Farielberry Lab

Deep Learning for Genomic Prediction

FACULTAD DE
INGENIERÍA

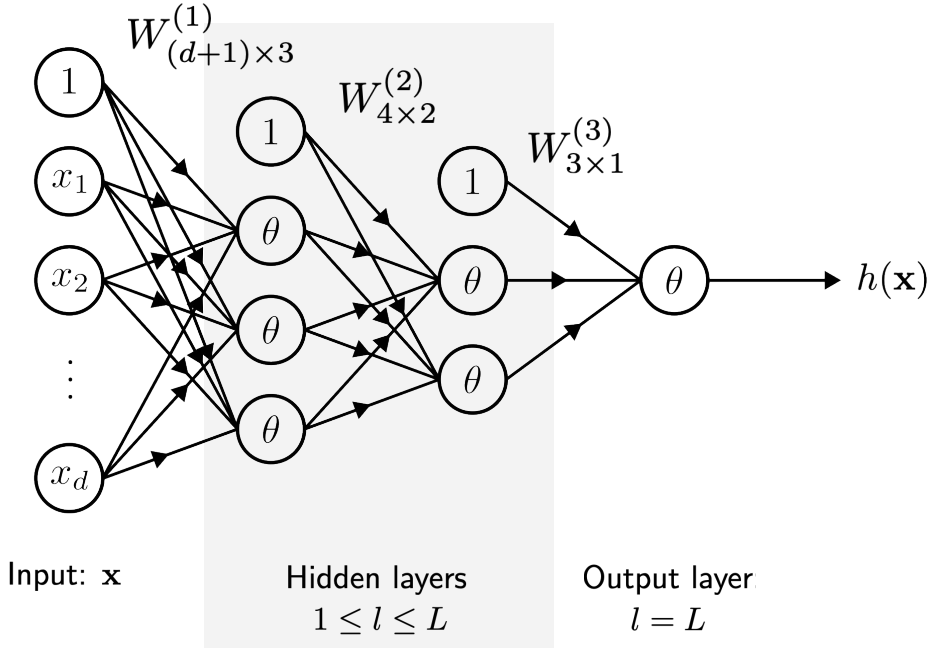
UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Convolutional Neural Network (CNN)
Residual Network (ResNet)

Agenda for today

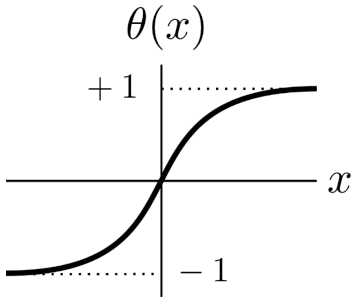
- Perceptron
- Multi-Layer Perceptron
- Backpropagation
- Neural Network
- Convolutional Neural Network
- Residual Neural Network
- Practical

Neural Networks



Activation Function

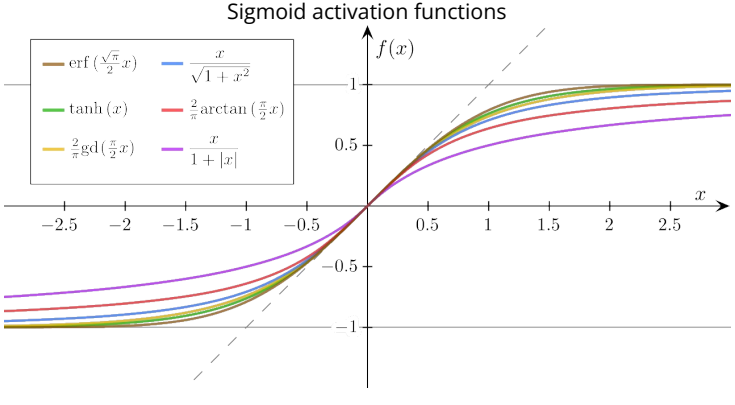
(differentiable!!!)



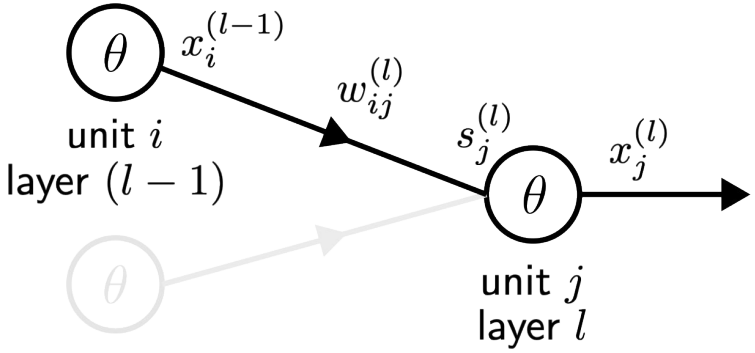
$$\theta(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Layers dimensions: $\mathbf{d} = [d, 3, 2, 1]$

$$\mathbf{d} = \left[d^{(0)}, d^{(1)}, \cdots, d^{(L)}\right]$$



Neural Networks

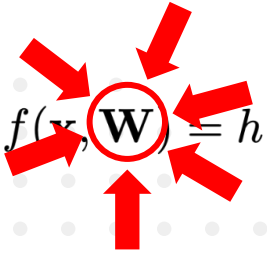


$$w_{ij}^{(l)} \quad \begin{cases} 1 \le l \le L & \text{layers} \\ 0 \le i \le d^{(l-1)} & \text{inputs} \\ 1 \le j \le d^{(l)} & \text{outputs} \end{cases}$$

Signal	Notation	Dimensions
Signal in	$\mathbf{s}^{(l)}$	$d^{(l)}$
Output	$\mathbf{x}^{(l)}$	$d^{(l)}+1$
Weights in	$\mathrm{W}^{(l)}$	$\left(d^{(l-1)}+1\right) \times d^{(l)}$

$$\mathbf{s}^{(l)} = \left(\mathrm{W}^{(l)}\right)^{\top} \mathbf{x}^{(l-1)} \qquad x_j^{(l)} = \theta\left(s_j^{(l)}\right) = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}\right) \qquad \mathbf{x}^{(l)} = \left[1, \theta\left(\mathbf{s}^{(l)}\right)\right]^{\top}$$

$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{\mathrm{W}^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{\mathrm{W}^{(2)}} \mathbf{s}^{(2)} \xrightarrow{\theta} \mathbf{x}^{(2)} \cdots \longrightarrow \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x})$$

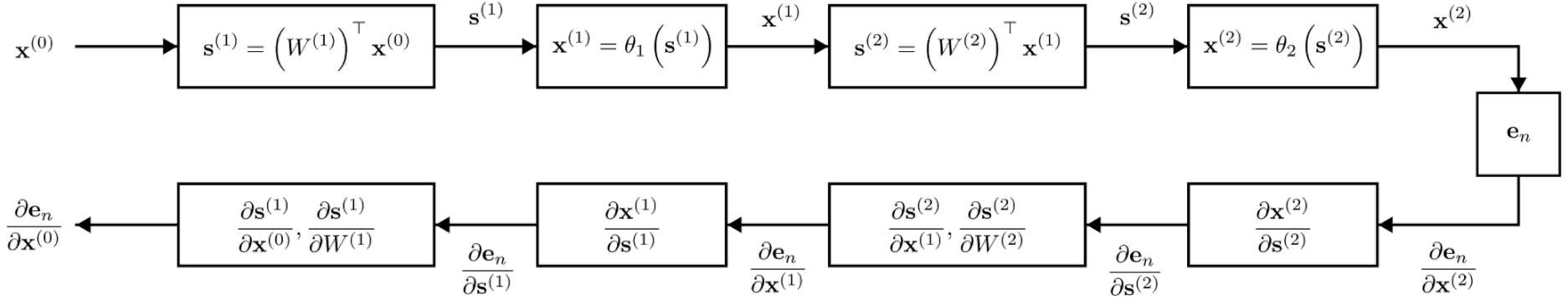


$$f(\mathbf{x}, \mathbf{W}) = h(\mathbf{x}) = \sigma\left(\left(W^{(L)}\right)^{\top} \sigma\left(\left(W^{(L-1)}\right)^{\top} \sigma\left(\cdots \sigma\left(\left(W^{(1)}\right)^{\top} \mathbf{x}^{(0)}\right)\right)\right)\right)$$

Neural Networks: forward and backward

Forward ⟶

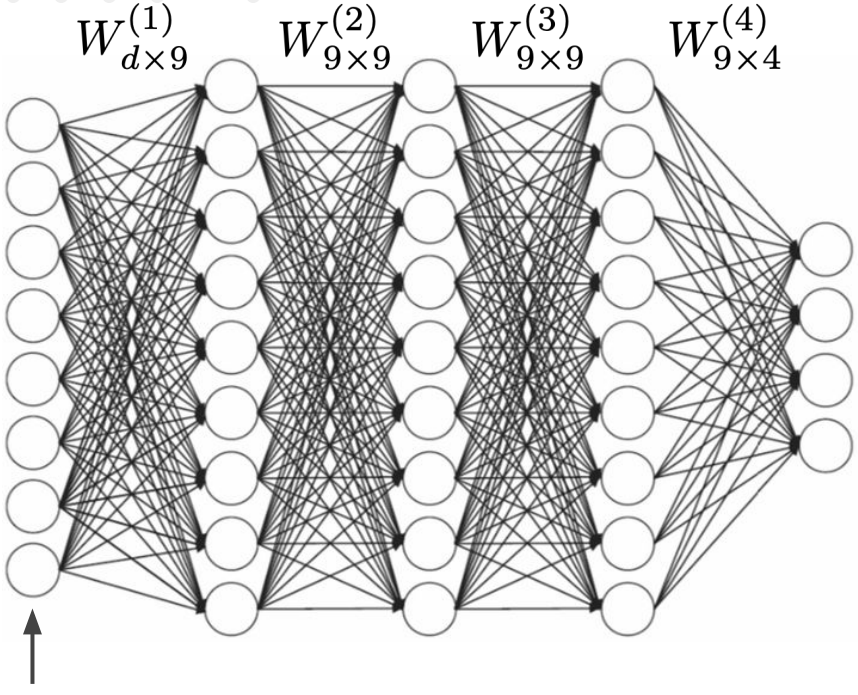
$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{\mathbf{W}^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{\mathbf{W}^{(2)}} \mathbf{s}^{(2)} \xrightarrow{\theta} \mathbf{x}^{(2)} \dots \longrightarrow \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x})$$



$$\boldsymbol{\delta}^{(1)} \longleftarrow \boldsymbol{\delta}^{(2)} \dots \longleftarrow \boldsymbol{\delta}^{(L-1)} \longleftarrow \boldsymbol{\delta}^{(L)}$$ ⟵ Backward

Neural networks and images

Neural Networks... and images



$$d = 1000 \times 1000 \times 3$$

$$d^{(1)} = 1000 \Rightarrow W^{(1)}_{(3\times 10^6)\times 1000}$$

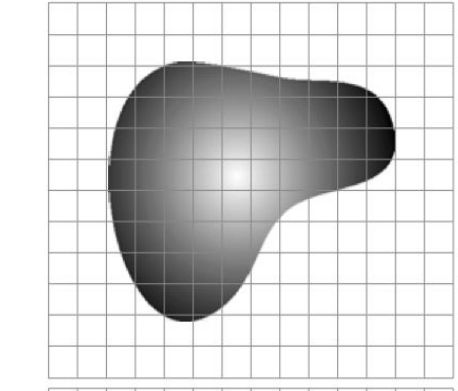
3 billion parameters

- Deep arquitectures
- Learned non-linear transformations
- GD with non-convex functions
  - does not guarantee global optimum
  - global minimum: overfitting
  - local minima with similar performance
- Large number of parameters
- Generalization & overfitting
- Data, data, data, ...
- Not suitable for (big) images
- Vectorized images lose their structure
- Non multi-scale analysis

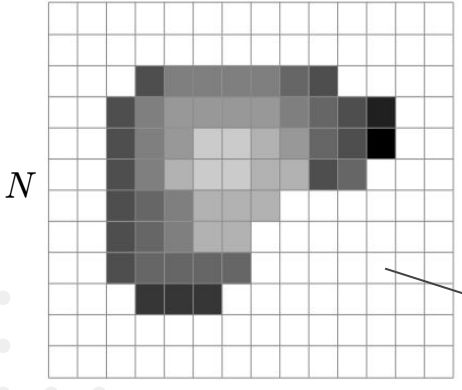
Digital images

Digital images representation

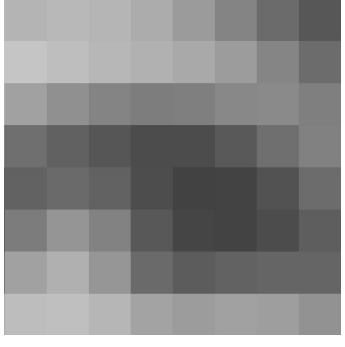
A digital image is a matrix of values in a certain range.



Continuous image in space and values.



Sampling and quantification result.



166	166	161	149	122	107	102	98
167	167	161	148	122	108	104	101
167	166	160	148	122	110	107	105
168	165	159	147	122	109	107	105
169	165	155	139	113	103	104	102
170	167	158	143	117	107	105	104
169	168	160	146	121	109	107	105
167	166	159	144	120	108	107	105

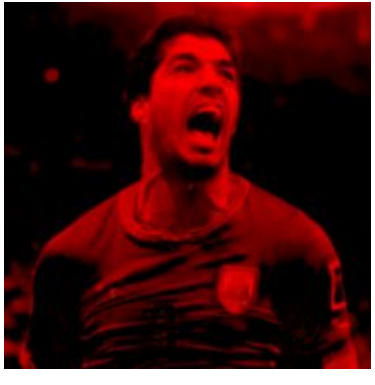
Digital images

A color (RGB) image is a three **channels** image, with three matrices.

Red channel

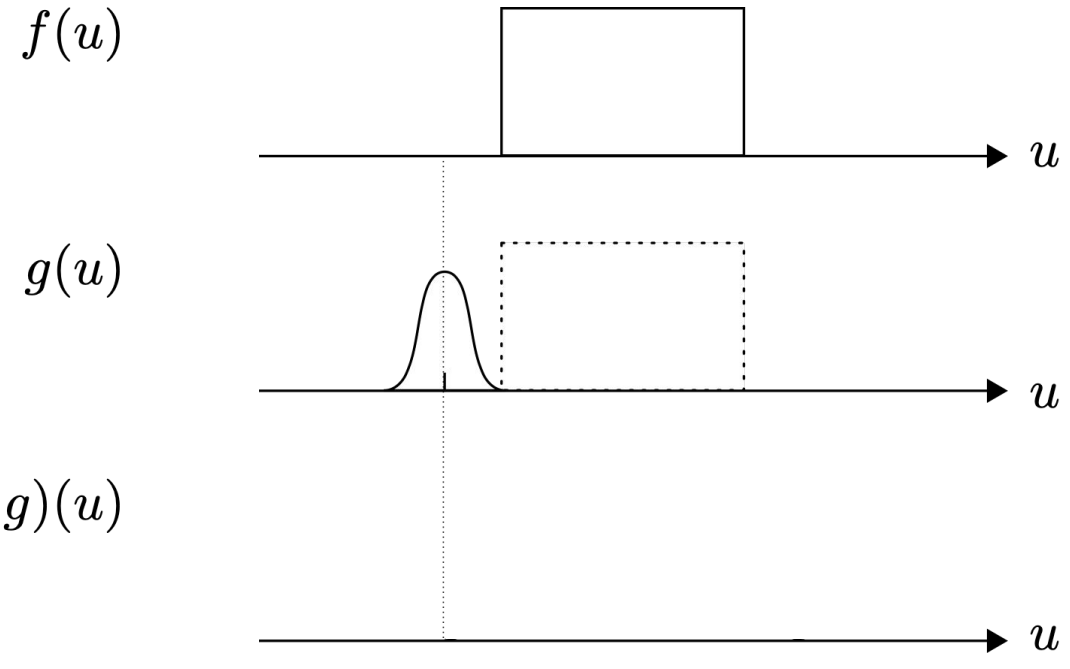
Green channel

Blue channel



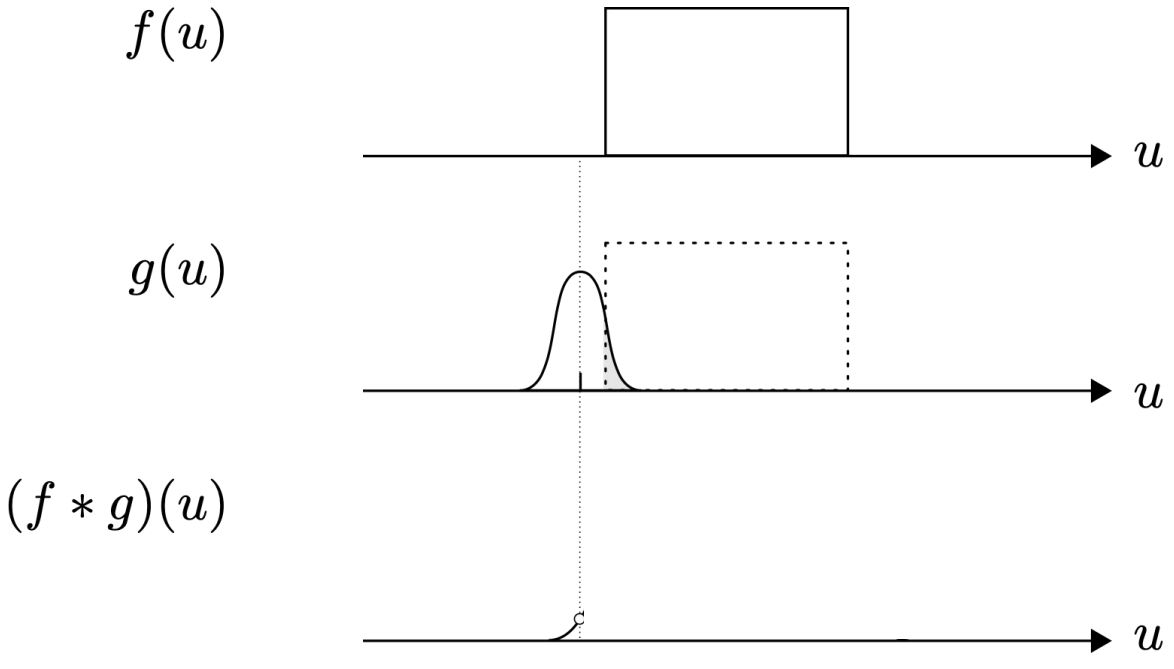
Convolution

$$(f * g)(u) = \int_{-\infty}^{\infty} f(v)g(u-v)\,dv = \int_{-\infty}^{\infty} f(u-v)g(v)\,dv$$



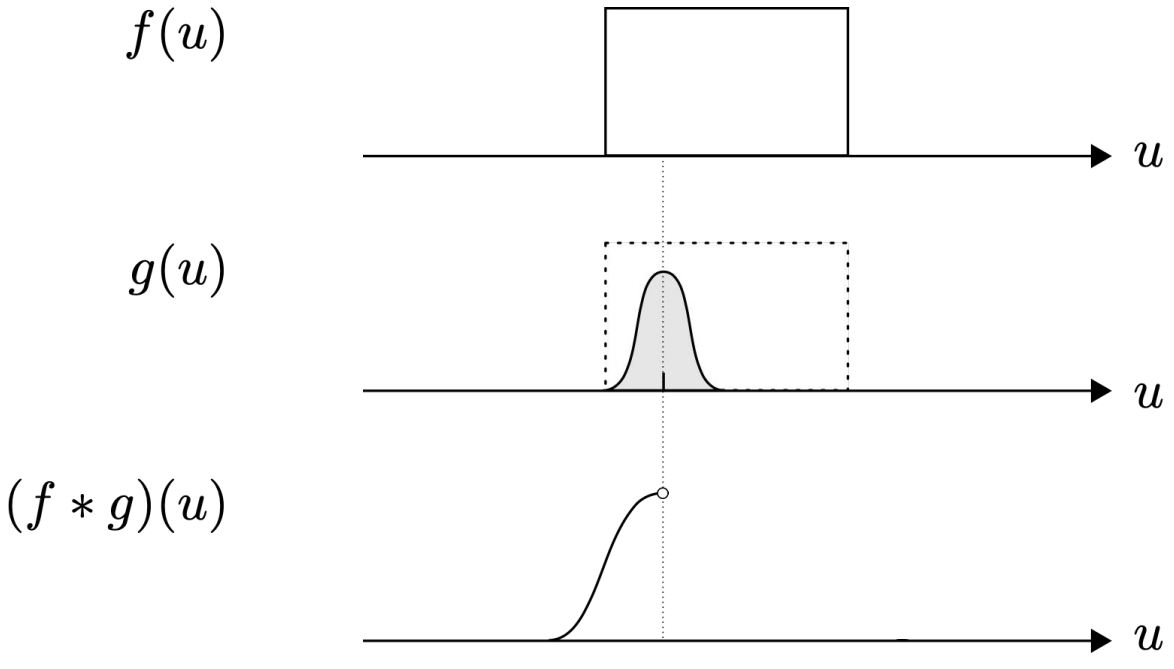
Convolution

$$(f * g)(u) = \int_{-\infty}^{\infty} f(v)g(u-v)\,dv = \int_{-\infty}^{\infty} f(u-v)g(v)\,dv$$



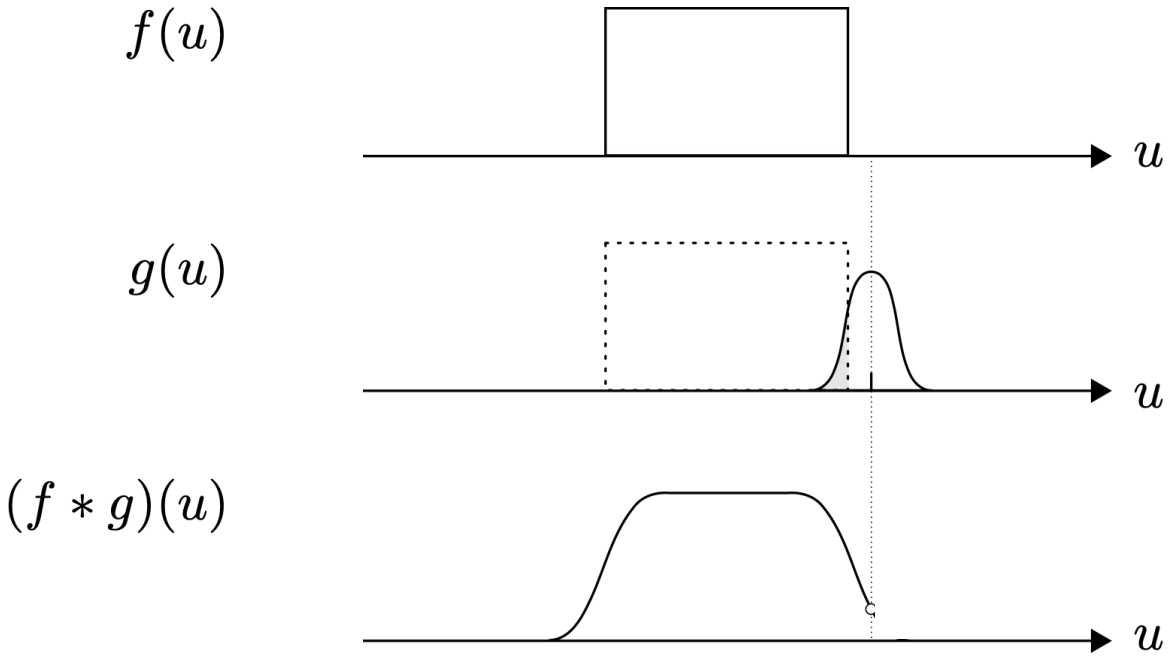
Convolution

$$(f * g)(u) = \int_{-\infty}^{\infty} f(v)g(u-v)\,dv = \int_{-\infty}^{\infty} f(u-v)g(v)\,dv$$



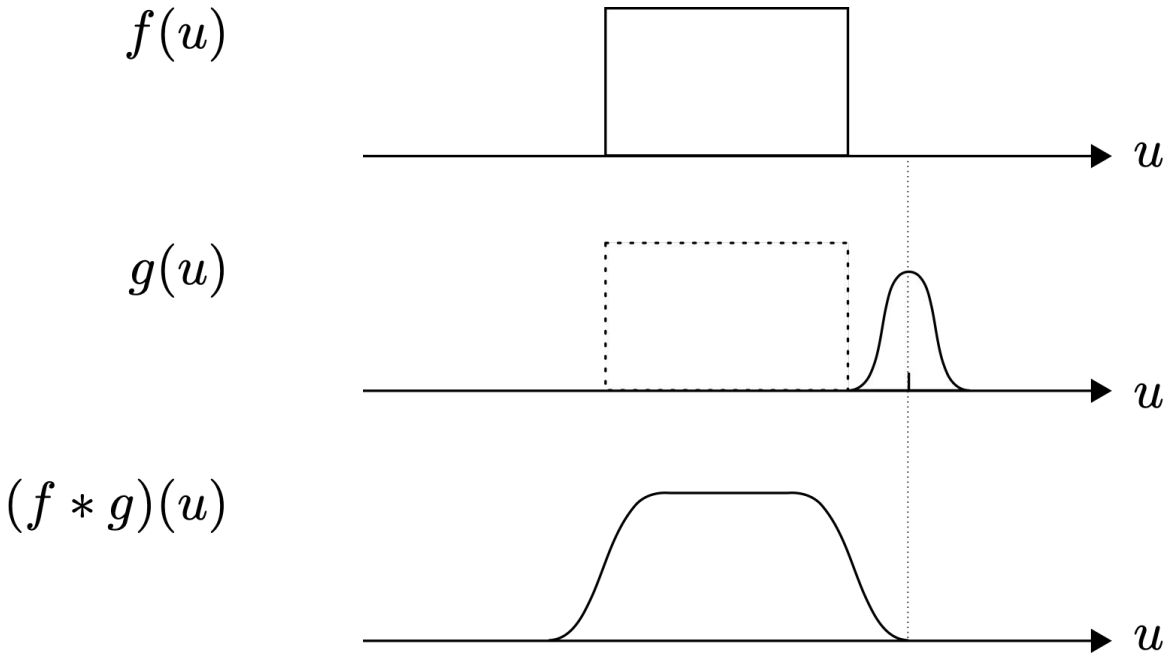
Convolution

$$(f * g)(u) = \int_{-\infty}^{\infty} f(v)g(u-v)\,dv = \int_{-\infty}^{\infty} f(u-v)g(v)\,dv$$

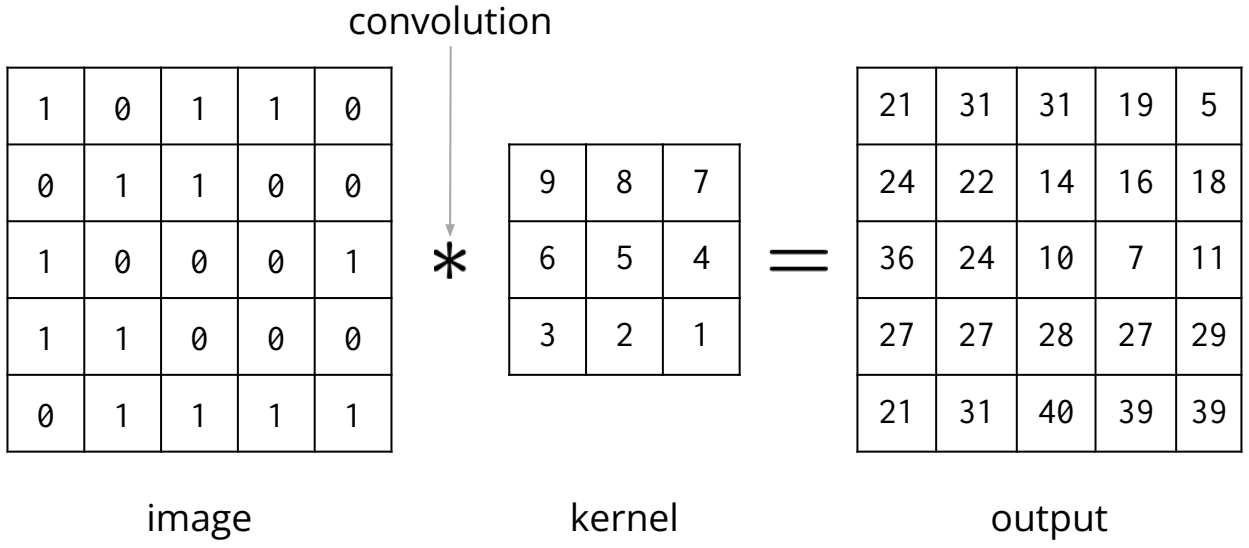


Convolution

$$(f * g)(u) = \int_{-\infty}^{\infty} f(v)g(u-v)\,dv = \int_{-\infty}^{\infty} f(u-v)g(v)\,dv$$



Convolution and images

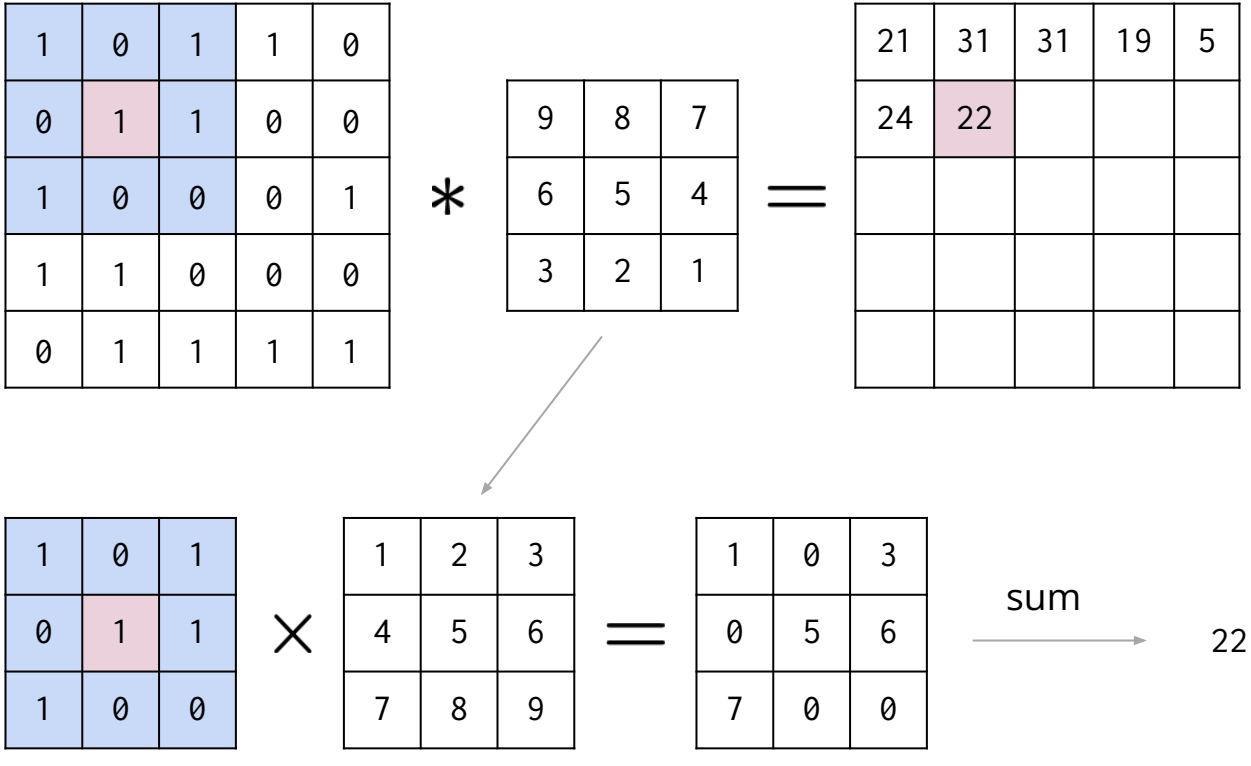


- 2D convolution

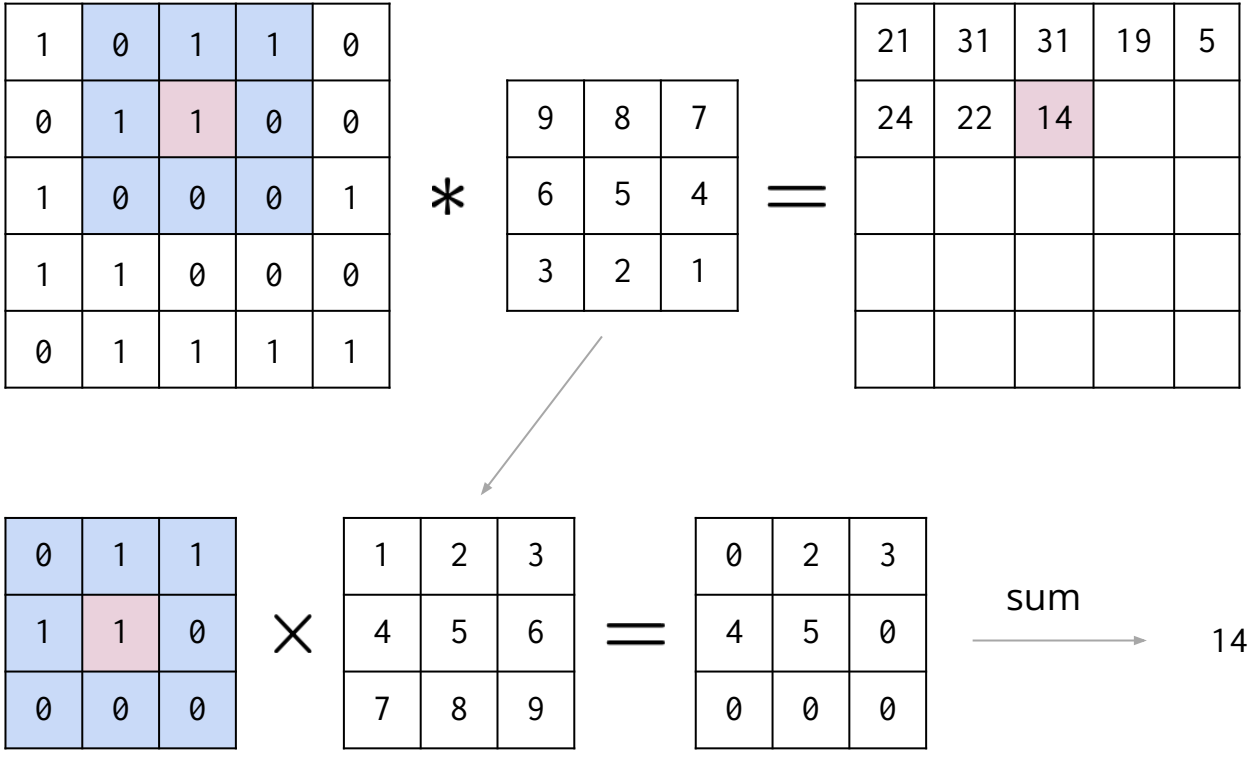
$$\hat{I}(u,v) = (I * K)(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(u,v)K(u-m, v-n)$$

- It is a cross-correlation!

Convolution and images



Convolution and images



Convolution and images: filtering

Feature descriptor 1

$*$

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

$=$

$*$

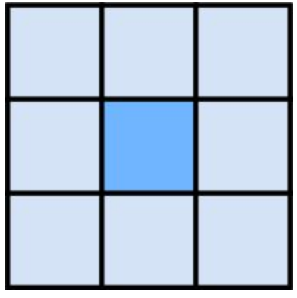
1	0	-1
2	0	-2
1	0	-1

$=$

Feature descriptor 2

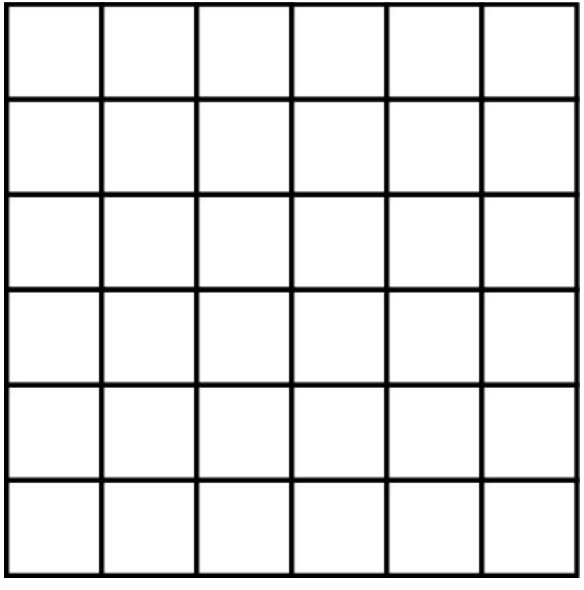
Convolution and images: padding

k_m



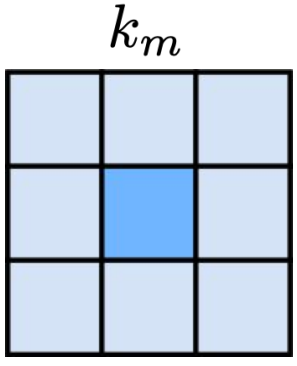
Filter (kernel)

M

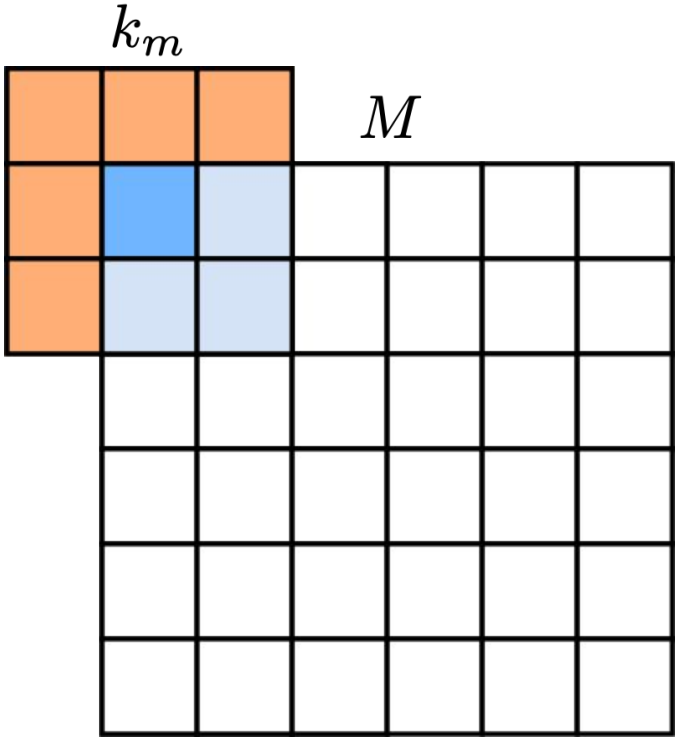


Image

Convolution and images: padding



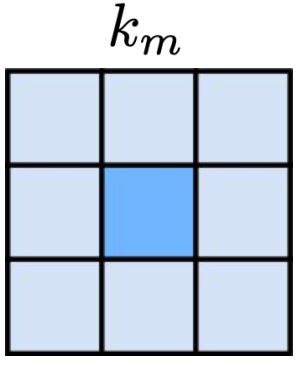
Filter (kernel)



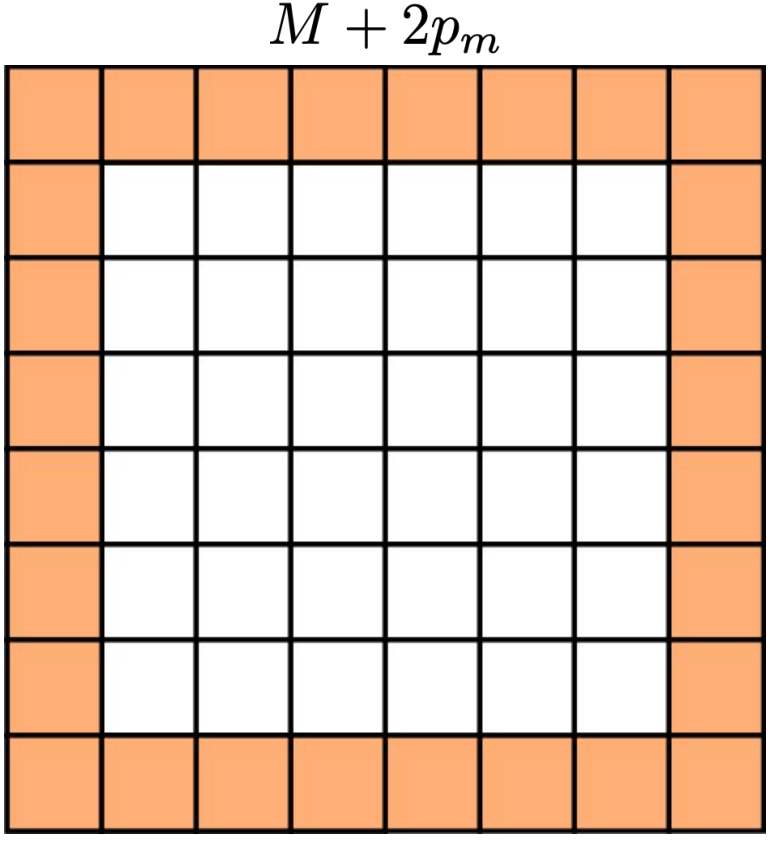
Padding pixel

$$p_m = \frac{k_m - 1}{2}$$

Convolution and images: padding



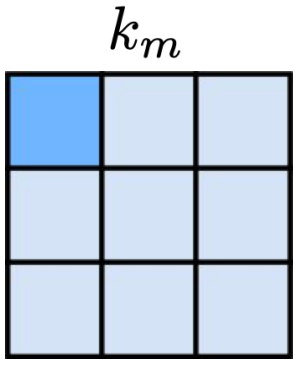
Filter (kernel)



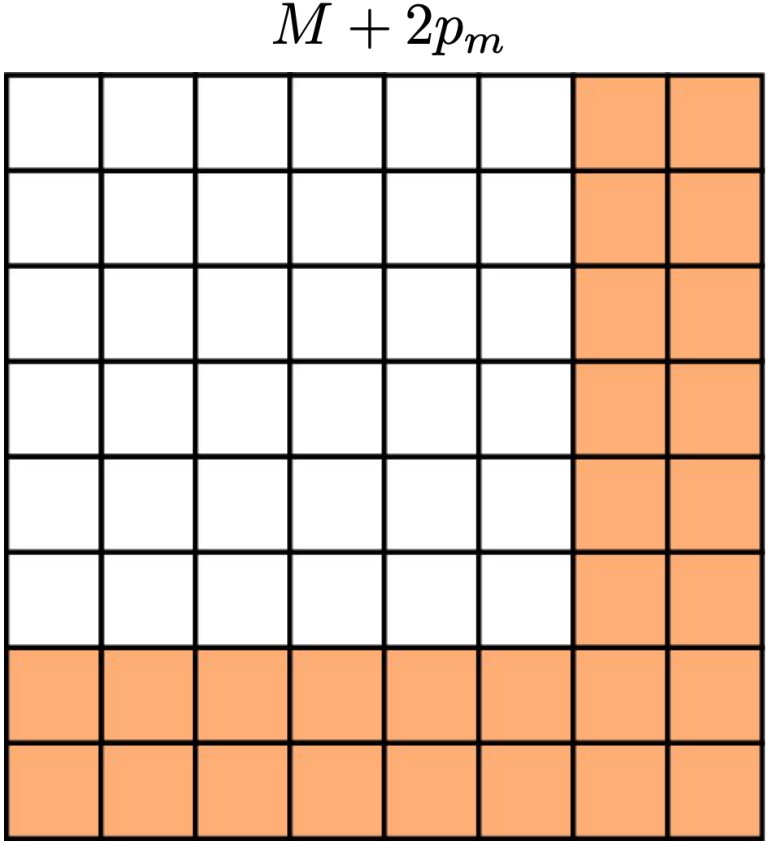
Padding pixel

$$p_m = \frac{k_m - 1}{2}$$

Convolution and images: padding



Filter (kernel)

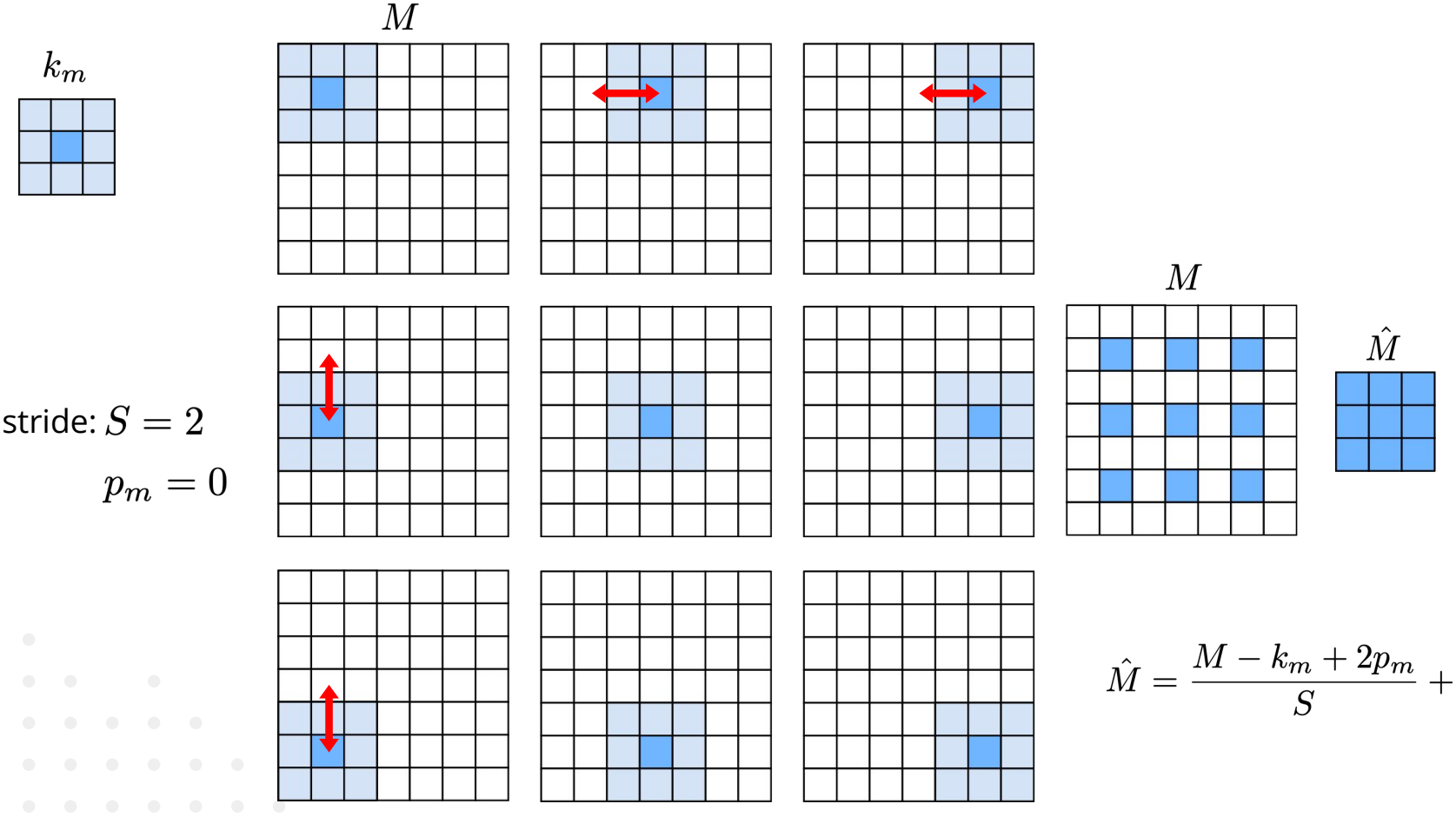


Padding pixel

$$p_m = \frac{k_m - 1}{2}$$

Convolution and images: stride

Stride: skip intermediate locations when *moving* the kernel.



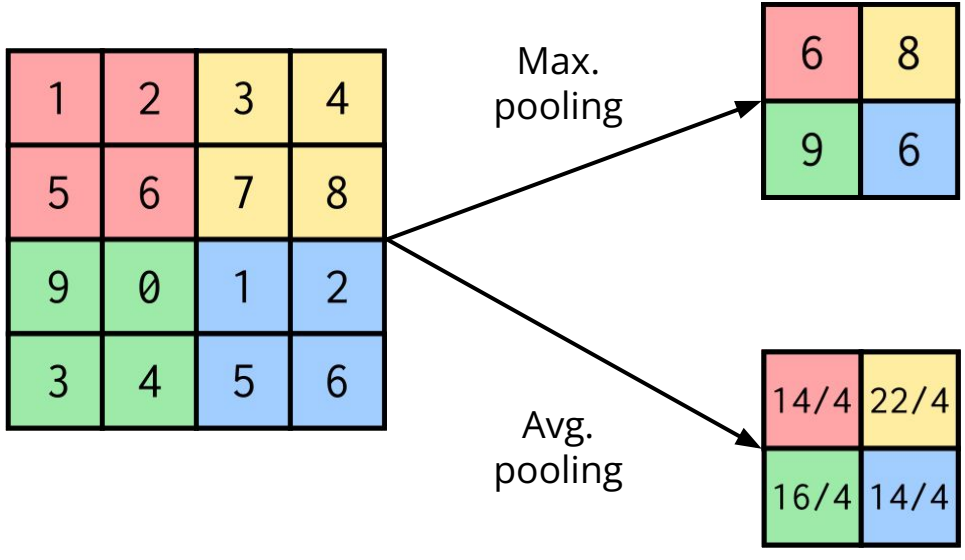
stride: $S = 2$

$p_m = 0$

$$\hat{M} = \frac{M - k_m + 2p_m}{S} + 1$$

Convolution and images: pooling

Reduces feature descriptor maps resolution,
and increases translation invariance.



$$z_n = \max_{k=0,\ldots,K-1} x_{n\times\text{stride}+k}$$

Neural networks: batch normalization

Normalization of the means and variances of each layer's inputs. It is computed in each mini-batch in the training process.

Denote B as a mini-batch of size N_B of the entire training set.

$$\mathbf{x} = (x_1, \cdots, x_d)^\top$$

$$\mu_B[k] = \frac{1}{N_B} \sum_{i=1}^{N_B} x_i[k] \text{ for } k = 1, \cdots, d$$

$$\sigma_B^2[k] = \frac{1}{N_B} \sum_{i=1}^{N_B} (x_i[k] - \mu_B[k])^2$$

$$\hat{x}_i[k] = \frac{x_i[k] - \mu_B[k]}{\sqrt{\sigma_B^2[k] + \epsilon}}$$

Normalized input, internal to each layer.

$$y_i[k] = \gamma[k]\, \hat{x}_i[k] + \beta[k]$$

Restored representation.

Neural networks: softmax

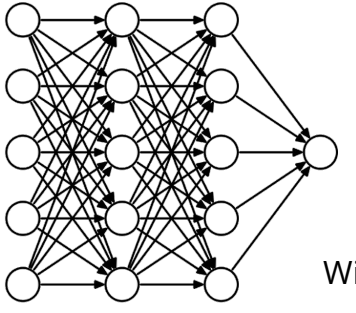
- Converts a vector of K real numbers into a probability distribution of K possible outcomes.
- *Probabilistic* output.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \cdots K \text{ and } \mathbf{z} = (z_1, \cdots, z_K) \in \mathbb{R}^K$$

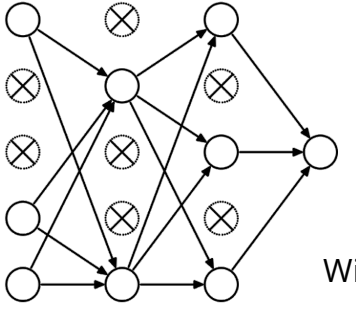
Neural networks: dropout

- Regularization techniques for reducing overfitting on training data.
- Randomly put to zero a fraction of the weights (except for the output layer).
- Layers neurons does not synchronously optimizing their weights, thus decorrelating the weights.
- The activations of the hidden units become sparse, which is desirable.

$$\hat{w}_{ij} = \begin{cases} w_{ij} & \text{with probability } (1-p) \\ 0 & \text{with probability } p \end{cases}$$



Without dropout.



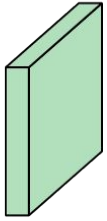
With dropout.


FILTERING IS CHANGING THE COLOR OF A PIXEL GIVEN THE COLORS OF ITS NEIGHBORING PIXELS
IS THE KEY TO CONVOLUTIONAL NEURAL NETWORKS
THE WAY IT CHANGES DEPENDS ON A "FILTER"

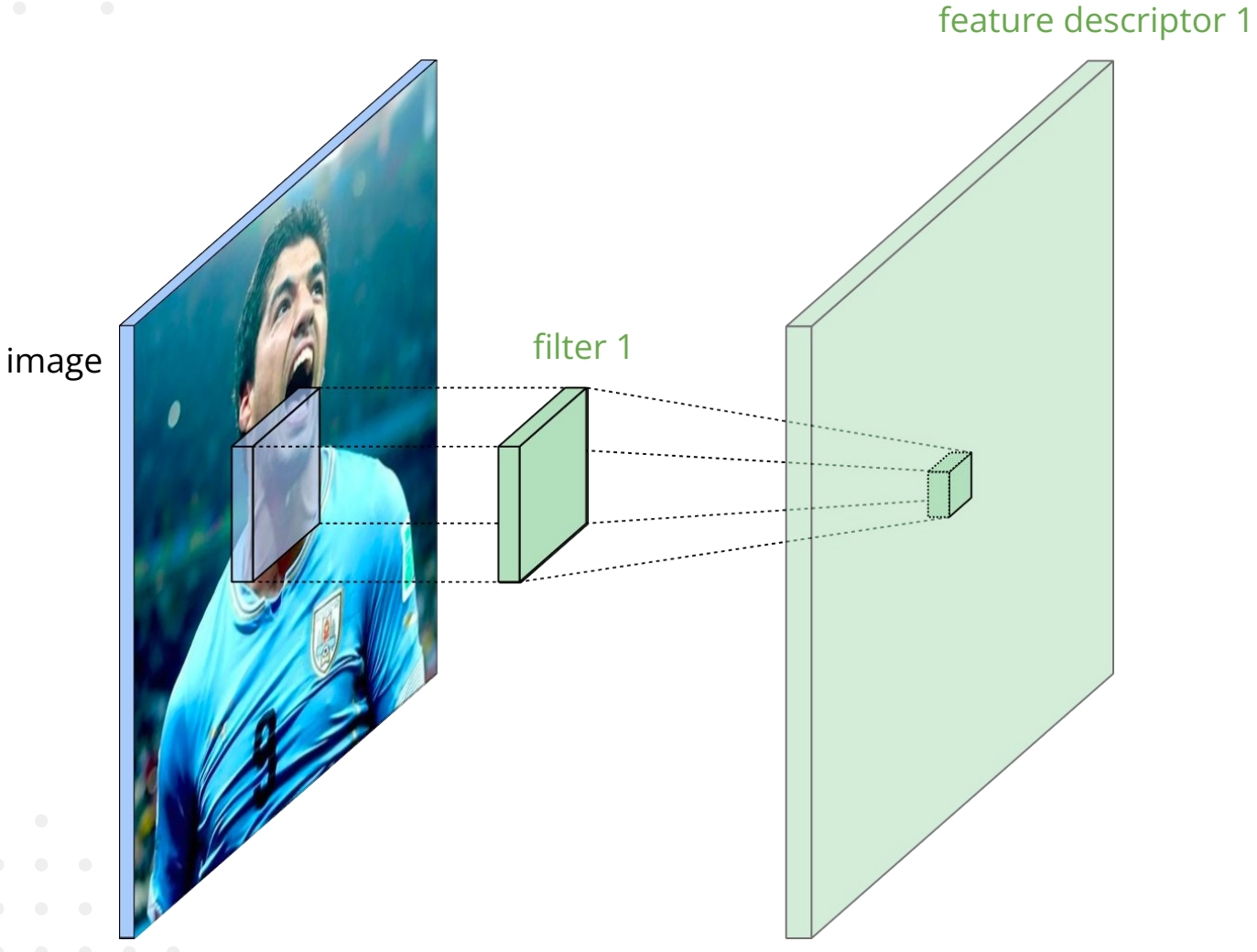
Convolutional Neural Network (CNN)

image

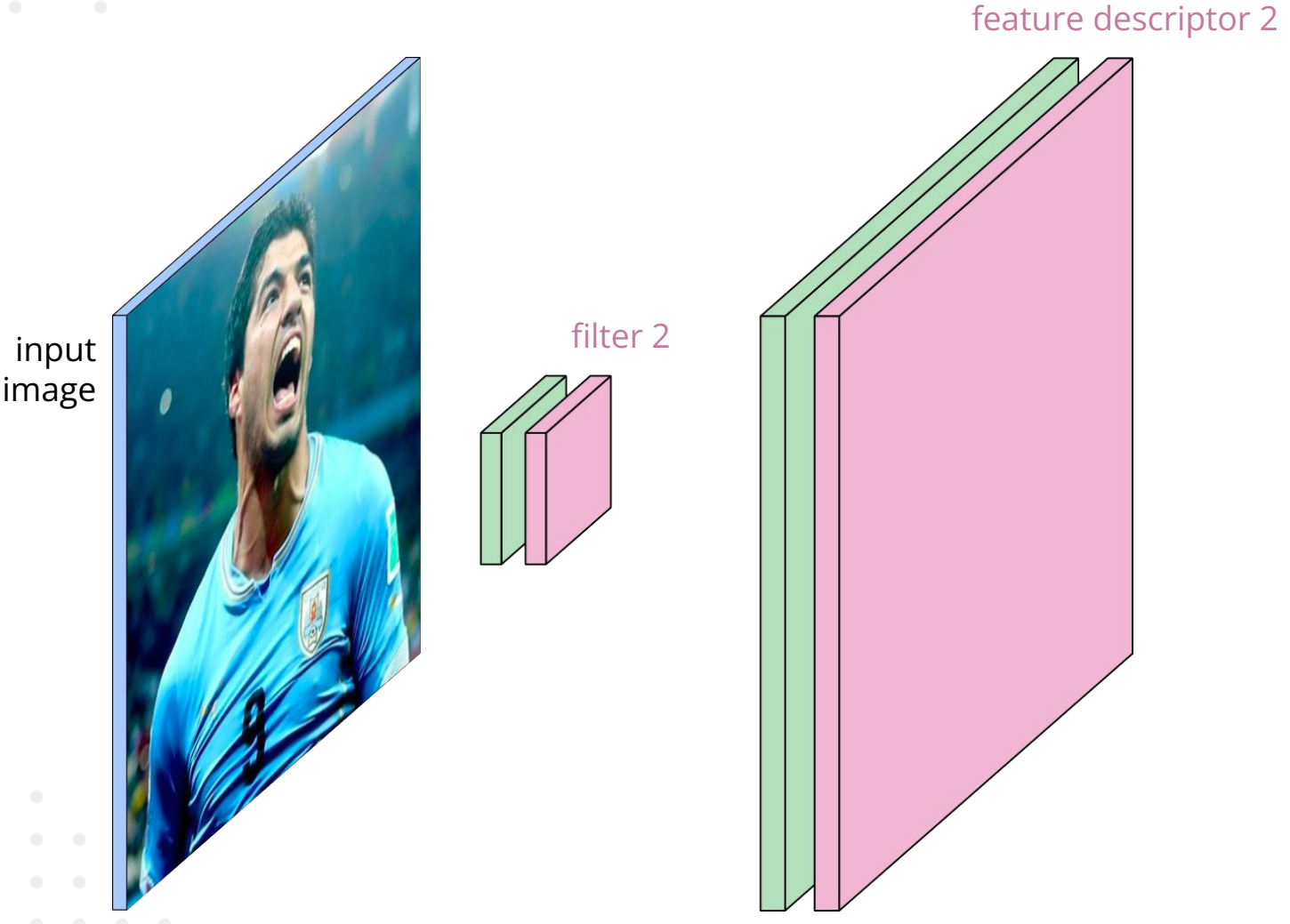
filter 1



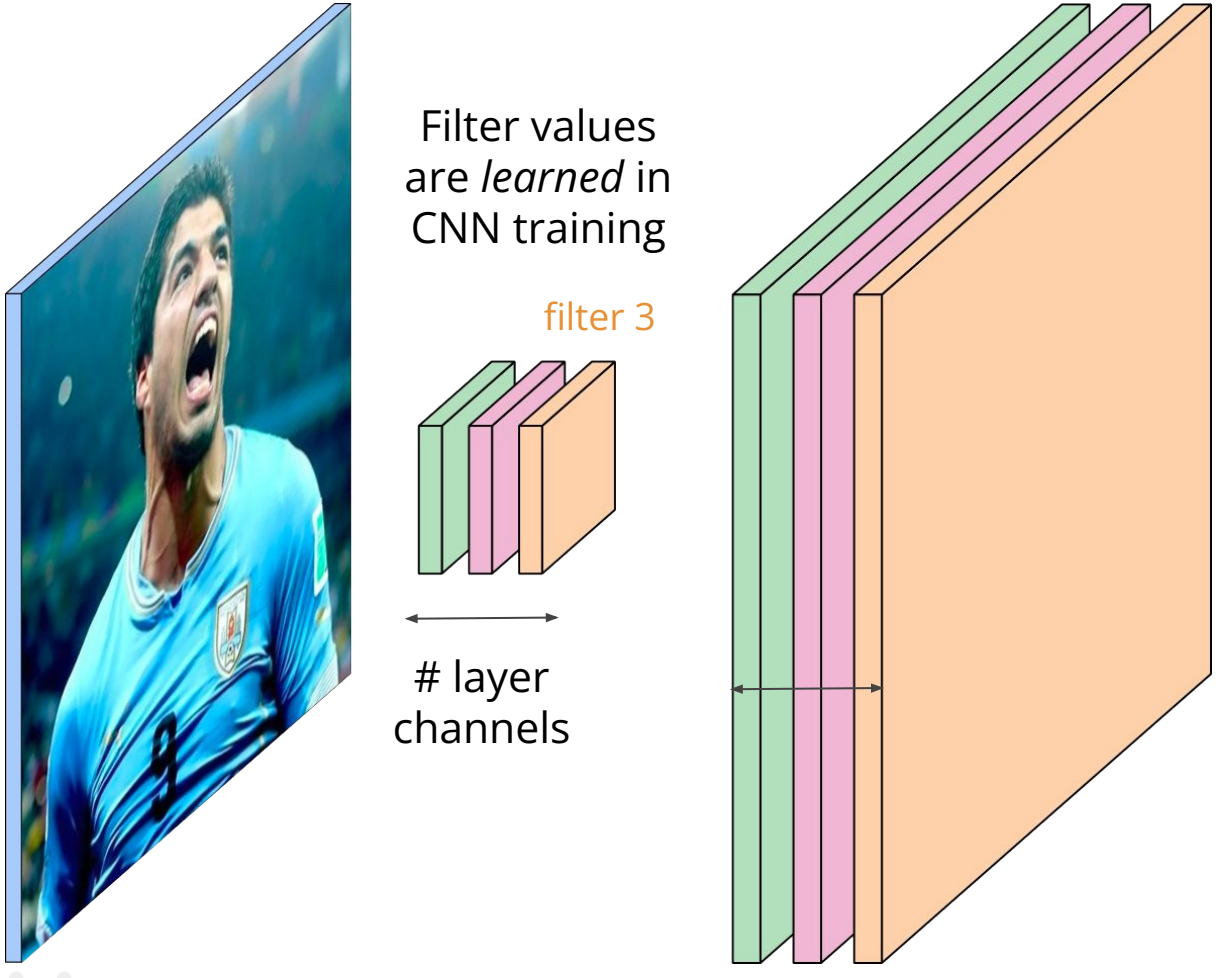
Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)



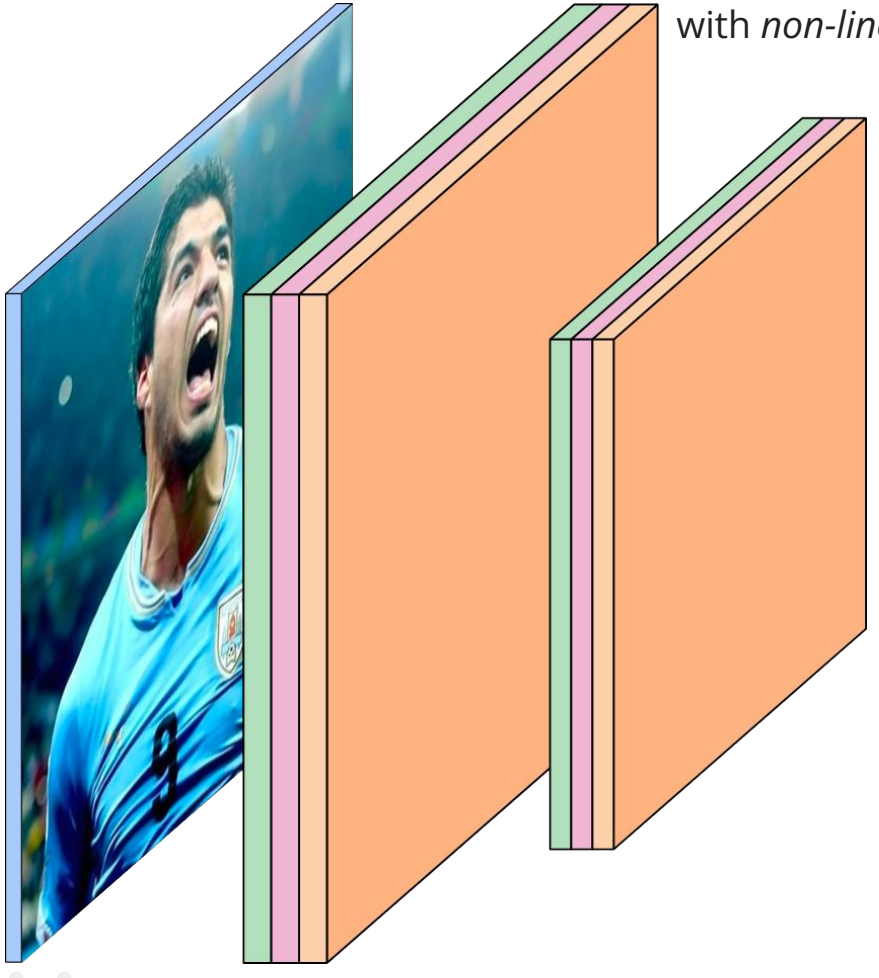
Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)

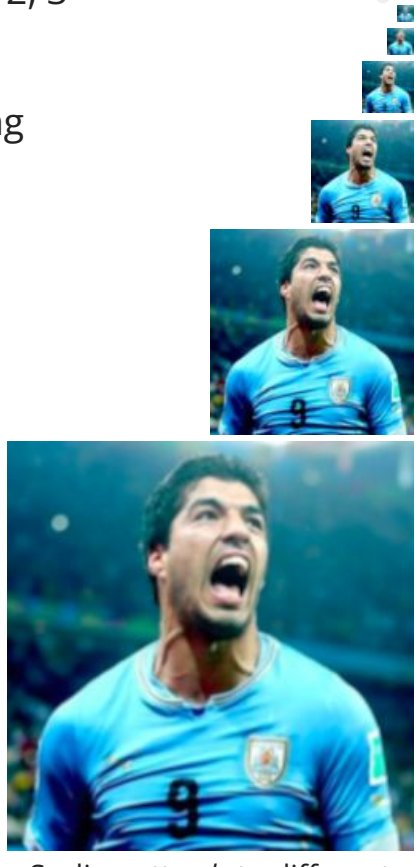
Feature descriptors 1, 2, 3
with *non-linear* stage

Convolutional Neural Network (CNN)



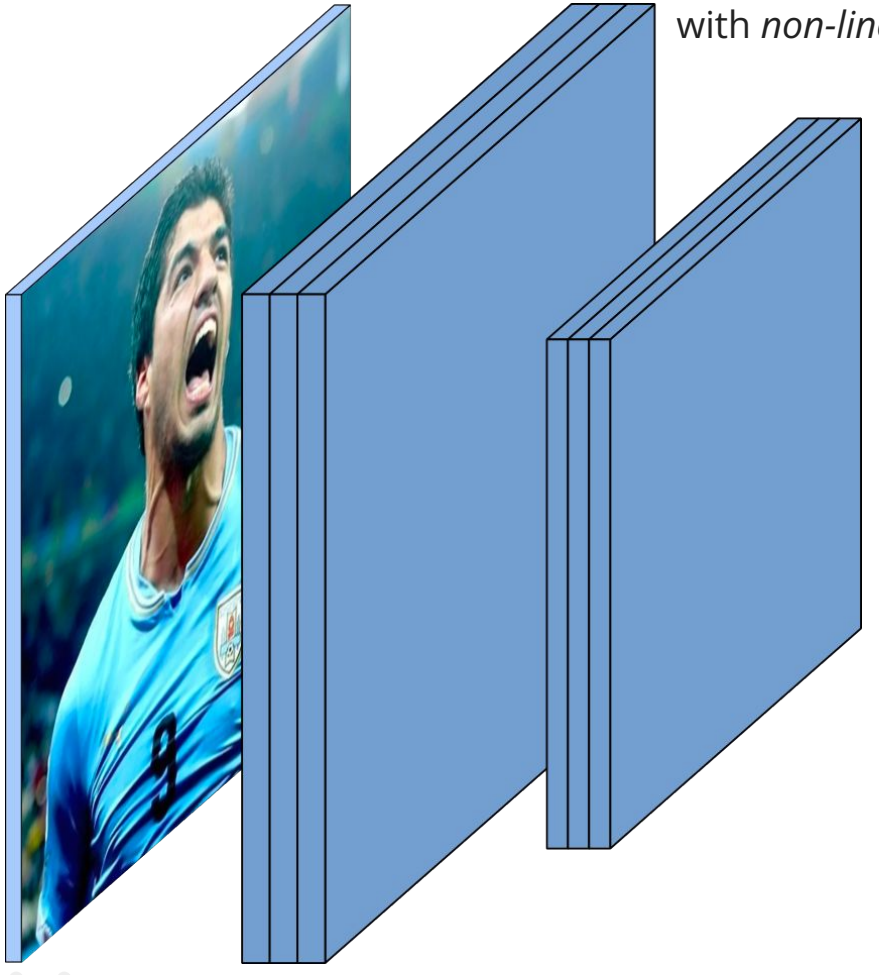
Feature descriptors 1, 2, 3
with *non-linear* stage

size scaling
(pooling)



Scaling *attends* to different
levels of details

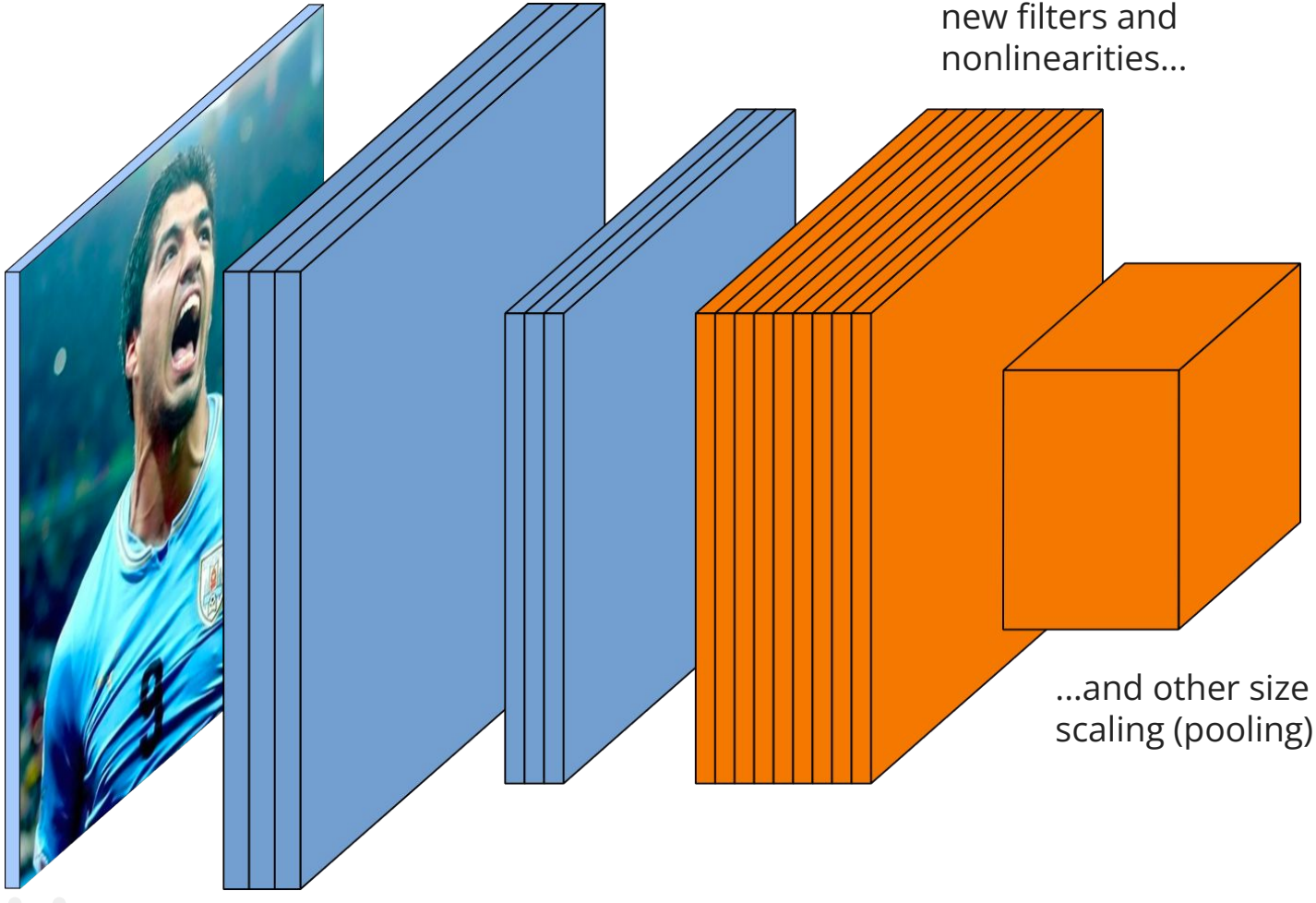
Convolutional Neural Network (CNN)



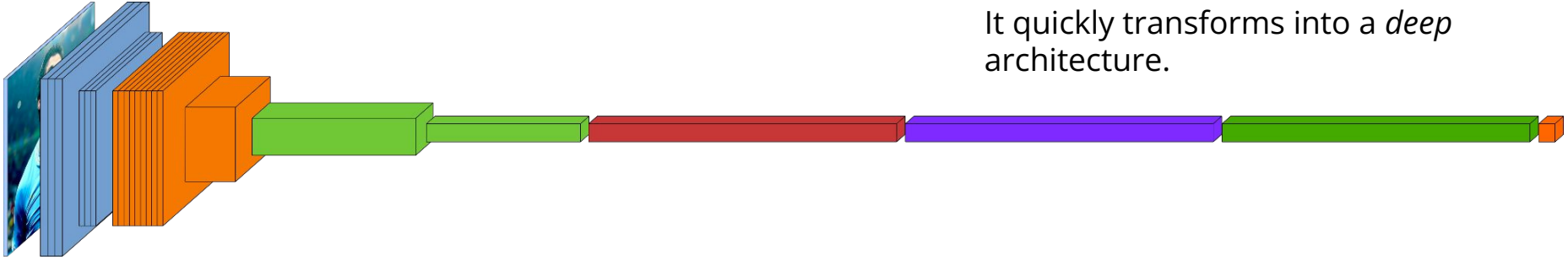
Feature descriptors 1, 2, 3
with *non-linear* stage

size scaling
(pooling)

Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)



It quickly transforms into a *deep* architecture.

Convolutional Neural Network (CNN)

Computation time on Intel Xeon 3rd Gen Scalable cpu: 4.073 s

wall	1.000
sky	1.000
ceiling	1.000
person	1.000
fence	1.000
grandstand	1.000
stage	1.000

Hugging Face

UperNet, ConvNeXt small-sized backbone

Convolutional Neural Network (CNN)

Training dataset

Training

Filter parameters are *learned*

CNN

Test dataset

Assign one of the learned *labels*

What does a CNN "see"?

Layer 1

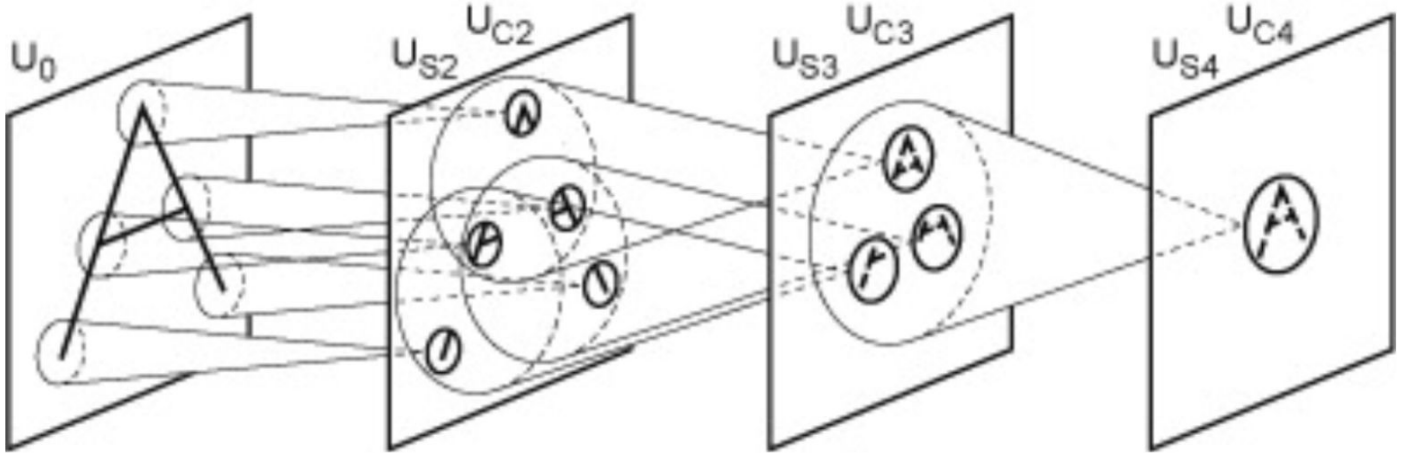
Layer 2

Layer 3

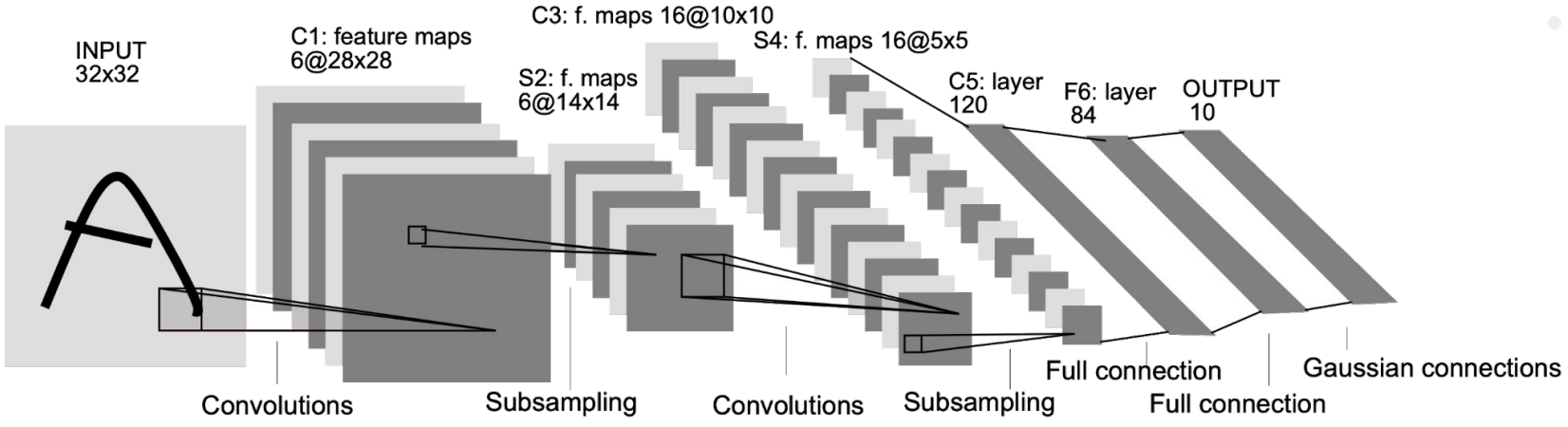
Layer 4

Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. Springer International Publishing, 2014.

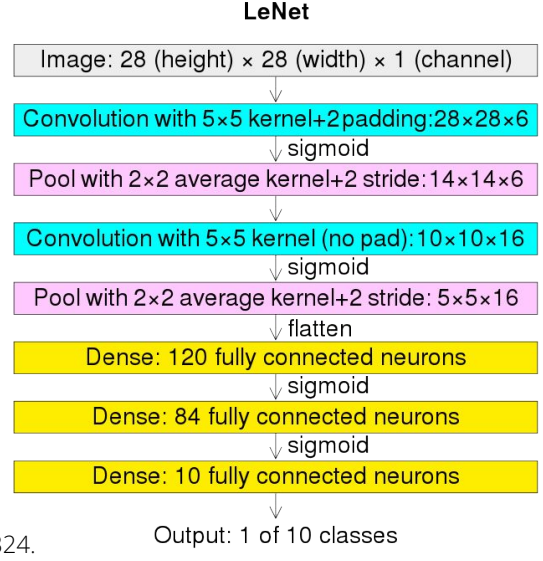
What does a CNN "see"? Receptive field



Convolutional Neural Networks: LeNet

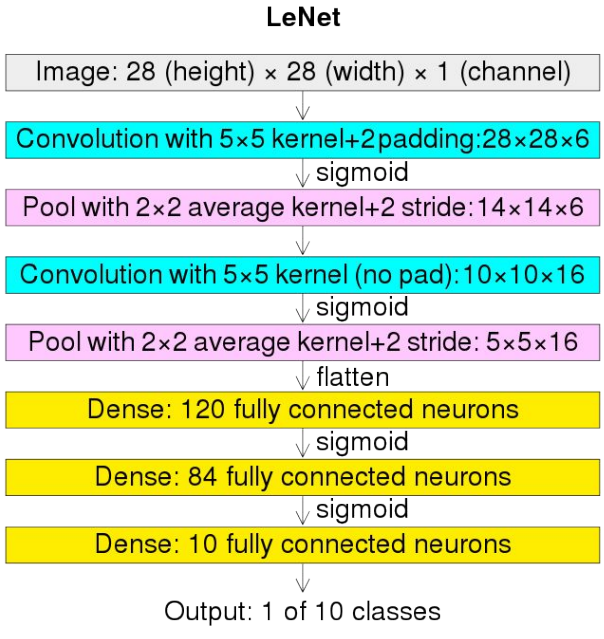


- 60,000 parameters



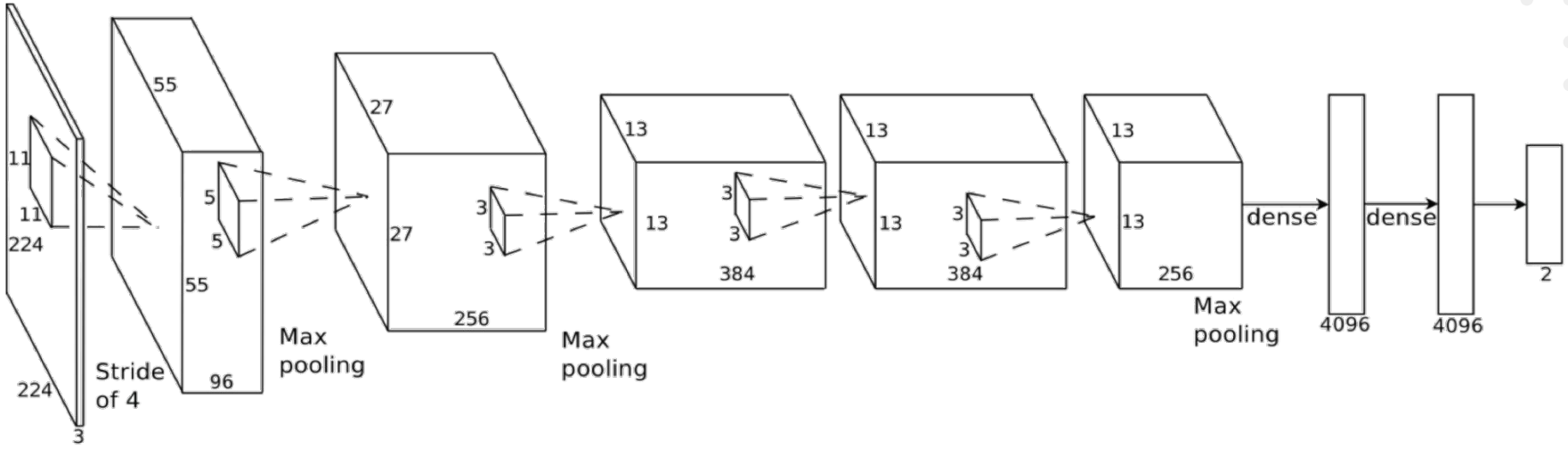
LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

Convolutional Neural Networks: LeNet



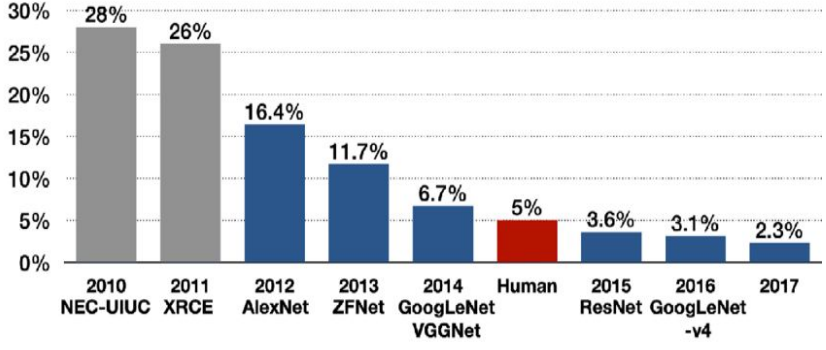

```python
# Pytorch: LeNet 5
class LeNet5(nn.Module):
    def __init__(self, num_classes):
        super(ConvNeuralNet, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=0),
            nn.BatchNorm2d(6),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = 2, stride = 2))
        self.layer2 = nn.Sequential(
            nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = 2, stride = 2))
        self.fc = nn.Linear(400, 120)
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(120, 84)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(84, num_classes)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer 2(out)
        out = out.reshape(out.size(0), -1)
        out = self.fc(out)
        out = self.relu(out)
        out = self.fc1(out)
        out = self.relu1(out)
        out = self.fc2(out)
        return out
```

Convolutional Neural Networks: AlexNet



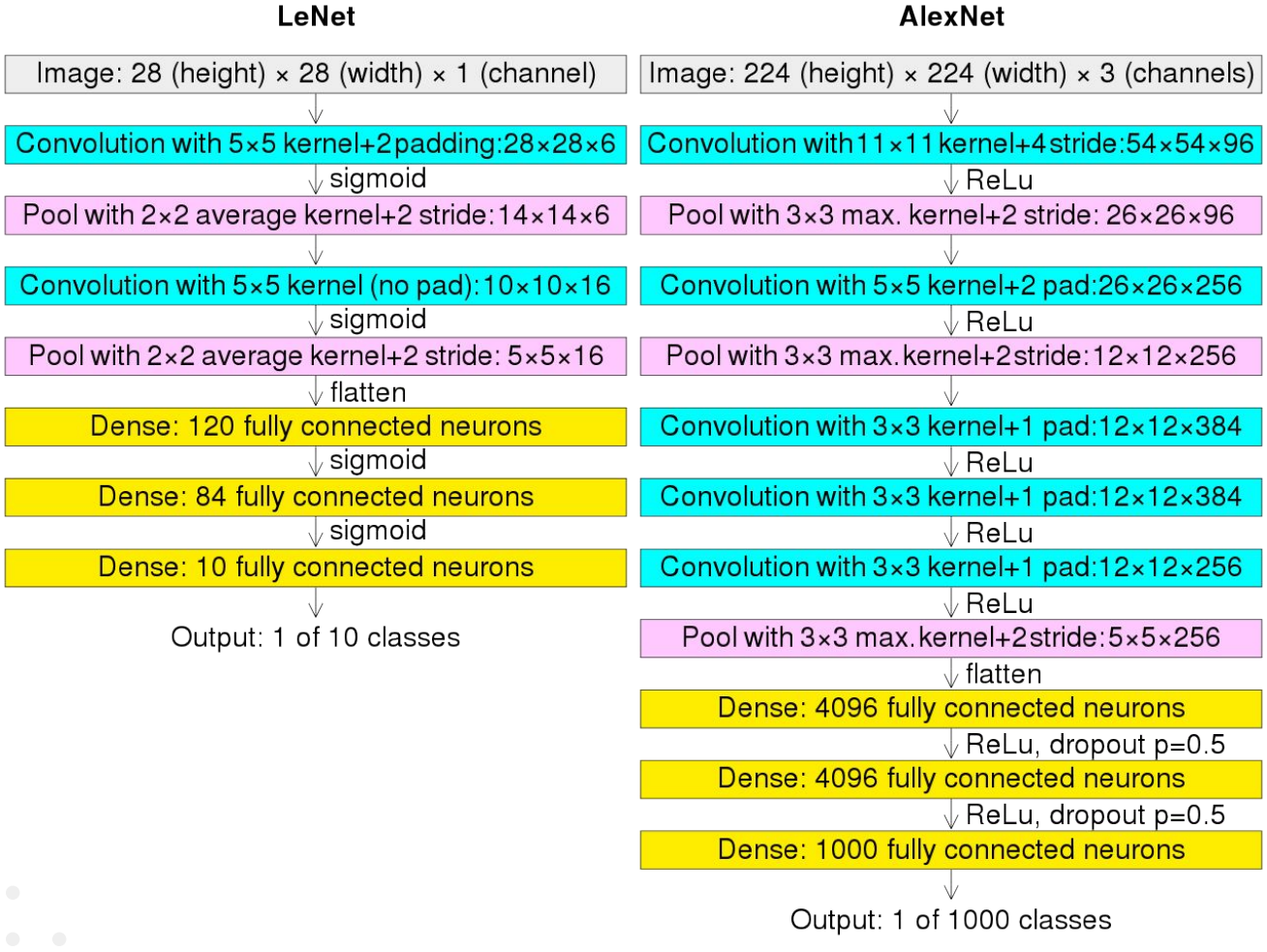
- ImageNet 2012
- 60 million parameters

Top 5 error for the algorithms that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

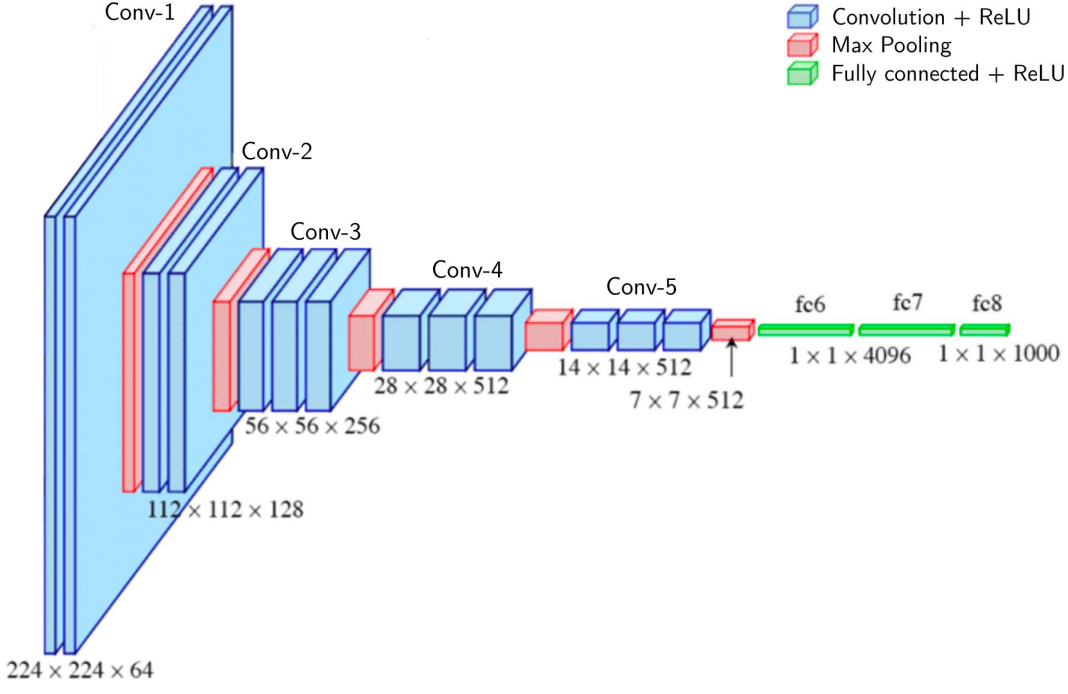


Krizhevsky, Alex et al. "ImageNet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012)

Convolutional Neural Networks: AlexNet



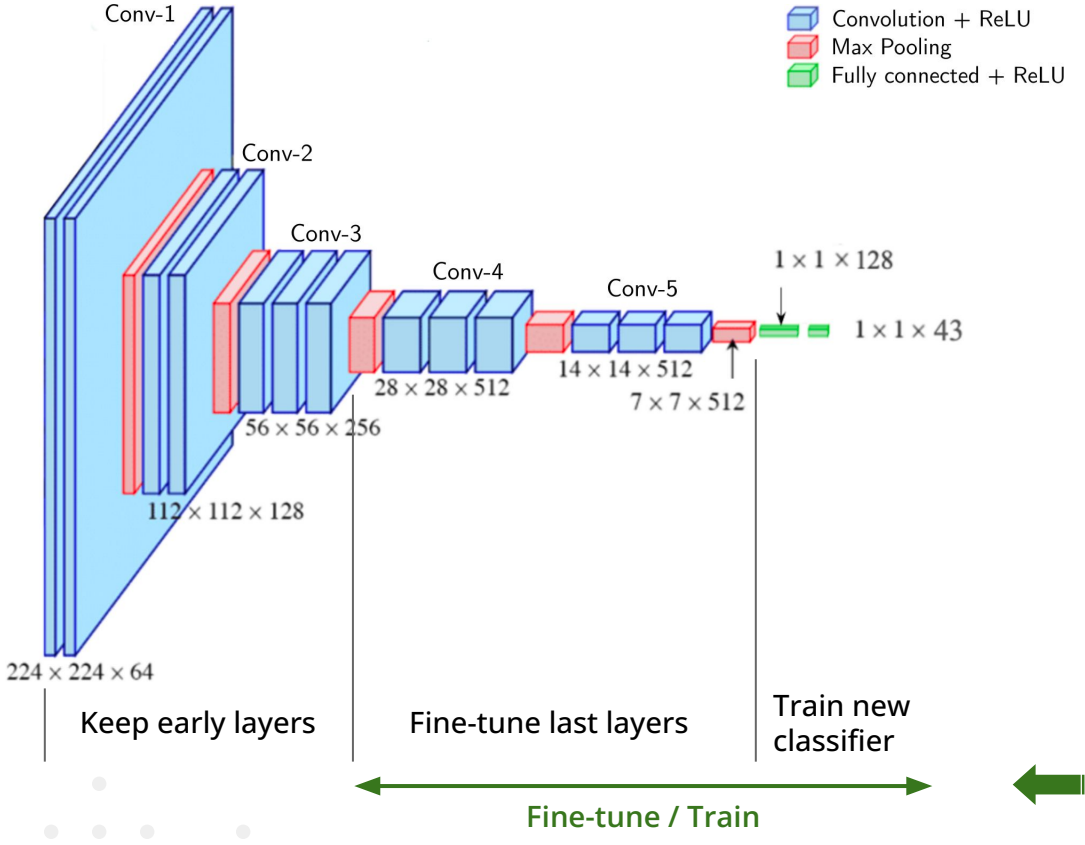
Convolutional Neural Networks: VGG



- Very simple architecture
  - 3x3 convolution
  - stride = 1, "same" padding
  - 2x2 max. pool
- Very large number of parameters: ~100 M
  - Features map: 256, 512, 512, ...
  - FCN: 4096, 4096, 1000
- Slow

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

Convolutional Neural Networks: VGG

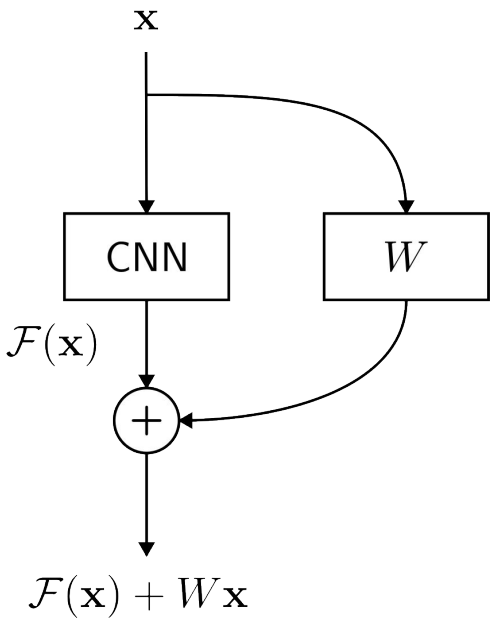


- Very simple architecture
  - 3x3 convolution
  - stride = 1, "same" padding
  - 2x2 max. pool
- Very large number of parameters: ~100 M
  - Features map: 256, 512, 512, ...
  - FCN: 4096, 4096, 1000
- Slow
-

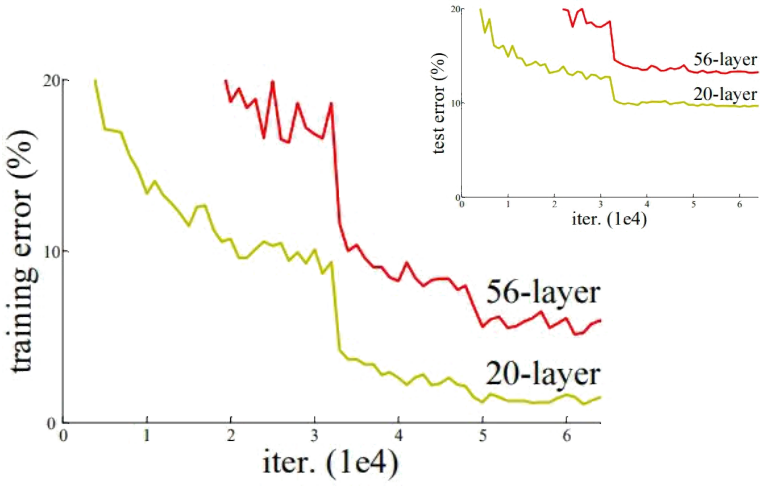
Transfer learning

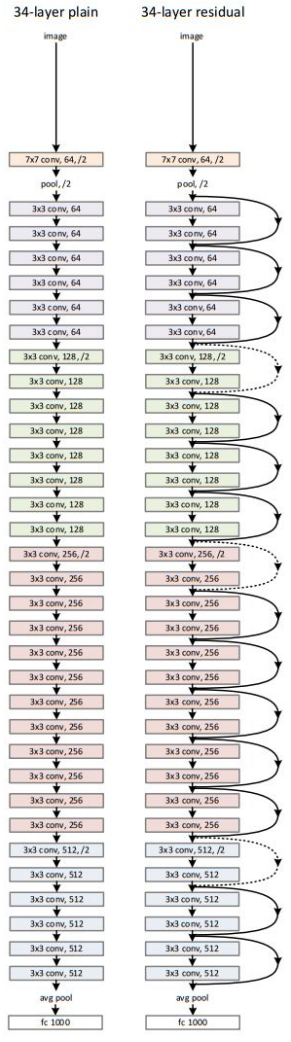
Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

Convolutional Neural Networks: Residual Network (ResNet)



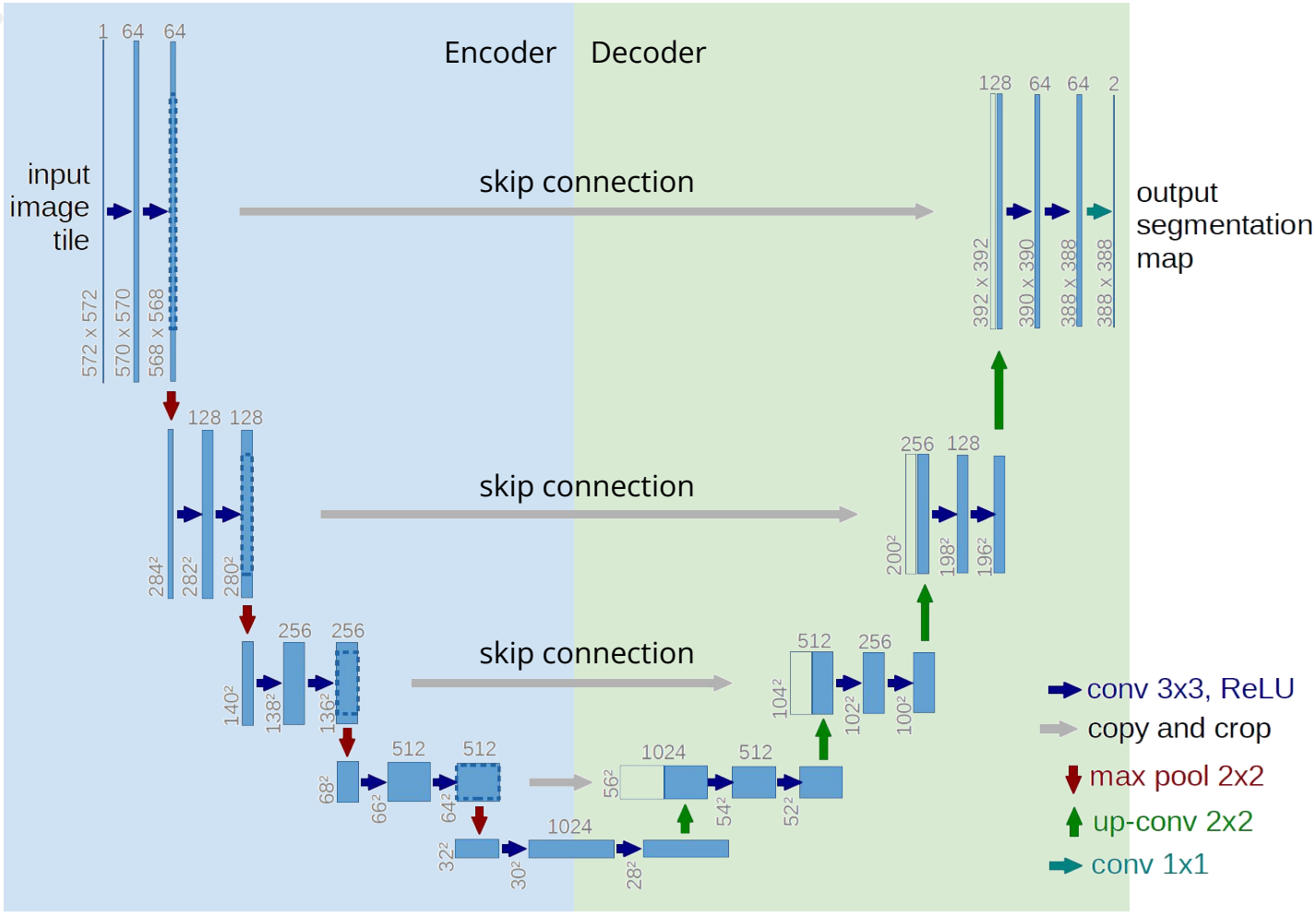
- Vanishing gradient:
  - Update proportional to gradient
  - No update
- Skip connection
  - Skip the *errors* by regularization
- ResNet-N: 18, 20, 34, 50, 101, 152, ...





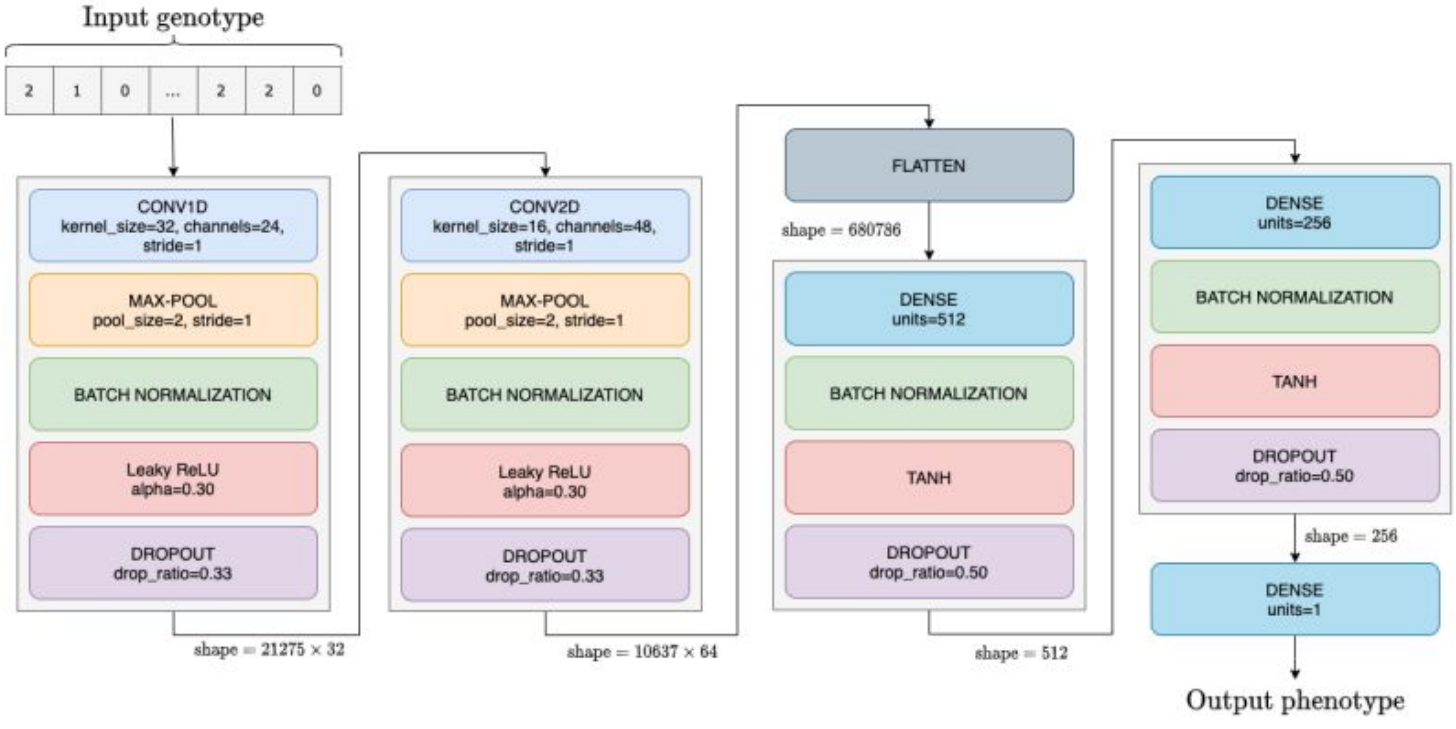
He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Convolutional Neural Networks: U-Net

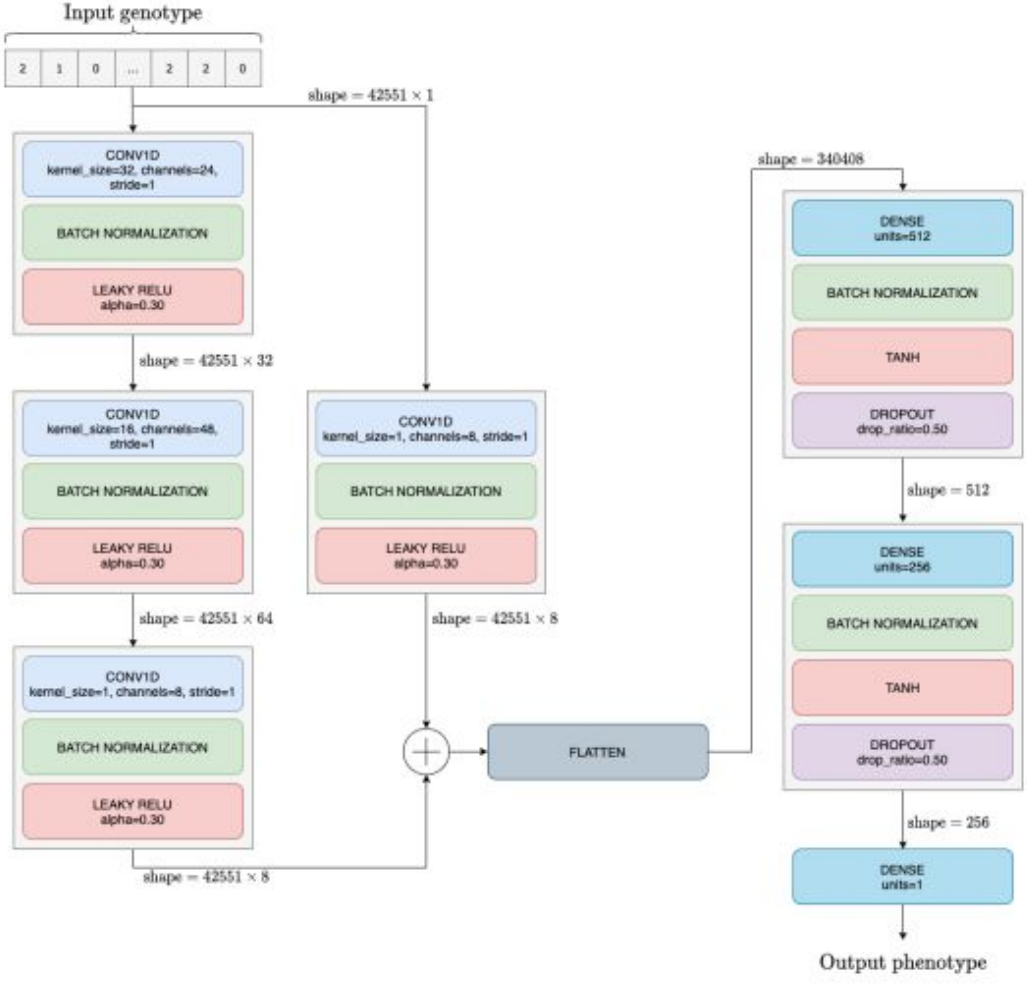


Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015.

Practicals


Input genotype
2
1
0
...
2
2
0
CONV1D
kernel_size=32, channels=24,
stride=1
MAX-POOL
pool_size=2, stride=1
BATCH NORMALIZATION
Leaky ReLU
alpha=0.30
DROPOUT
drop_ratio=0.33
shape = 21275 × 32
CONV2D
kernel_size=16, channels=48,
stride=1
MAX-POOL
pool_size=2, stride=1
BATCH NORMALIZATION
Leaky ReLU
alpha=0.30
DROPOUT
drop_ratio=0.33
shape = 10637 × 64
FLATTEN
shape = 680786
DENSE
units=512
BATCH NORMALIZATION
TANH
DROPOUT
drop_ratio=0.50
shape = 512
DENSE
units=256
BATCH NORMALIZATION
TANH
DROPOUT
drop_ratio=0.50
shape = 256
DENSE
units=1
Output phenotype

Practicals


Input genotype
2
1
0
...
2
2
0
shape = 42551 × 1
CONV1D
kernel_size=32, channels=24, stride=1
BATCH NORMALIZATION
LEAKY RELU
alpha=0.30
shape = 42551 × 32
CONV1D
kernel_size=16, channels=48, stride=1
BATCH NORMALIZATION
LEAKY RELU
alpha=0.30
shape = 42551 × 64
CONV1D
kernel_size=1, channels=8, stride=1
BATCH NORMALIZATION
LEAKY RELU
alpha=0.30
shape = 42551 × 8
CONV1D
kernel_size=1, channels=8, stride=1
BATCH NORMALIZATION
LEAKY RELU
alpha=0.30
shape = 42551 × 8
FLATTEN
shape = 340408
DENSE
units=512
BATCH NORMALIZATION
TANH
DROPOUT
drop_ratio=0.50
shape = 512
DENSE
units=256
BATCH NORMALIZATION
TANH
DROPOUT
drop_ratio=0.50
shape = 256
DENSE
units=1
Output phenotype

References

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H. T. (2012). Learning from data (Vol. 4, p. 4). New York: AMLBook.

Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc.".