2D Convolutional Layers

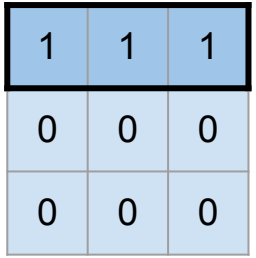
Input image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

Filter (kernel)







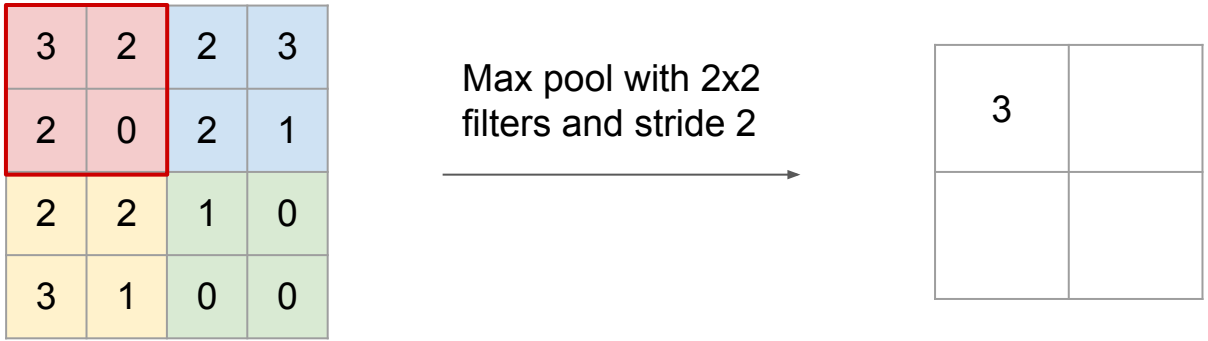
We can multiple filters to capture different patterns in the input

- In a CNN, we do not have to manually design those filters.
- The CNN will automatically learned those filters from data

Pooling layer

- We have introduced convolutional layers. Next, we introduce another important component in a CNN -- **pooling layer**

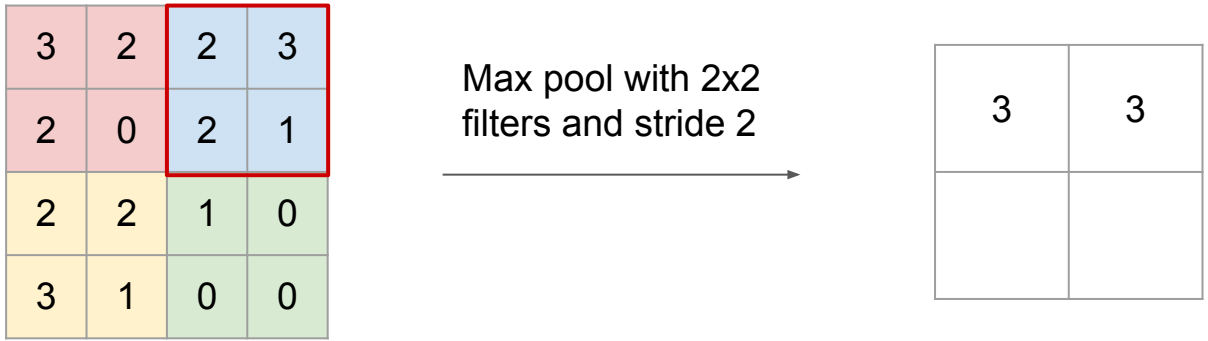
- Example: Max Pooling



Pooling layer

- We have introduced convolutional layers. Next, we introduce another important component in a CNN -- **pooling layer**

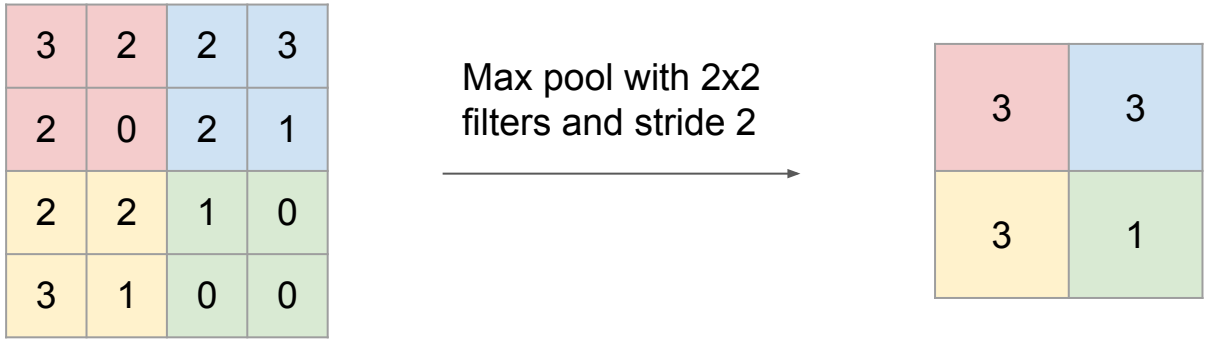
- Example: Max Pooling



Pooling layer

- We have introduced convolutional layers. Next, we introduce another important component in a CNN -- **pooling layer**

- Example: **Max** Pooling



Pooling layer

- We have introduced convolutional layers. Next, we introduce another important component in a CNN -- **pooling layer**

- Example: Average Pooling

3	2	2	3
2	0	2	1
2	2	1	0
3	1	0	0

Average pool with 2x2 filters and stride 2

7/4	2
2	1/4

Pooling layer

A pooling layer is a model $f : \mathbb{R}^d \to \mathbb{R}^p$ that applies pooling operations to an input x.

$$f(x) = \text{pooling}(x)$$

where $pooling$ is a pre-defined operation applied over the input x.

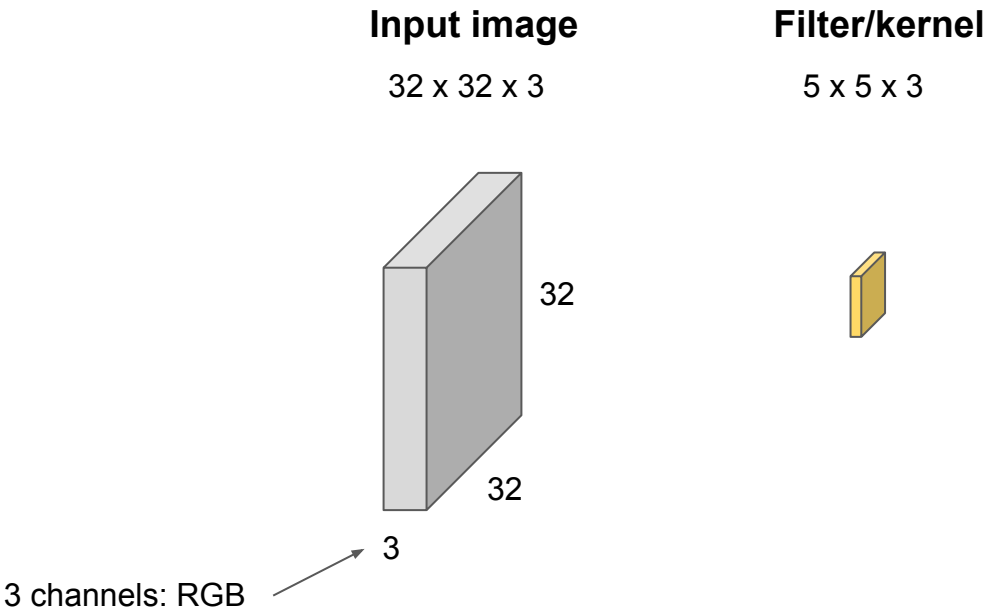
Pooling layer does not have learnable weights.

Purpose of Pooling:

- Allows slight translation invariance inside the pooling window: the exact location of the relevant feature is not critical. This improves feature detection.
- When used together with convolutions, pooling reduces spatial size of feature maps to capture more abstract representations.

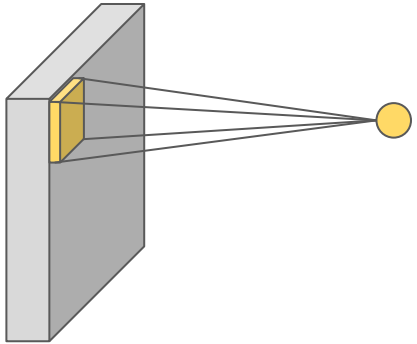
Convolutional Neural Networks

Applying Convolution on an RGB Image



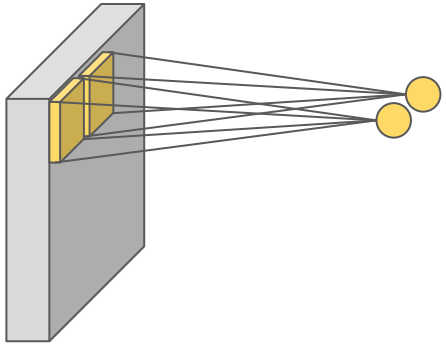
Convolution: slide the filter over the image spatially and compute the dot products

Convolution layer



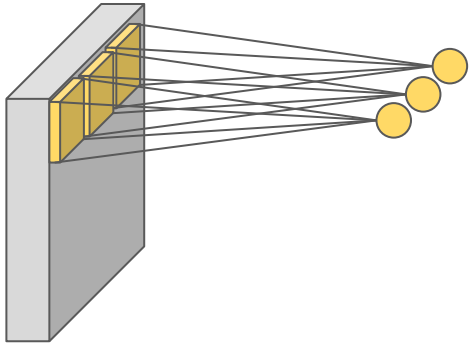
Adapted from CS231N, Stanford

Convolution layer



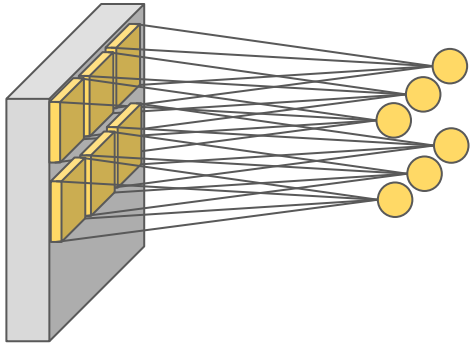
Adapted from CS231N, Stanford

Convolution layer



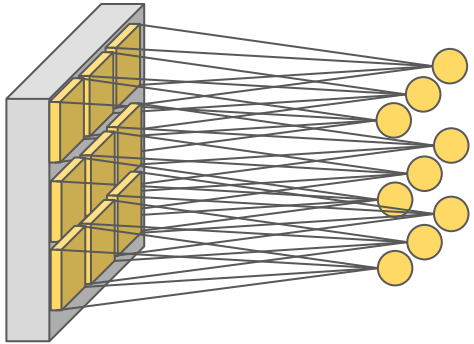
Adapted from CS231N, Stanford

Convolution layer



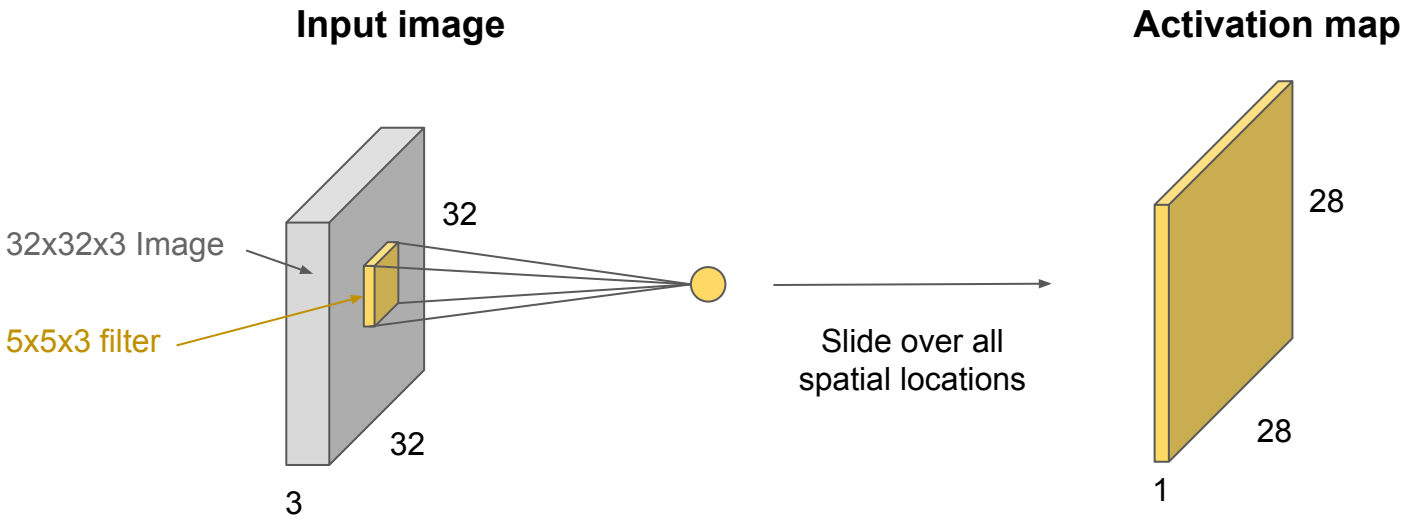
Adapted from CS231N, Stanford

Convolution layer



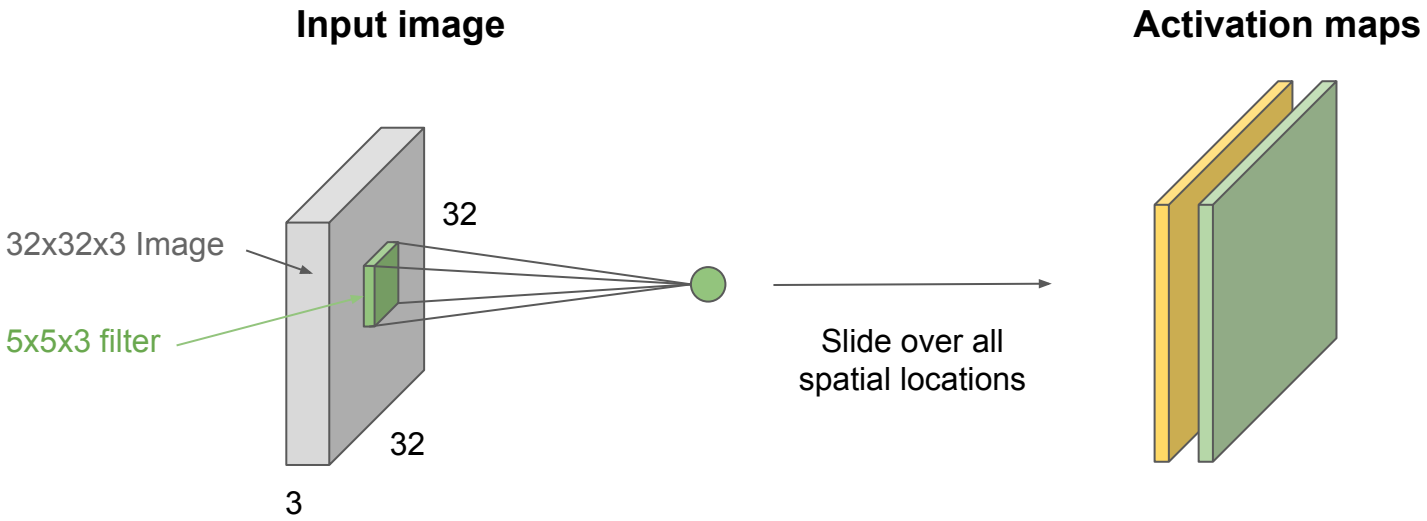
Adapted from CS231N, Stanford

Convolution layer



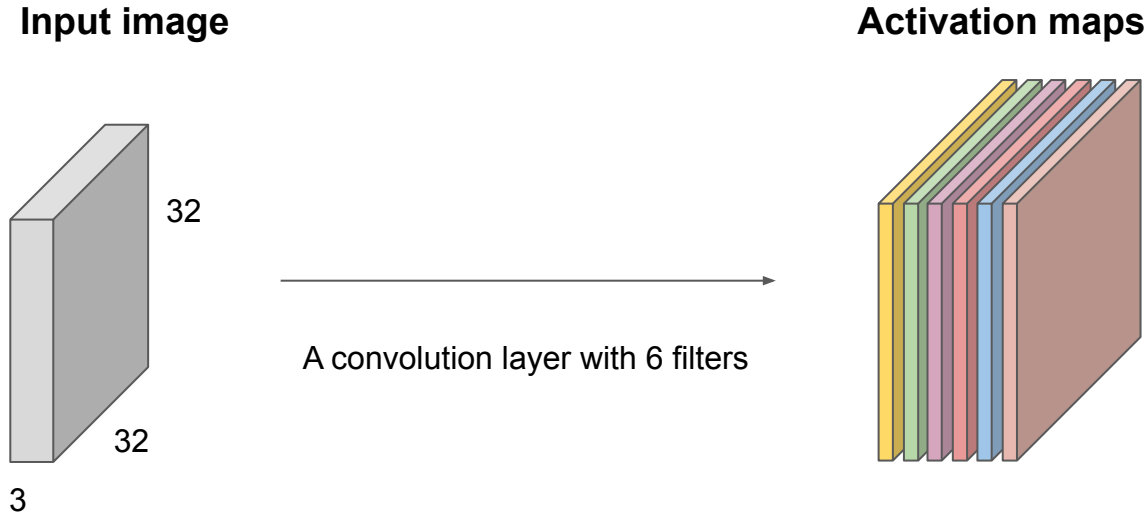
Adapted from CS231N, Stanford

Convolution layer



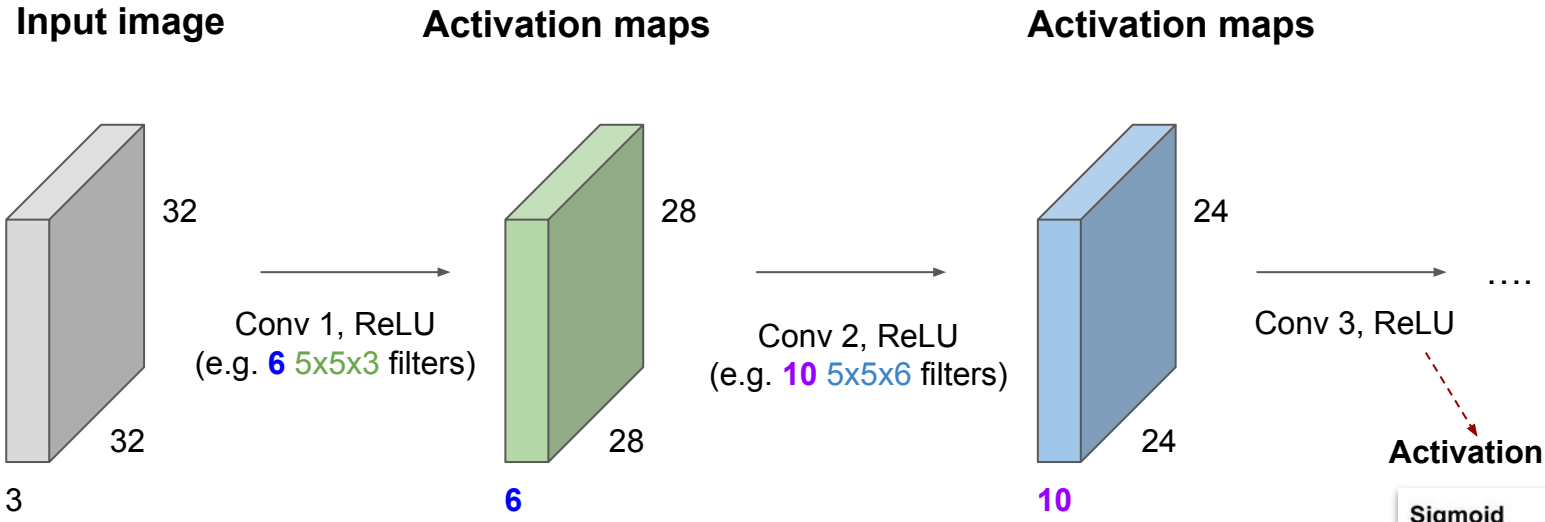
Add a second filter

Convolution layer



Adapted from CS231N, Stanford

CNN is a sequence of convolution layers, interspersed with activation functions



Adapted from CS231N, Stanford

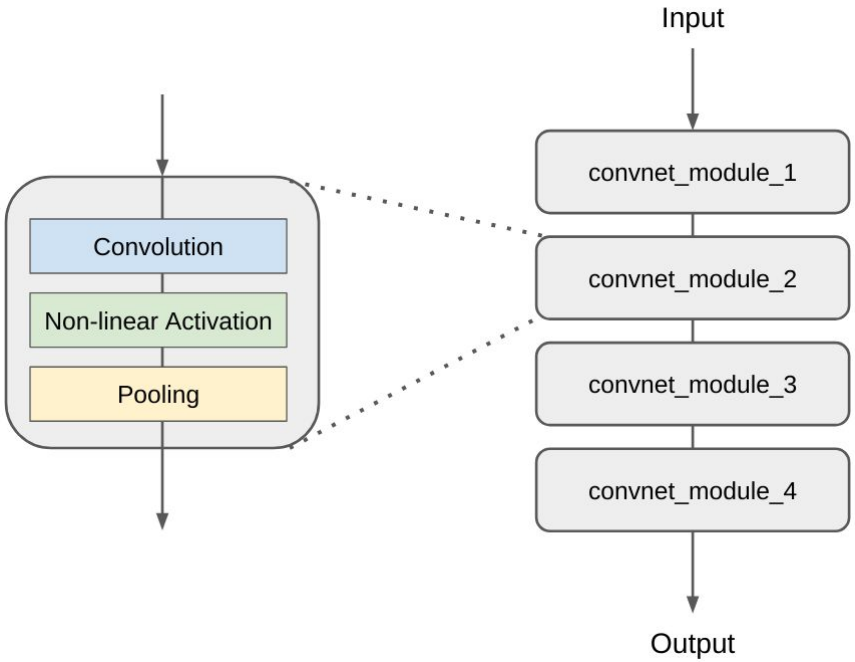
Convolutional Neural Networks

A convolutional neural network (CNN) is a model $f : \mathbb{R} \to \mathbb{R}$ that consists of a composition of L neural network layers that contain convolutions:

$$f(x) = f_L \circ f_{L-1} \circ \dots f_1(x).$$

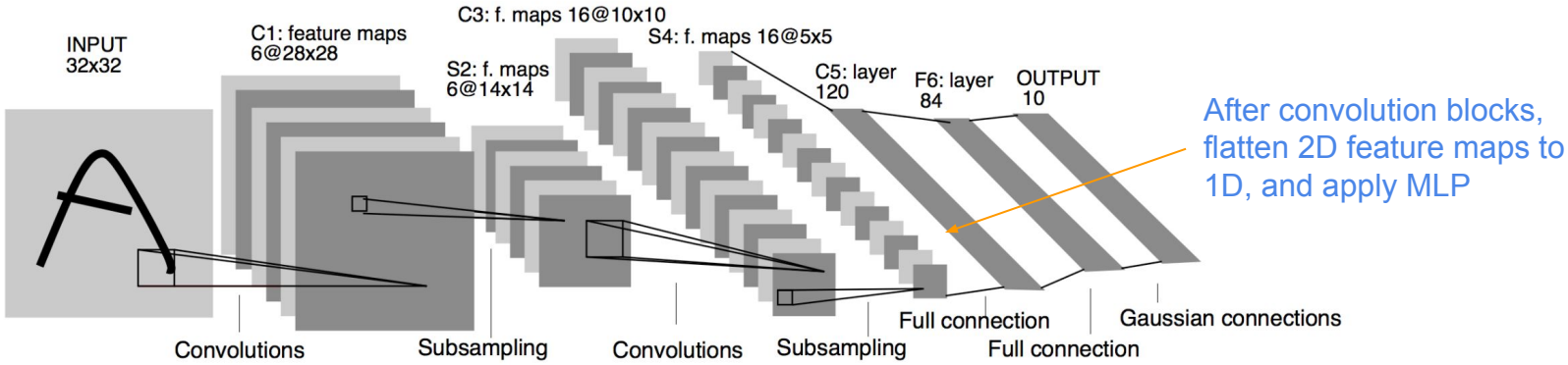
The final f_L is often a fully connected output later of size one.

Typically, CNNs are made of consecutive convolution + activation + pooling layers that form into blocks.



Example 1: LeNet for MNIST Digits Recognition

- LeNet successfully used CNNs for digit recognition [LeCun, Bottou, Bengio, Haffner 1998].



- LeNet contains multiple convolutional and max-pooling (called subsampling in the original paper) layers and a fully connected MLP in the end.
- Given an image of a handwritten digit, LeNet predicts it to be one of the 10 classes (0-9).
- This work inspired many subsequent CNN architectures.

CNN in PyTorch

Layers definition

```
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

CNN in PyTorch

Conv layer

Pooling layer

Fully-connected (linear) layer

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

CNN in PyTorch

Function to call when running the layer

```
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

CNN in PyTorch

Conv -> ReLU -> MaxPool

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

CNN in PyTorch

The second conv block

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

CNN in PyTorch

Flatten to 1D vectors


```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

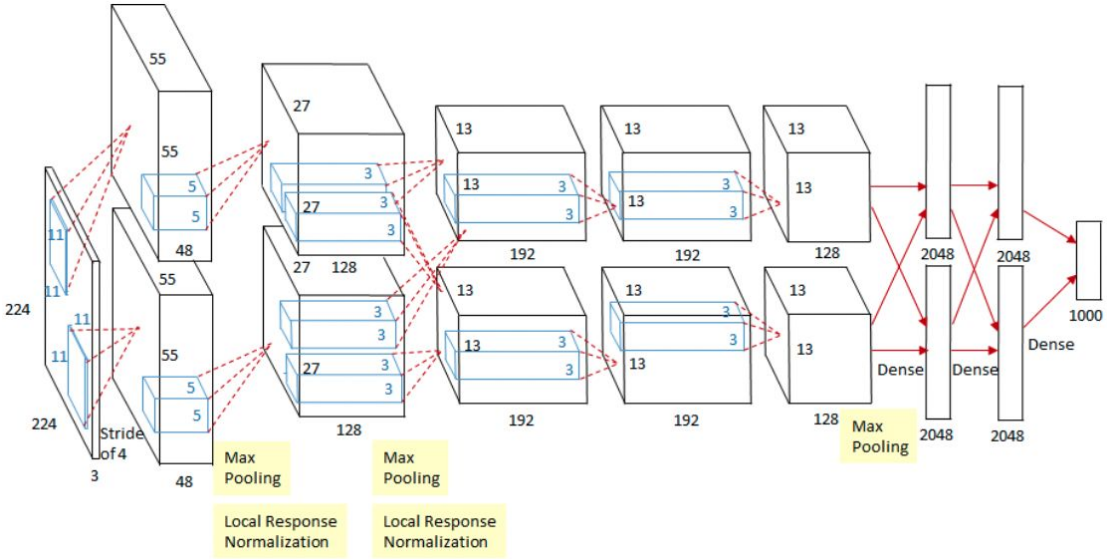
CNN in PyTorch

Fully-connected neural network for classification

```
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

Example 2: AlexNet for ImageNet

- ImageNet classification with deep convolutional neural networks [Krizhevsky, Sutskever, Hinton 2012]



- AlexNet contains multiple convolutional layers followed by max pooling. The prediction is 1000 dimensional because that is the number of classes in the ImageNet classification task.
- Note that in the original implementation the image is separated into two parts that are in two different GPUs due to hardware limits.

Example 3: ResNet

- Winner of the ImageNet Competition 2015
- Default architecture for many CNN models
- The “residual link” idea is widely used
- The most cited neural network of the 21st century

Data as of 2024/02/06

Deep residual learning for **image recognition**

K He, X Zhang, S Ren, J Sun - … and **pattern recognition**, 2016 - openaccess.thecvf.com

… **Deeper** neural **networks** are more difficult to train. We present a **residual learning** frame to ease the training of **networks** that are substantially **deeper** than those used previously. …

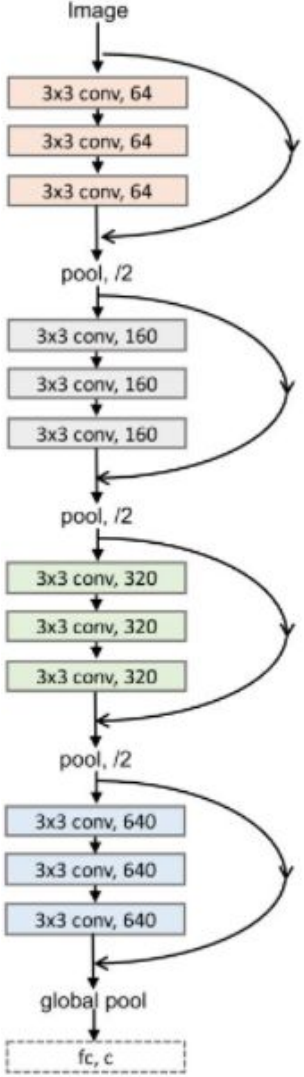
☆ Save Cite Cited by 199144 Related articles All 76 versions Import into BibTeX

This lecture in 2023:

☆ Save Cite Cited by 152947 Related articles All 73 versions »

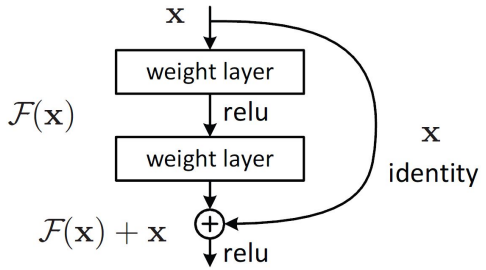
This lecture in 2022:

☆ Save Cite Cited by 103017 Related articles All 63 versions »



Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



Residual link

What have been learned in CNN ? (for image classification)

- CNNs can be seen as extracting features at low-, mid-, and high-levels of abstraction that are relevant to corresponding visual concepts.
- Below, we reproduce **feature visualization** from the paper *Visualizing and Understanding Convolutional Networks* [Zeiler, Fergus 2013].

For a convolutional network with

$$\text{Input} \rightarrow \text{Conv Layer1} \rightarrow \text{Conv Layer2} \rightarrow \cdots \rightarrow \text{Conv Layer5} \rightarrow \cdots \rightarrow \text{Output},$$

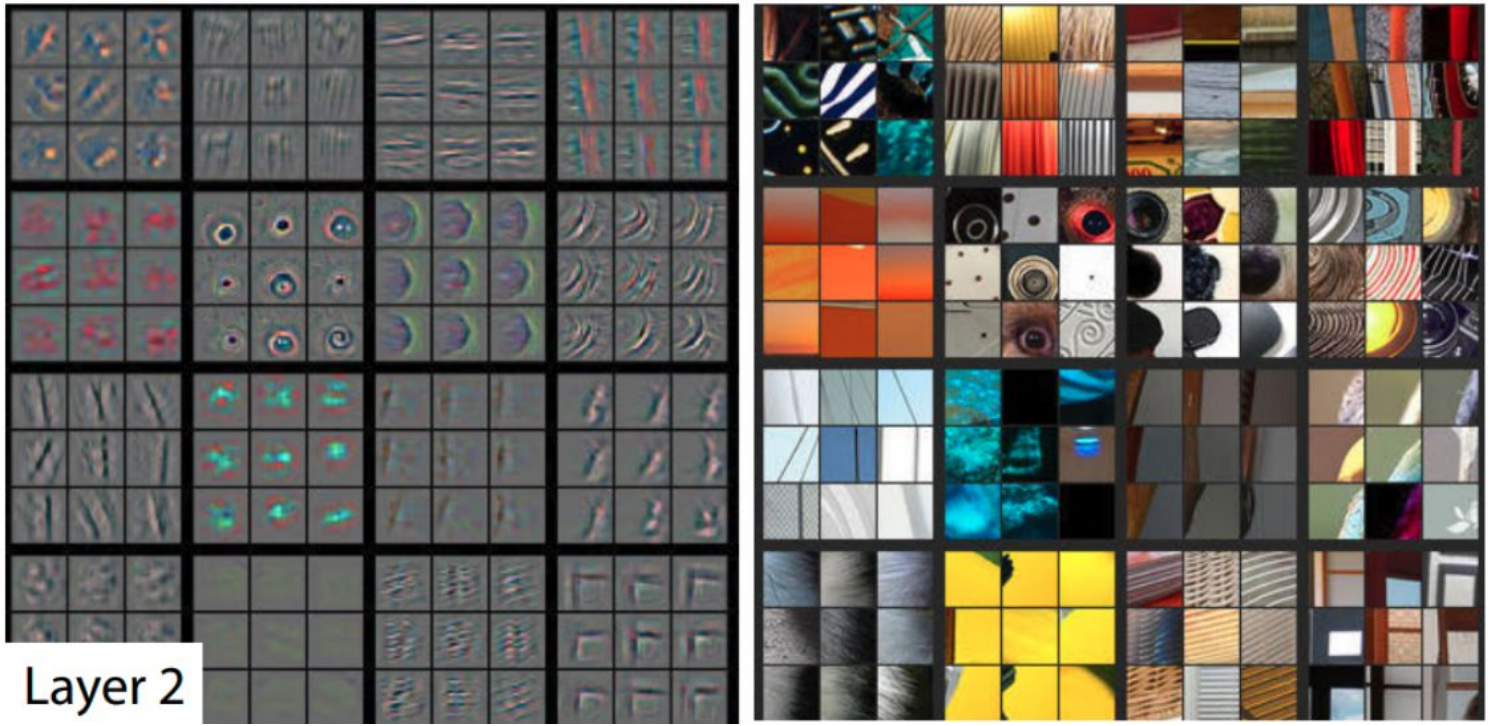
we can visualize what each layer is doing by finding the image patch with highest activation responses.

What have been learned in CNN ? (for image classification)

- **Layer 1** focuses on colored **edges** and **blobs**.

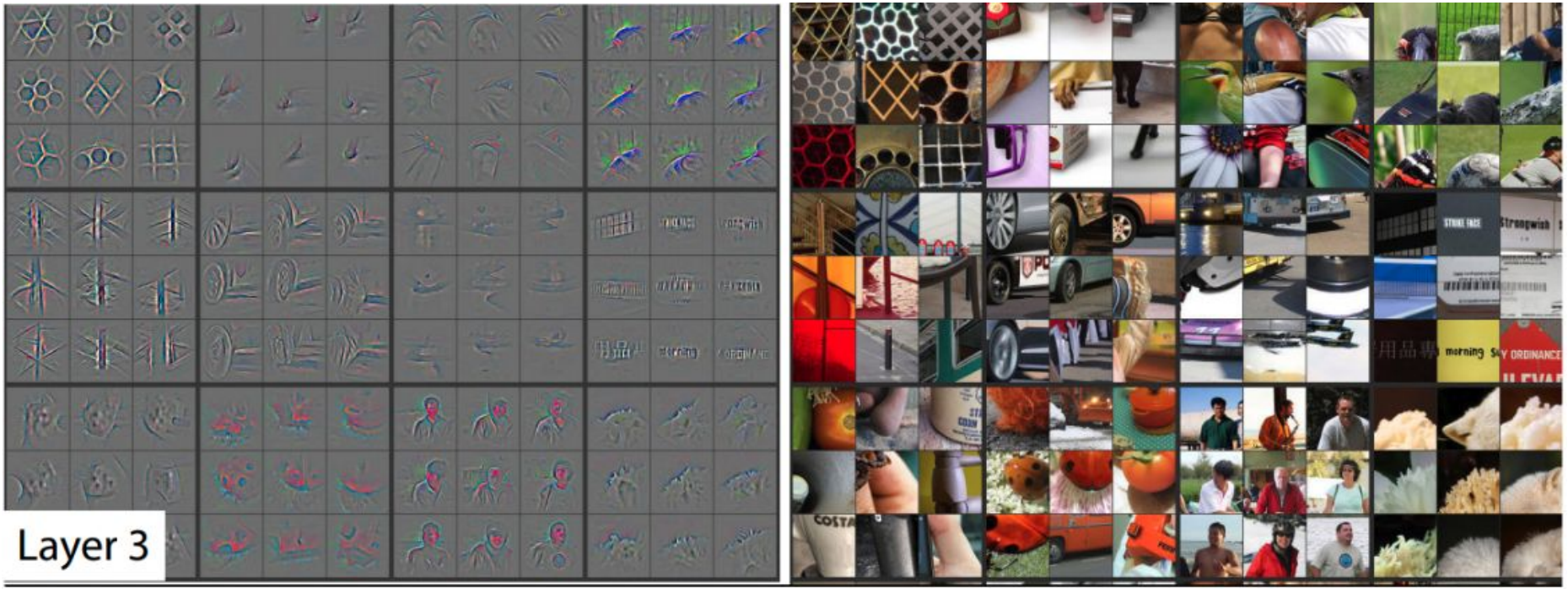
What have been learned in CNN ? (for image classification)

- **Layer 2-3** focus on **object parts**, such as the wheels of a car, or the beak of a bird.



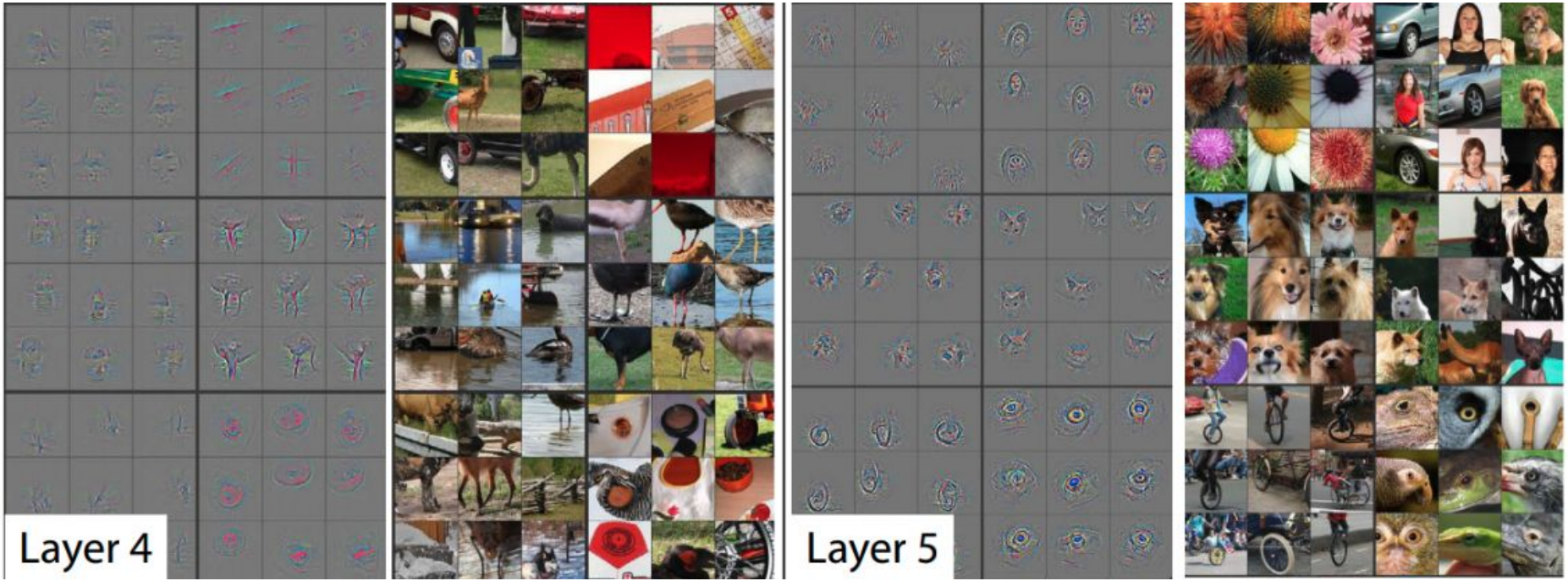
What have been learned in CNN ? (for image classification)

- **Layer 2-3** focus on **object parts**, such as the wheels of a car, or the beak of a bird.

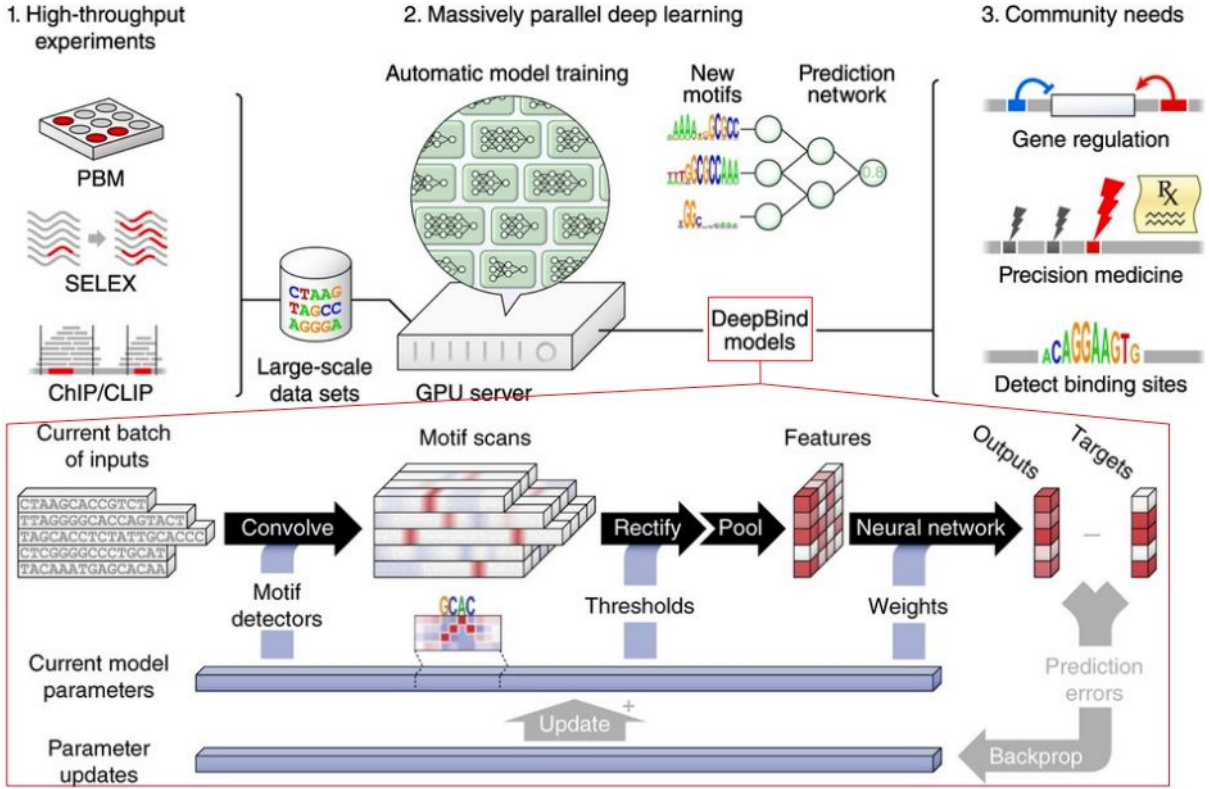


What have been learned in CNN ? (for image classification)

- **Layer 4-5** focus on **object parts** and even **entire objects**.



Example: predicting DNA/RNA-protein binding



Alipanahi, Babak, et al. "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning." *Nature biotechnology* 33.8 (2015): 831-838.

Summary: key ideas of CNNs

- Use a set of filters to extract **local** features
- Use **multiple** filters to extract different features
- Spatially **share** parameters of each filter

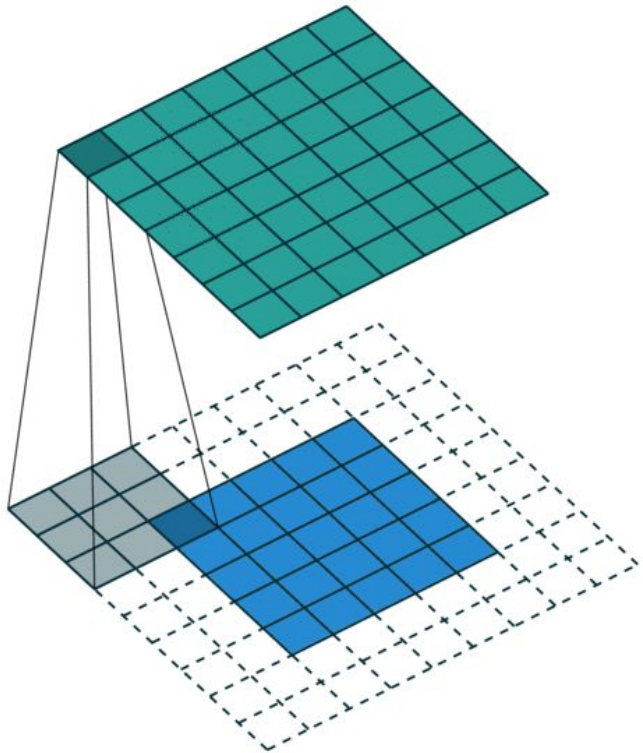


Image from Google

CNN summary

CNNs

- Are used for all aspects of **computer vision**, and have won numerous pattern recognition competitions
- Able learn **interpretable features** at different levels of abstraction
- Typically, consist of **convolution** layers, **pooling** layers, **nonlinearities**, and **fully connected** layers

A CSE Faculty Candidate Seminar Tomorrow (02/08)

Some of the coolest algorithms (e.g., sketching, graph theory, alignment) for biological sequence data analyses!

Title: Digital Genome Reconstruction with De Novo Assembly

Speaker: Haoyu Cheng (postdoctoral scholar at Dana-Farber Cancer Institute and Harvard Medical School)

Location: **CODA** Building, Second Floor, Room 230

Date: Thursday, **February 8, 2024 at 11:00 am**

Abstract:
https://www.cse.gatech.edu/events/2024/02/08/cse-faculty-candidate-seminar-haoyu-cheng