Computer Vision

CVI620

Session 22
03/2025

What is Left?

4 sessions

1. Optimization and Loss Function
2. Code + Logistic Regression
3. ML and Images
4. Perceptron and Neural Networks
5. Deep Neural Networks
6. Convolution Neural Networks (CNN)
7. Advanced CNNs
8. Project
9. Segmentation
10. Introduction to object detection and image generation methods with AI
11. Project

Agenda

01 Review

02 Convolutional Neural Networks

02 Code

Review

Data preprocessing

- Resize
- Normalize
- Flatten
- Create labels
- Label encoding
- One-hot encoding
- Train-test split

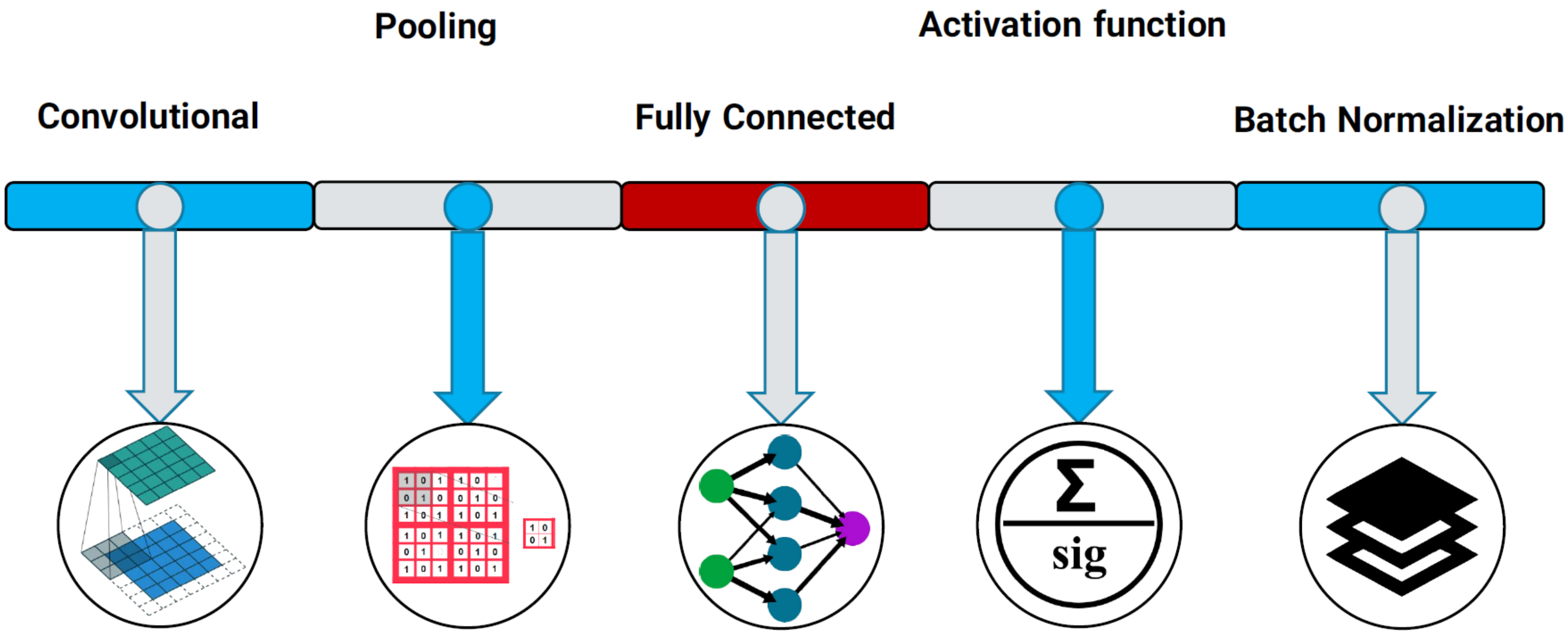
Review

Model

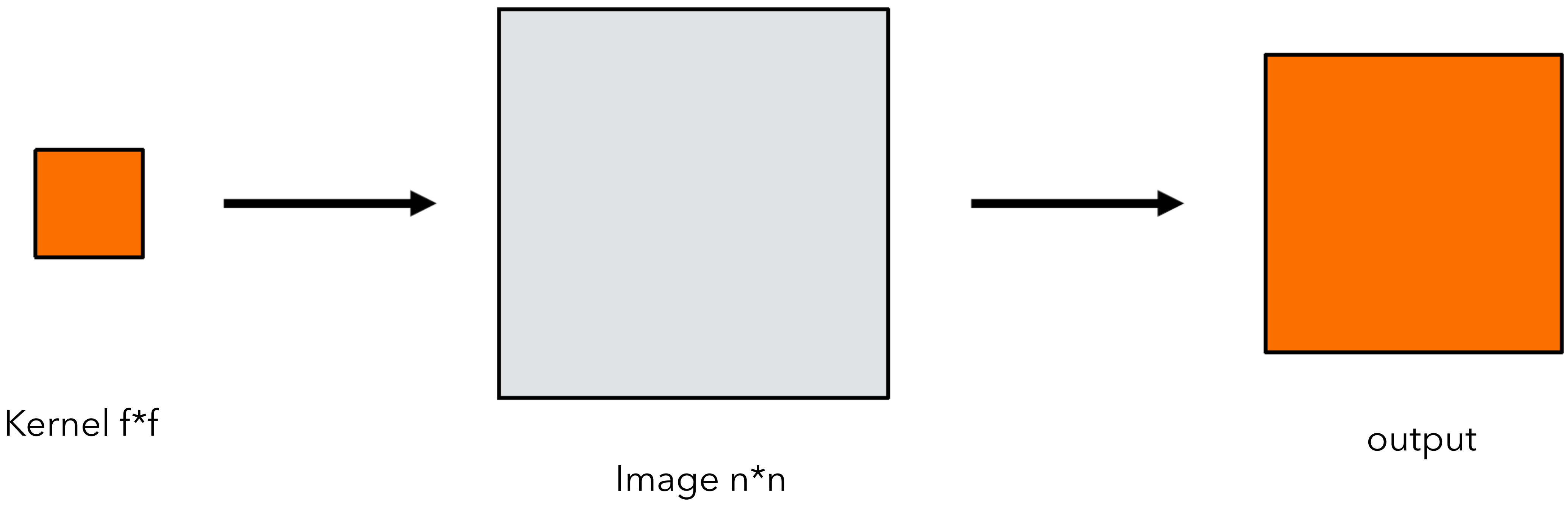
- Sequential
- Layers.Dense
- Activation: step, sigmoid, softmax, relu
- Optimzier: SGD, adam
- Loss: MAE, MSE, categorical_crossentropy
- batch_size
- epoch

Convolutional Neural Networks (CNNs)

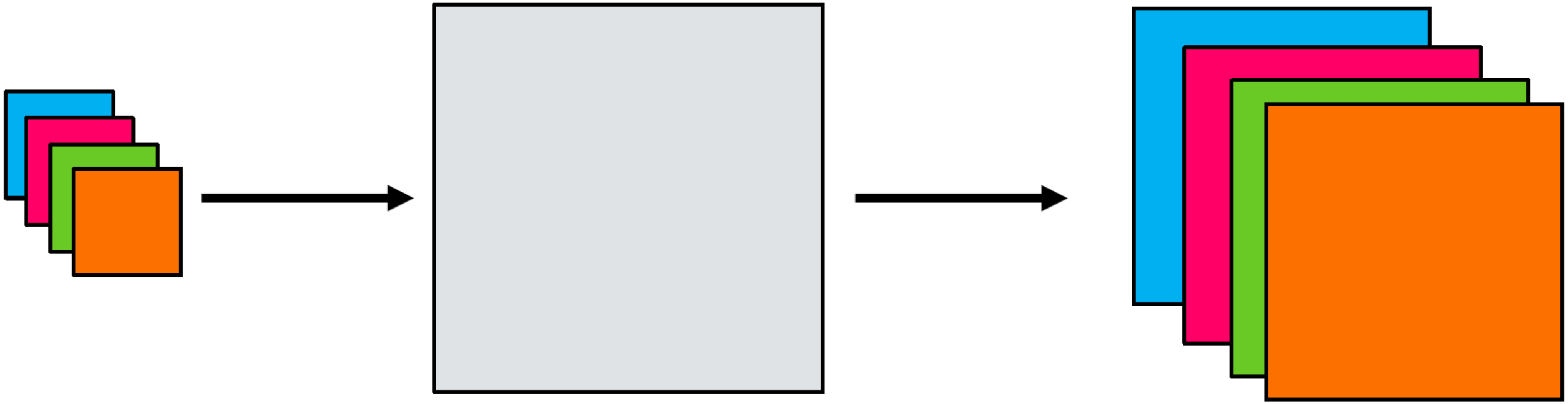
Layers in CNNs



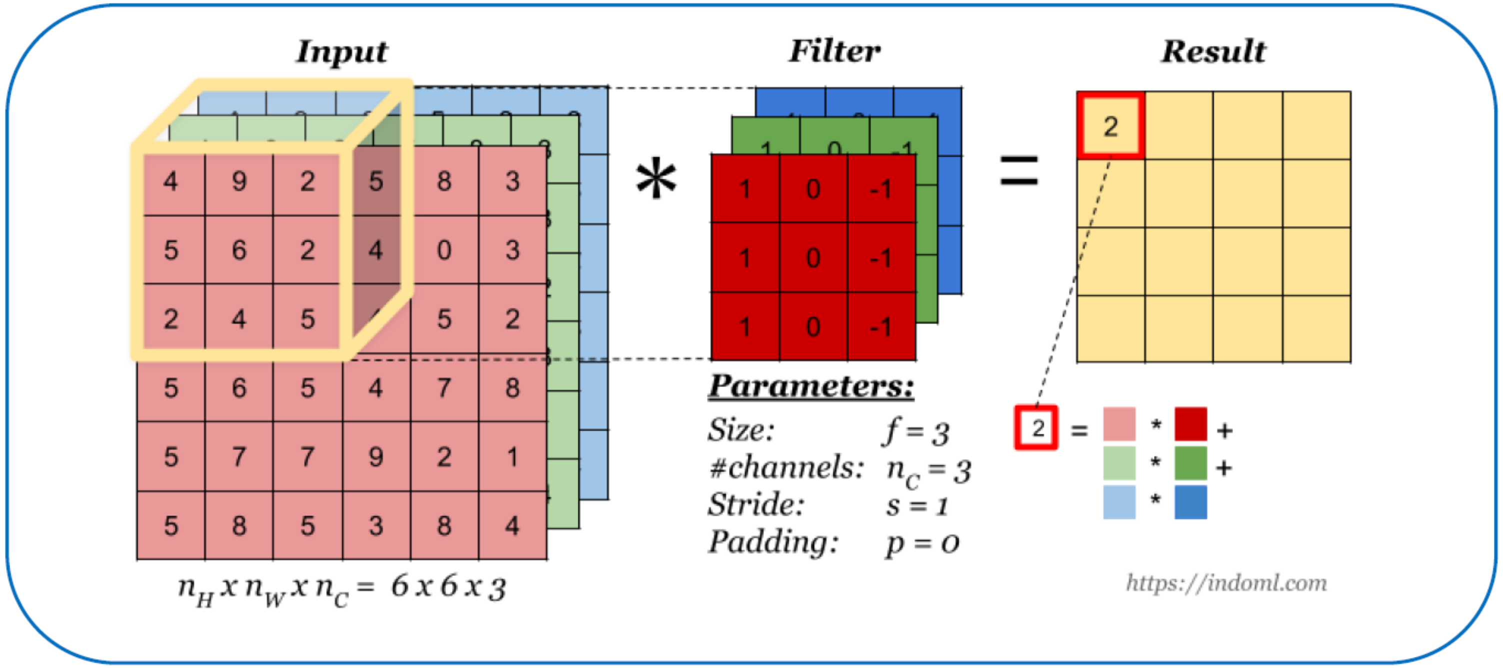
Convolution Layer



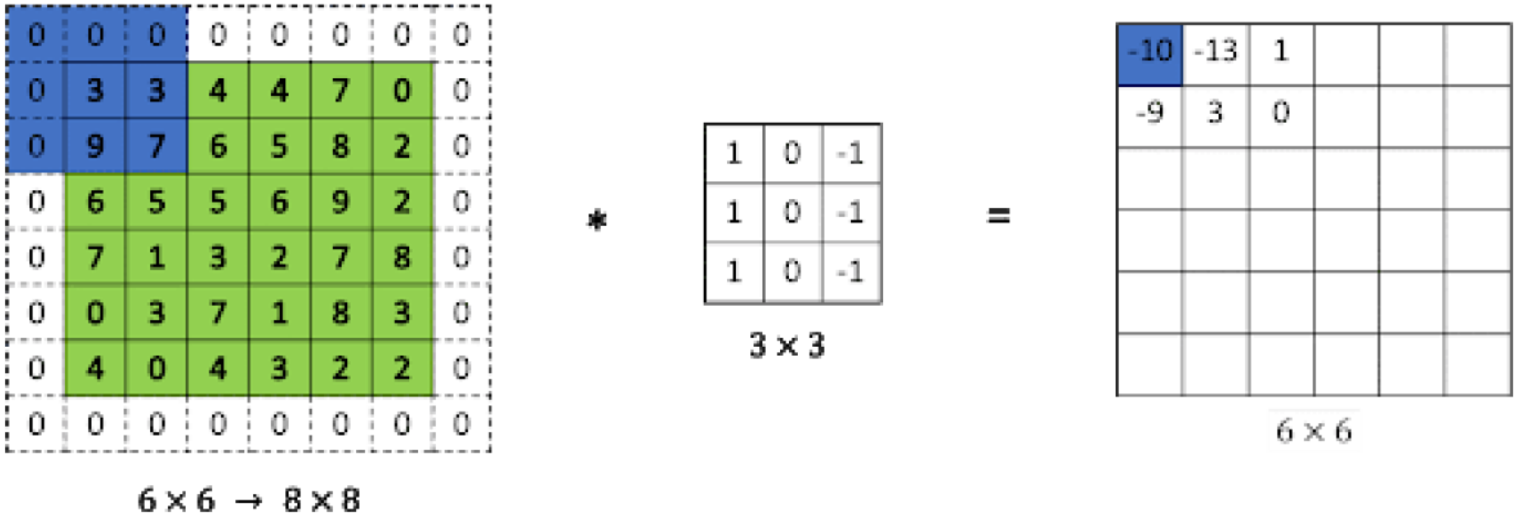
Multiple Convolutions



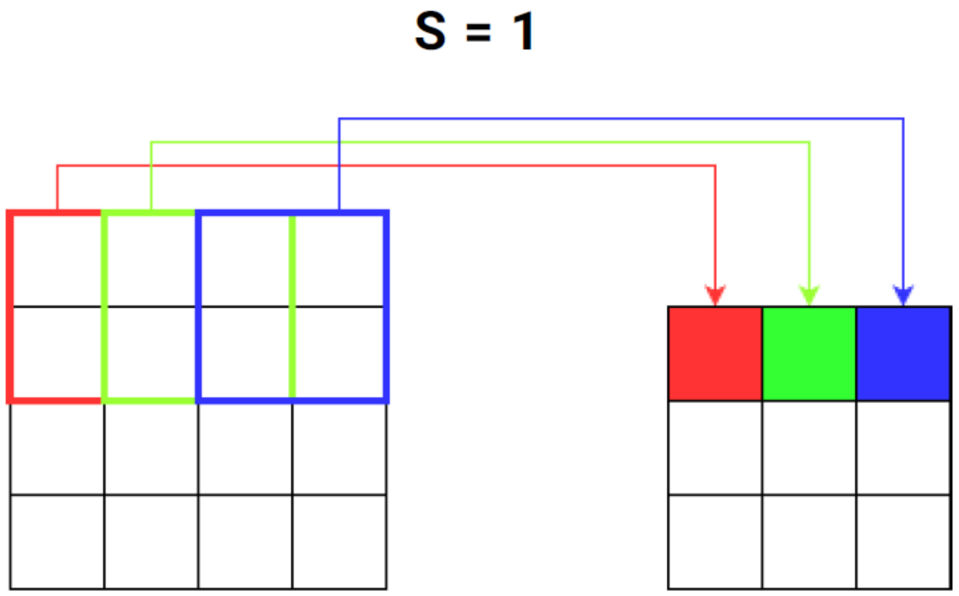
Convolution for Colored Images

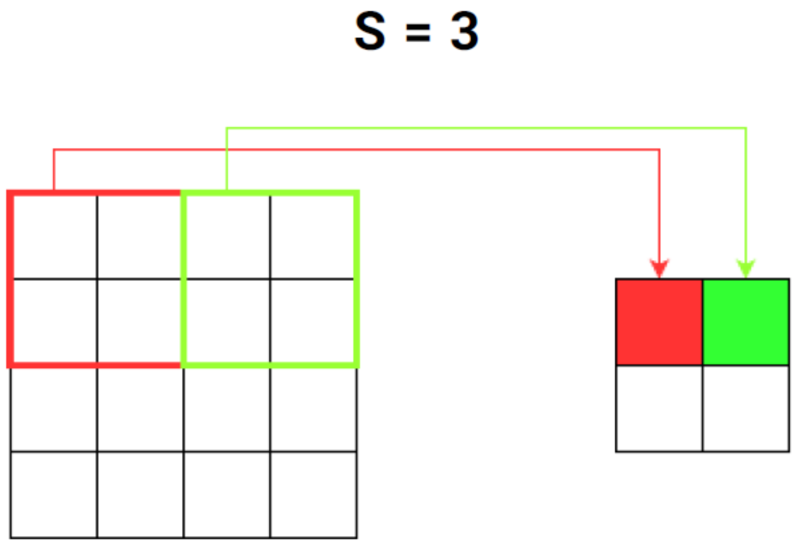


Padding

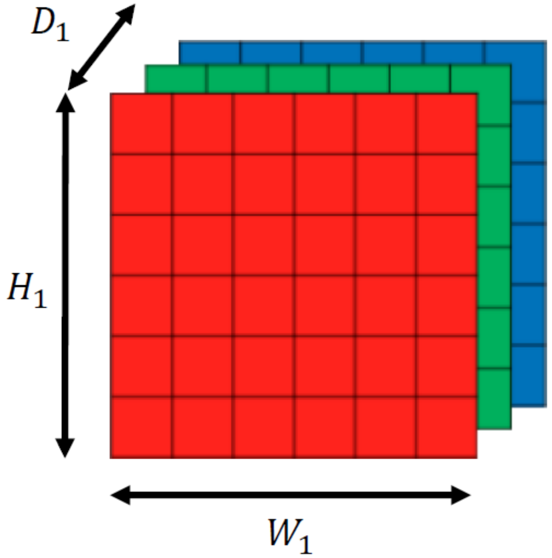

0 0 0 0 0 0 0 0
0 3 3 4 4 7 0 0
0 9 7 6 5 8 2 0
0 6 5 5 6 9 2 0
0 7 1 3 2 7 8 0
0 0 3 7 1 8 3 0
0 4 0 4 3 2 2 0
0 0 0 0 0 0 0 0
6 × 6 → 8 × 8
*
1 0 -1
1 0 -1
1 0 -1
3 × 3
=
-10 -13 1
-9 3 0
6 × 6

Stride





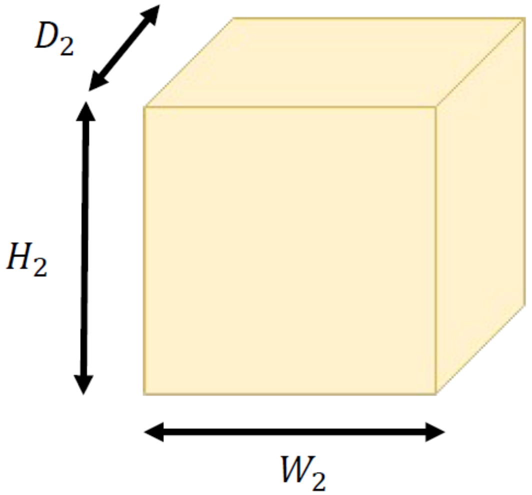
Post Convolution Dimensions



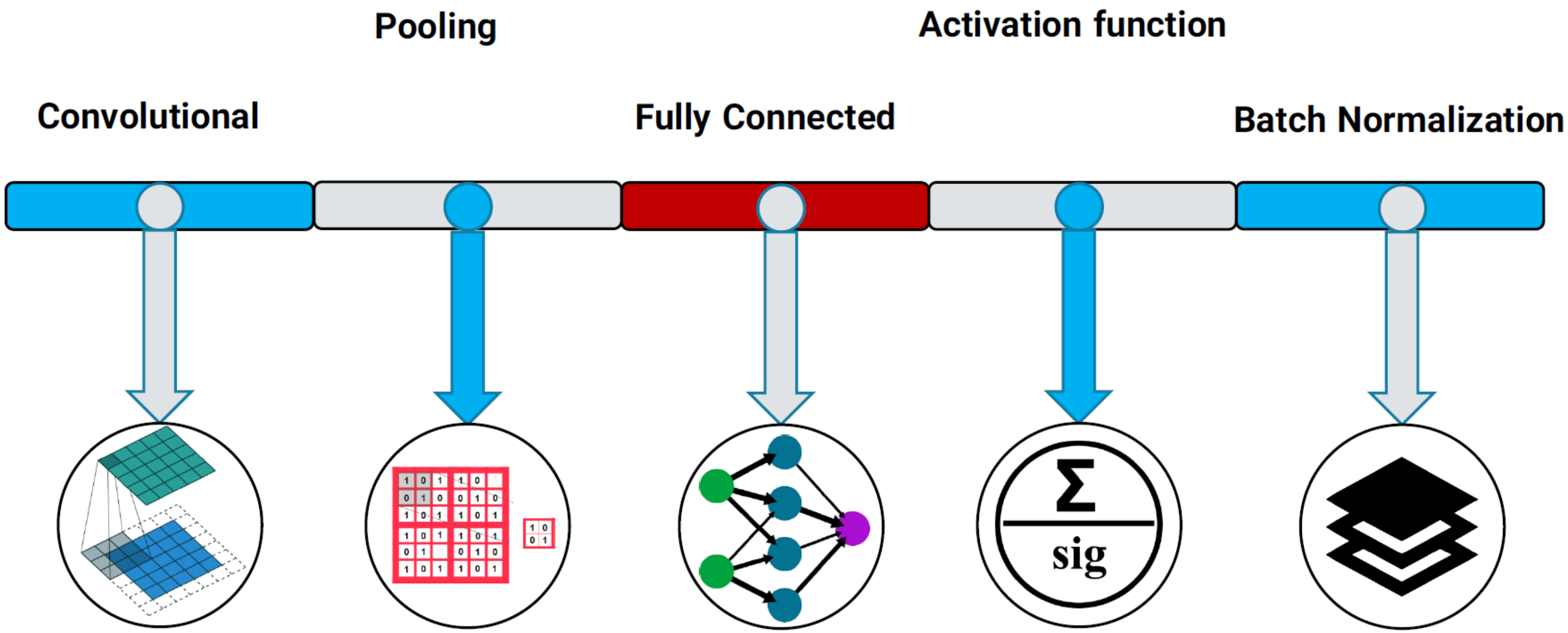
$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

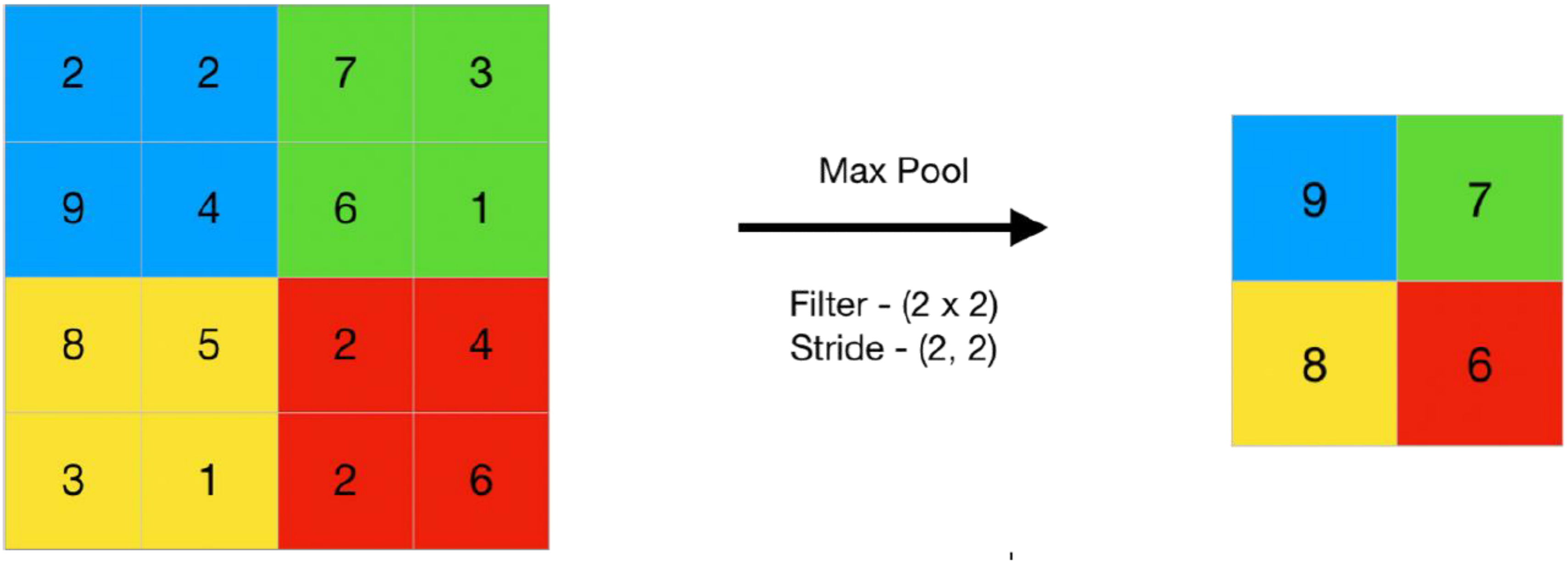
$$D_2 = K$$



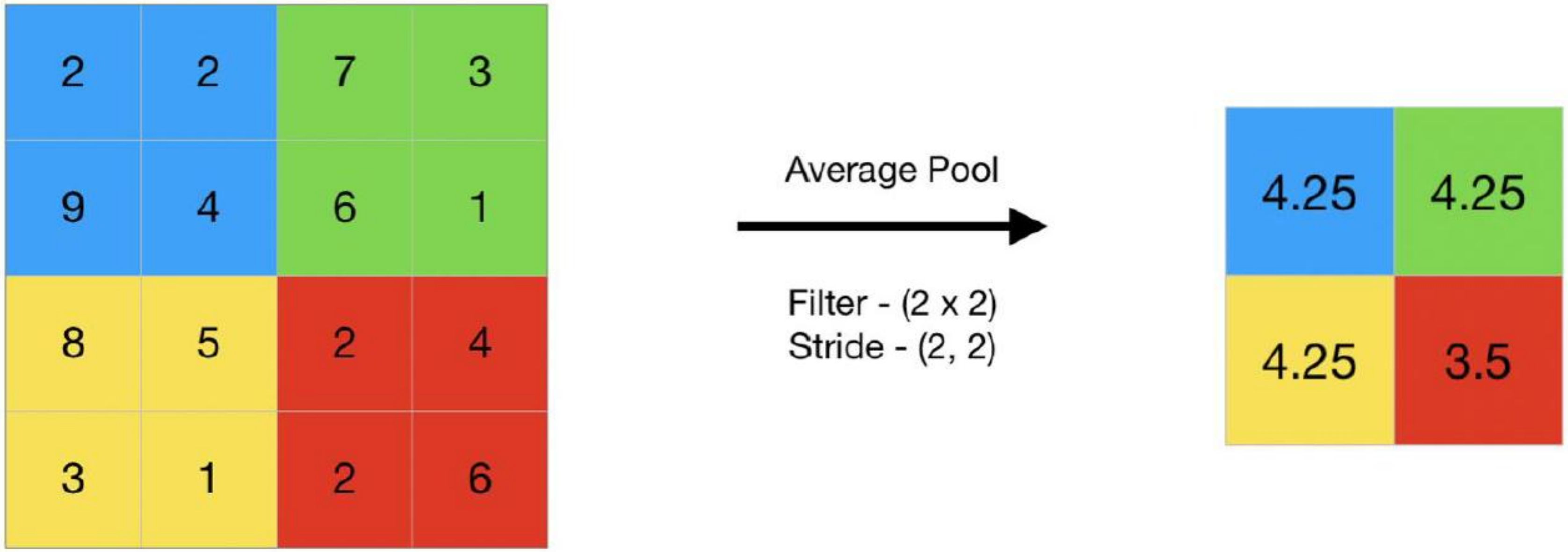
Layers in CNNs



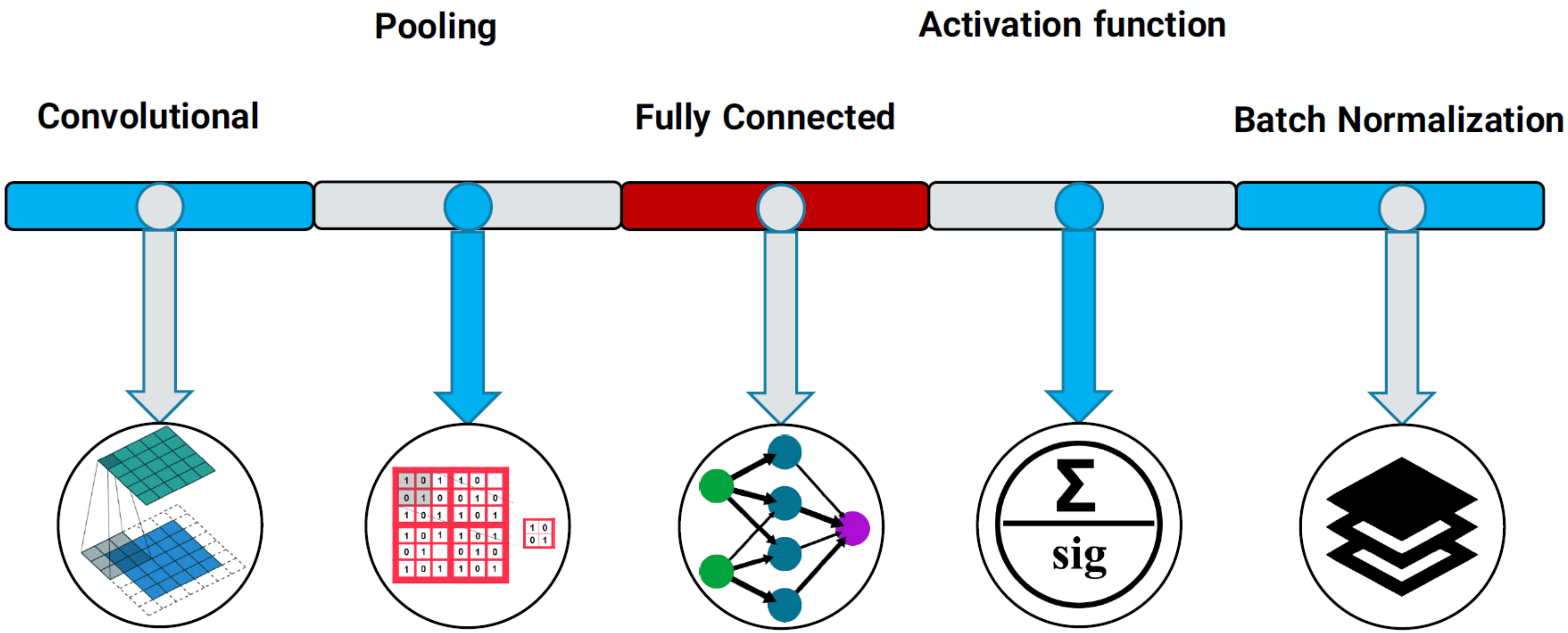
MaxPooling



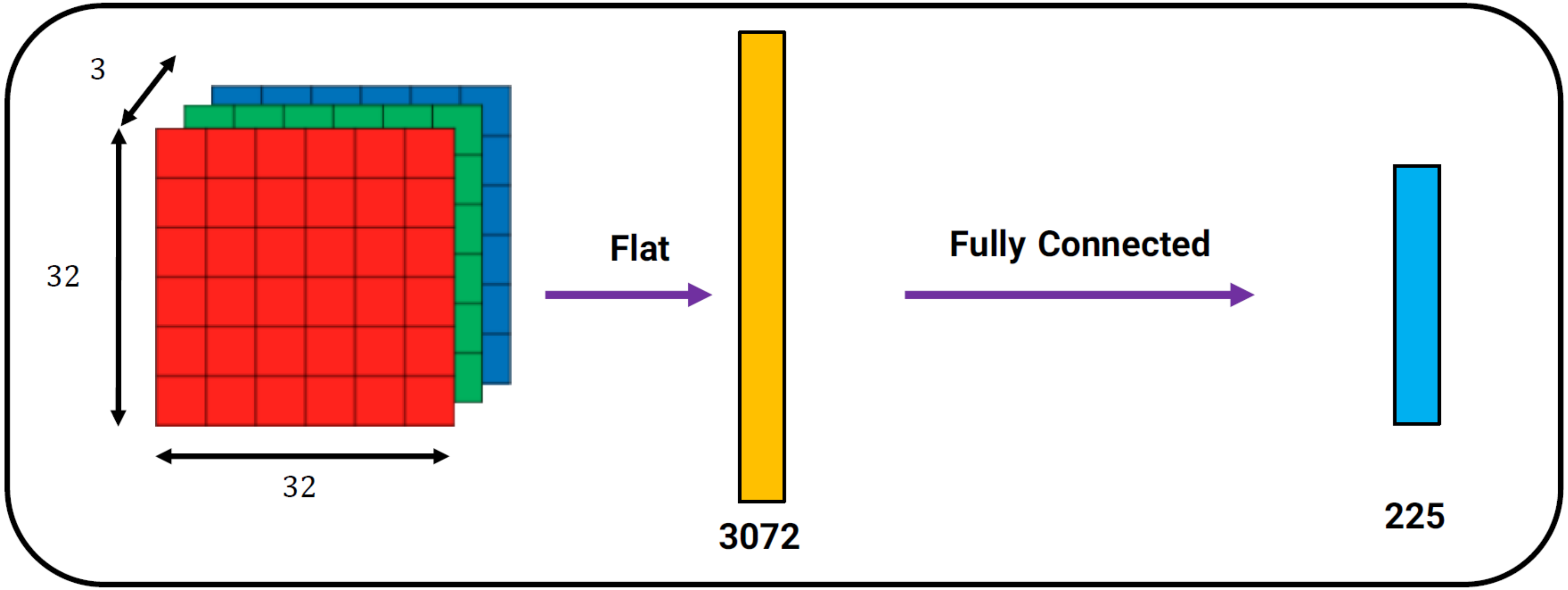
Average Pooling



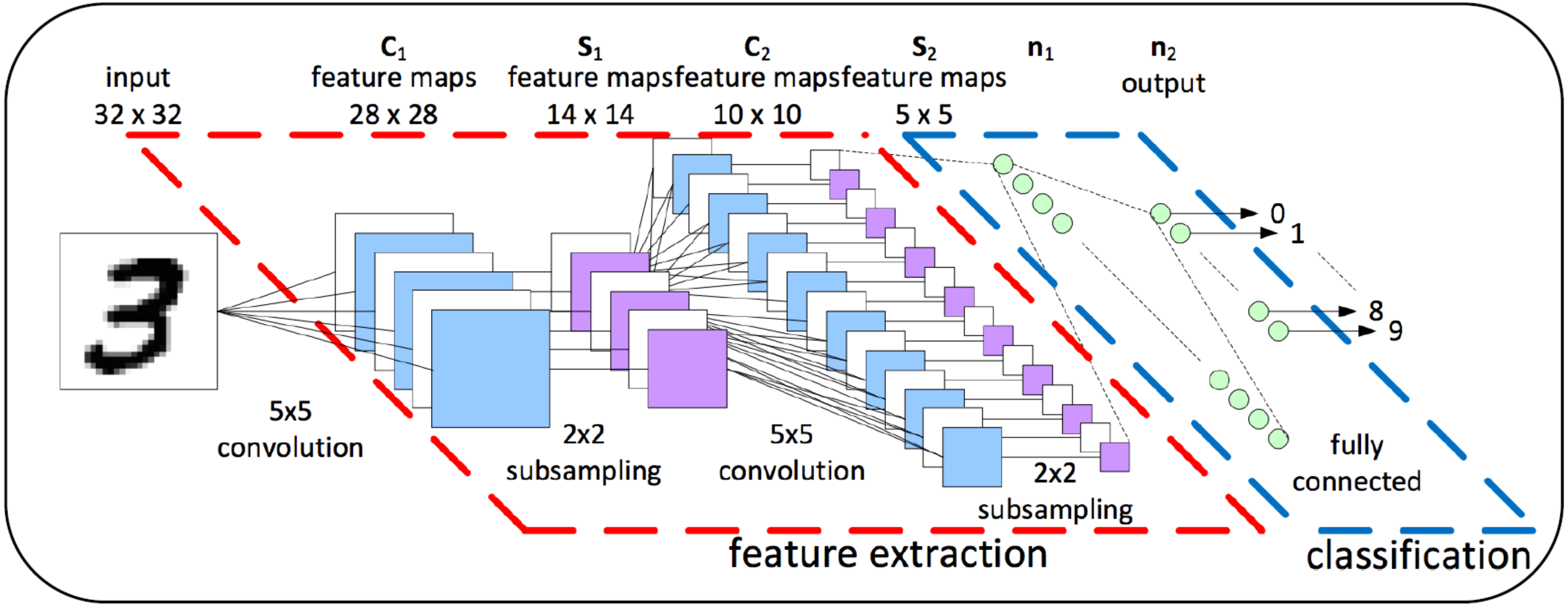
Layers in CNNs



MLP



CNNs



input
32 x 32
C_1
feature maps
28 x 28
S_1
feature maps
14 x 14
C_2
feature maps
10 x 10
S_2
feature maps
5 x 5
n_1
n_2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

Conv2D

```
tf.keras.layers.Conv2D(
    filters, kernel_size, strides=(1, 1), padding='valid', data_format=None,
    dilation_rate=(1, 1), groups=1, activation=None, use_bias=True,
    kernel_initializer='glorot_uniform', bias_initializer='zeros',
    kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, bias_constraint=None, **kwargs
)
```

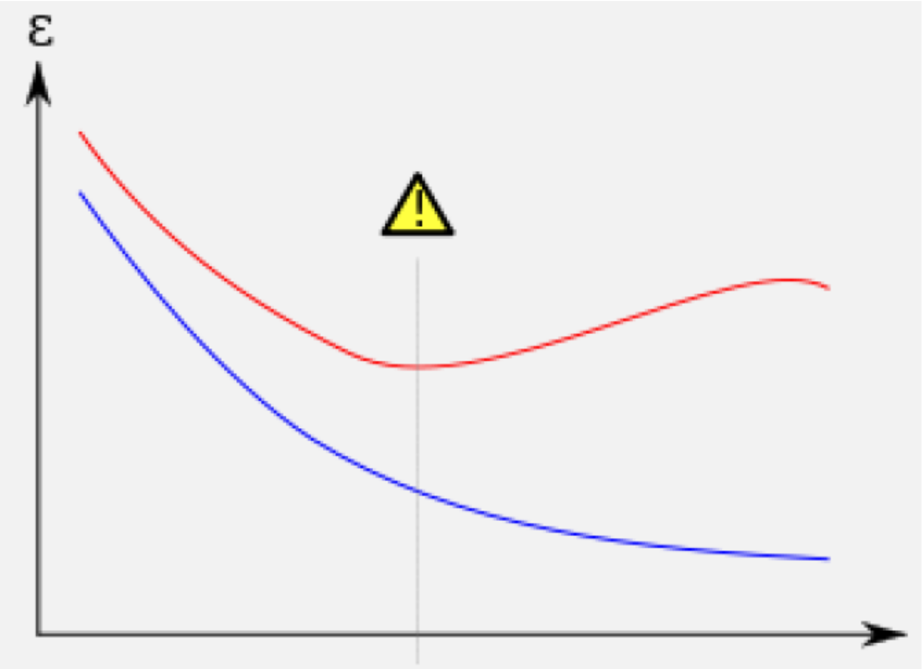
```python
net = models.Sequential([
                        layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
                        layers.MaxPool2D(),
                        layers.Conv2D(32, (3,3), activation='relu'),
                        layers.MaxPool2D(),
                        layers.Flatten(),
                        layers.Dense(100, activation='relu'),
                        layers.Dense(2, activation='softmax')
                        ])


net.compile(optimizer='SGD',
            loss='categorical_crossentropy',
            metrics='accuracy')

H = net.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=64, epochs=10)


net.save('S22/CNN_classifier.h5')
```

Overfitting



- Model works good on train data but not on test data.
- It memorizes and not generalize

Batch Normalization

We normalized the input data.

Why not normalizing values after each layer?

A technique to normalize activations in a neural network, layer by layer, during training.

- Stabilizes learning
- Speeds up convergence
- Reduces internal covariate shift
- Allows higher learning rates

Data Augmentation

- A technique to increase the size and diversity of training data by applying random transformations.
- More robust models with better performance on unseen data.

Rotation




Width Shift




Brightness

Shear

Zoom




ImageDataGenerator

- A tool to apply data augmentation

```
aug = ImageDataGenerator(rotation_range=30,
                         width_shift_range=0.1,
                         height_shift_range=0.1,
                         shear_range=0.2,
                         zoom_range=0.2,
                         horizontal_flip=True,
                         fill_mode="nearest")
```

Learning Rate

A hyperparameter that controls how much the model updates its weights in response to the loss gradient.

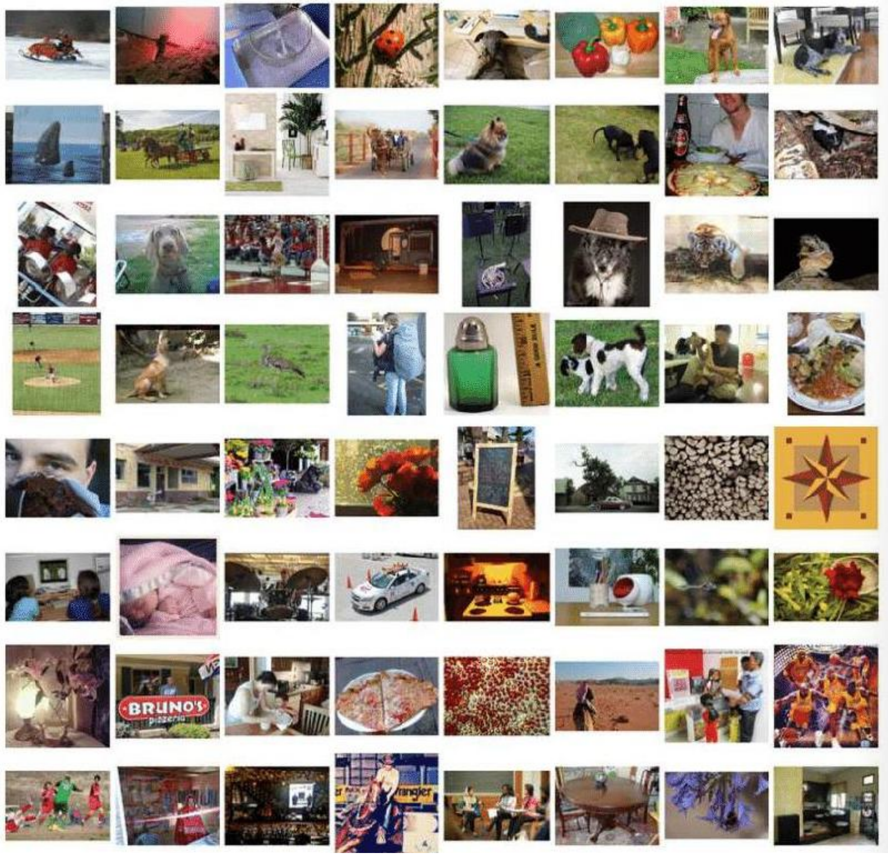
- Too high → Model diverges
- Too low → Slow convergence
- Just right → Fast & stable training

```
from keras.optimizers import SGD
...
opt=SGD(lr=..., decay=...)
net.compile(..., optimizer=opt)
```

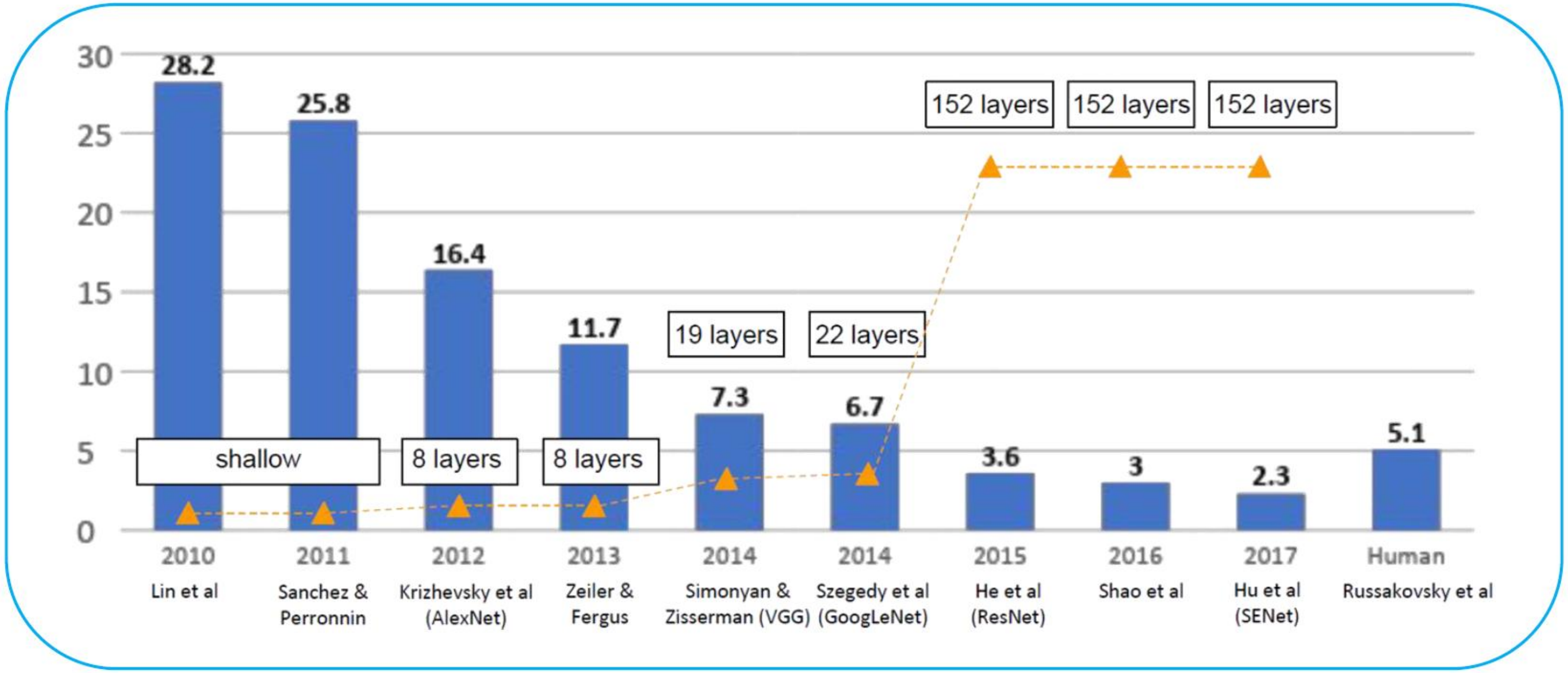
ILSVRC Challenge

- ImageNet dataset
- 1.2 m images with 1000 classes

- Classification
- Single object detection
- Object detection

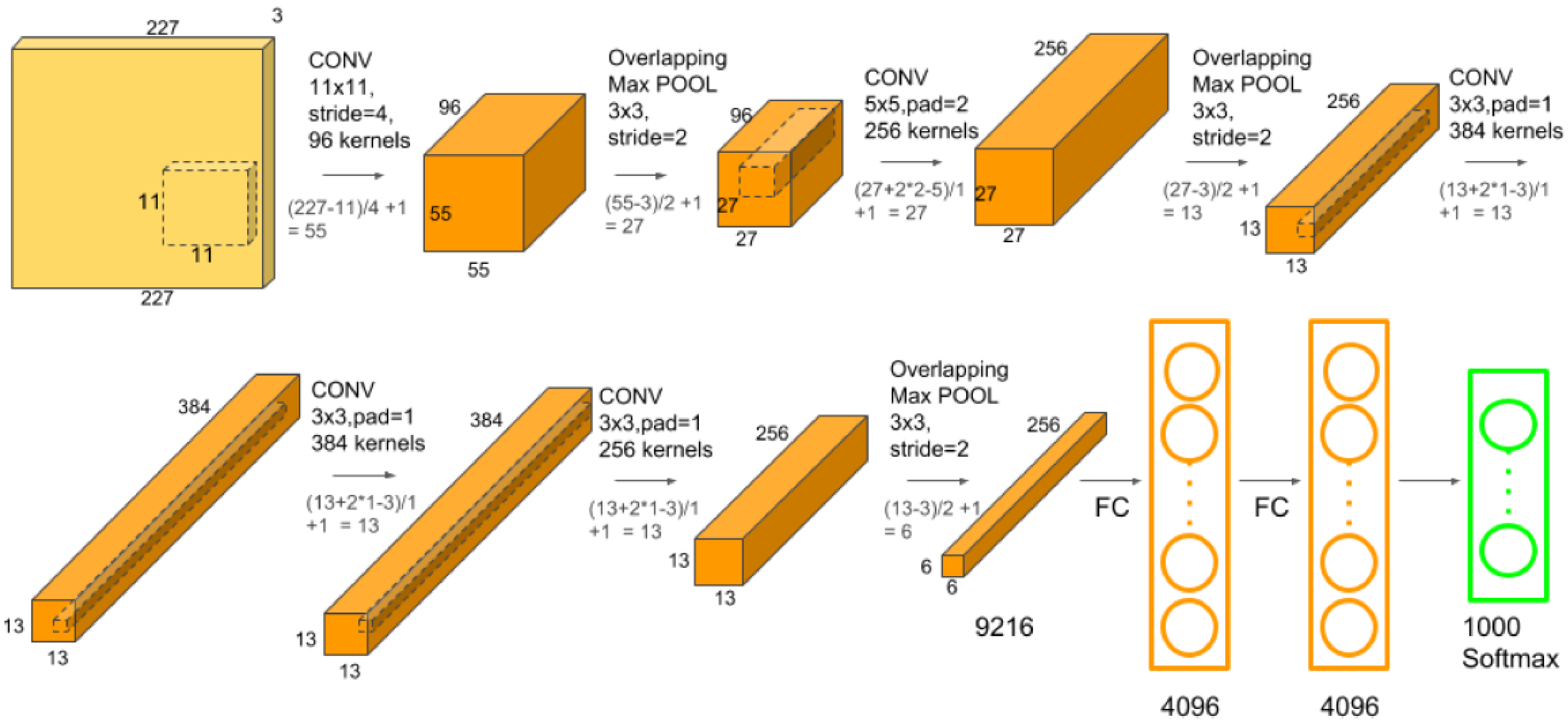


Results

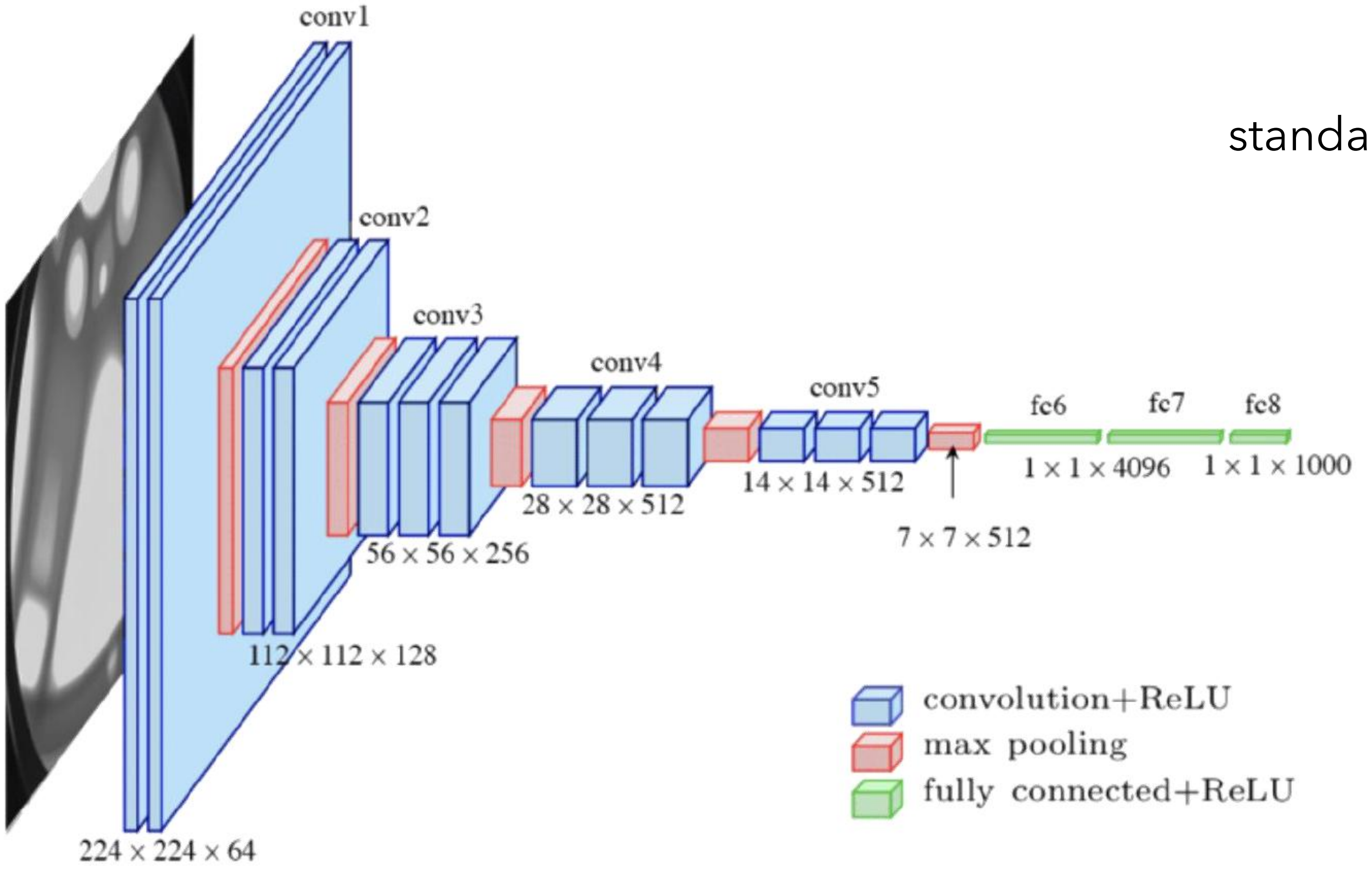


30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
3
2.3
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
152 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
2016
Shao et al
2017
Hu et al (SENet)
Human
Russakovsky et al

AlexNet


227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

VGG



standardize