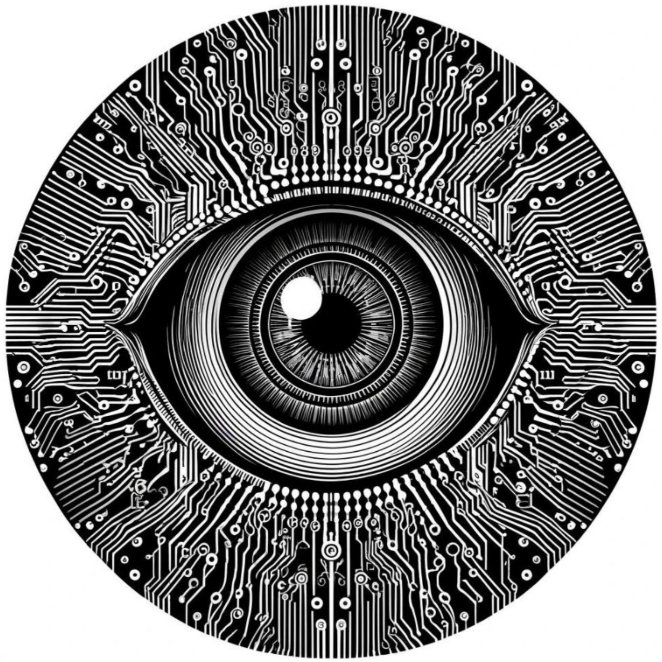
Pooling in Neural Networks



Antonio Rueda-Toicen

SPONSORED BY THE

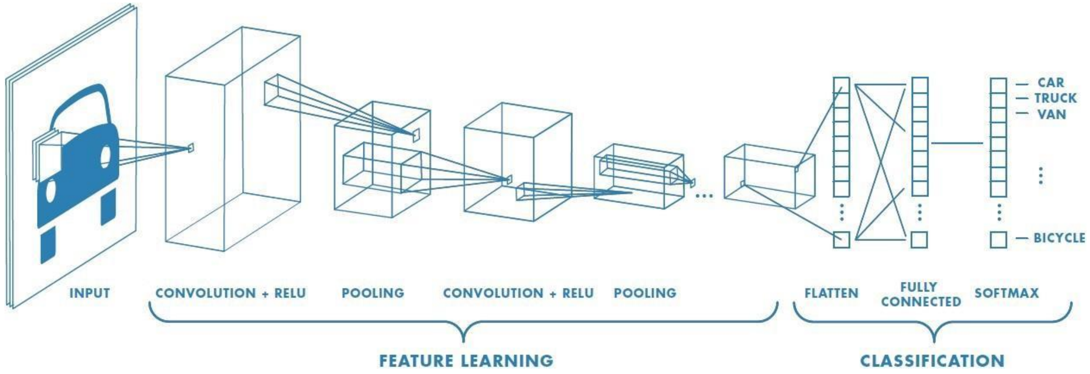
Federal Ministry
of Education
and Research

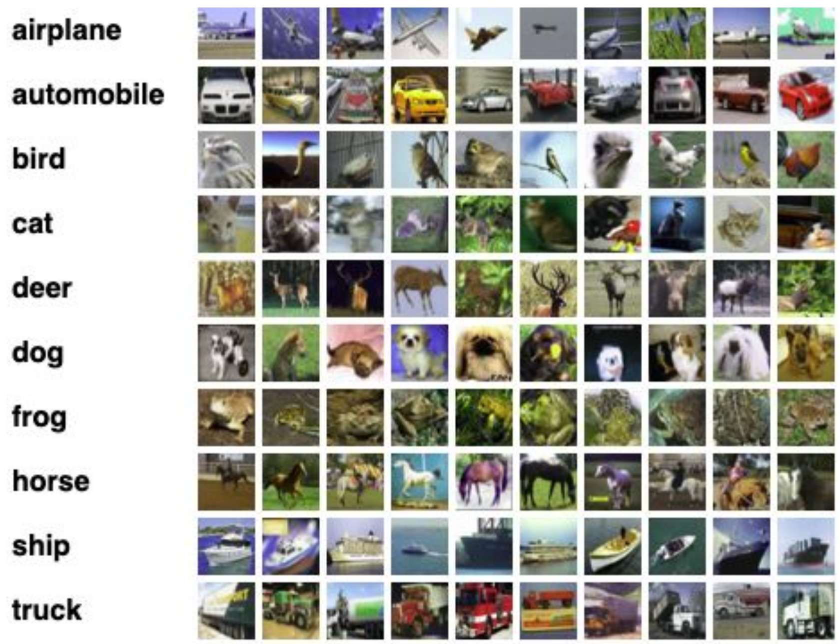
Learning goals

- Explore image downsampling and reduction of network parameters with mean, max, and global pooling

Architecture of a convolutional network with pooling

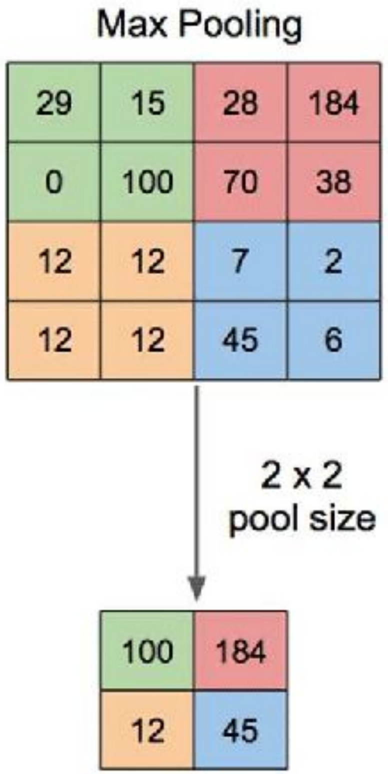
```
nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Flatten(),
            nn.Linear(32 * 8 * 8, 120),
            nn.ReLU(),
            nn.Linear(120, 84),
            nn.ReLU(),
            nn.Linear(84, 10), # 10 classes, we are working with CIFAR 10
        )
```



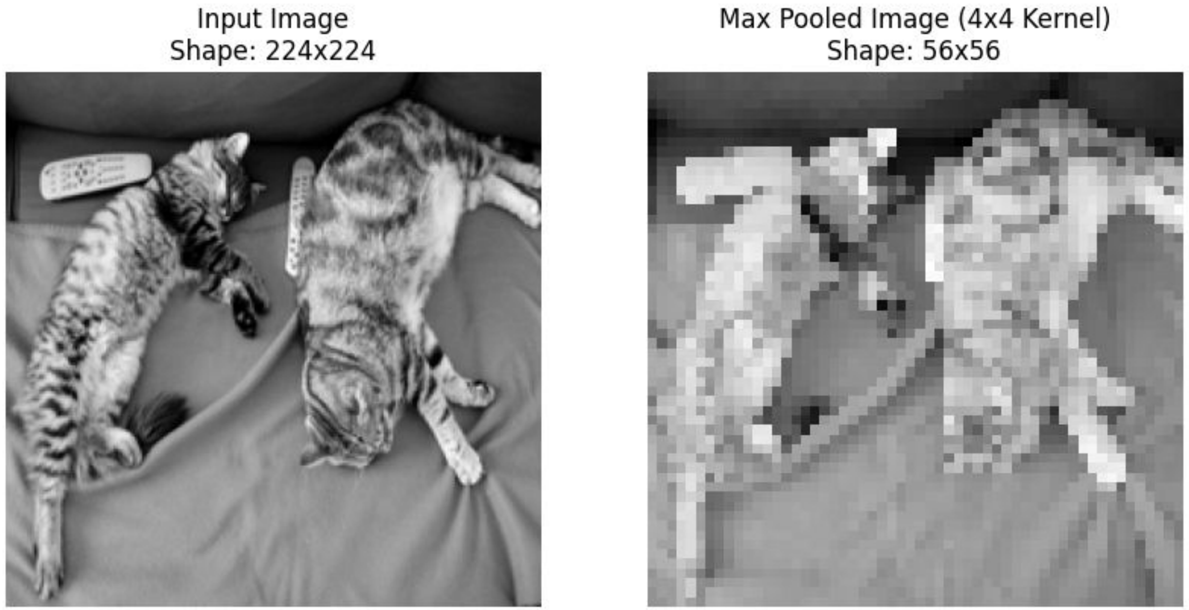


[The CIFAR-10 dataset]

Max Pooling

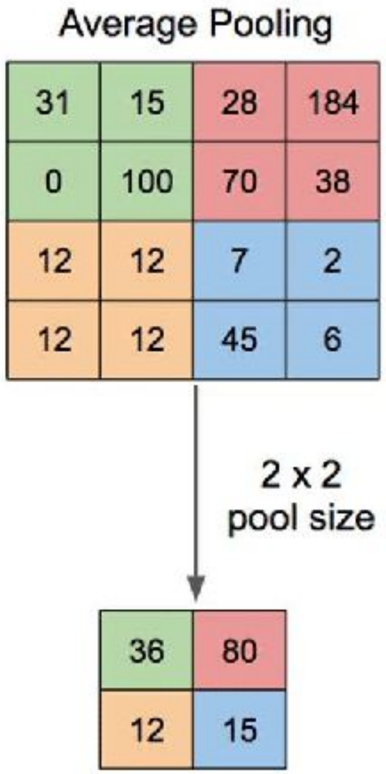


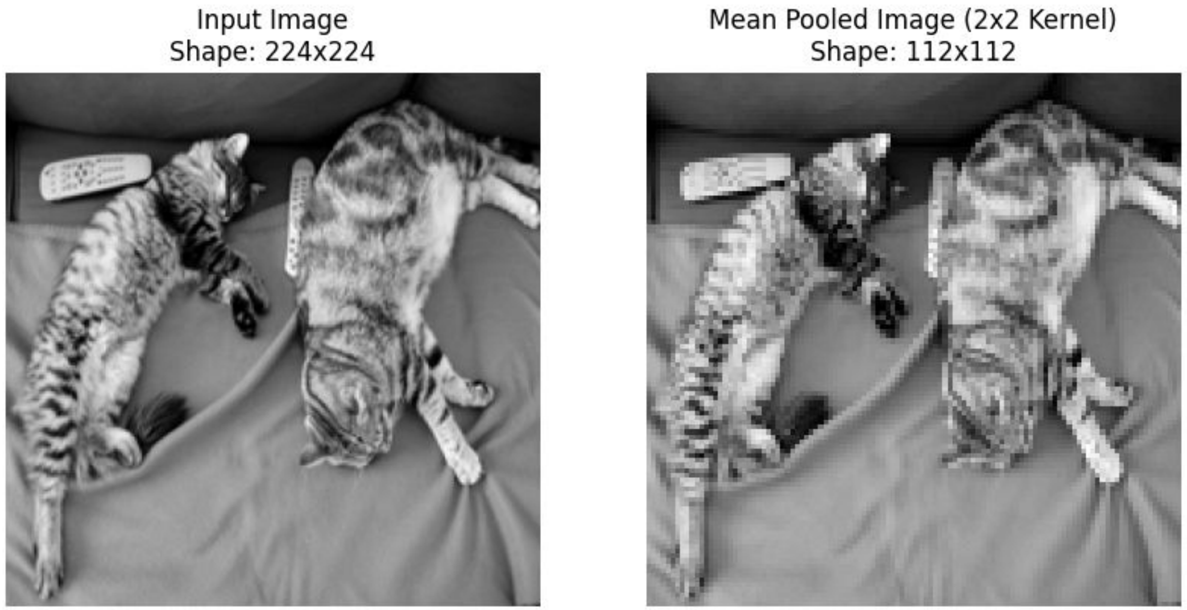
Max Pooling
29
15
28
184
0
100
70
38
12
12
7
2
12
12
45
6
2 x 2
pool size
100
184
12
45



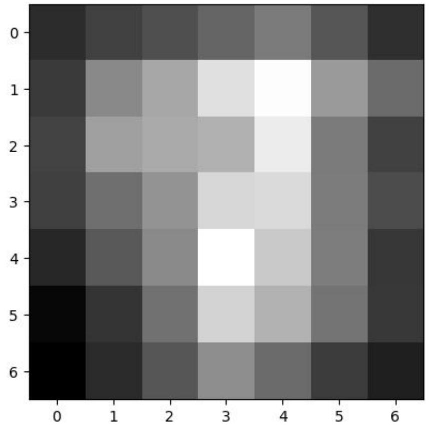
Input Image
Shape: 224x224
Max Pooled Image (4x4 Kernel)
Shape: 56x56

Mean Pooling

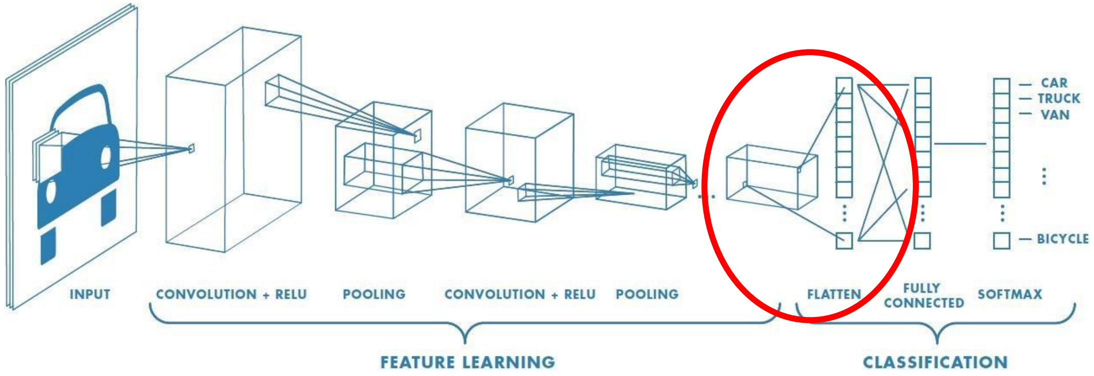




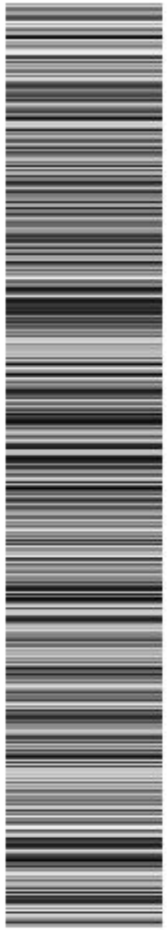
Global pooling aka adaptive pooling




```python
import torch.nn as nn


class GlobalPooling(nn.Module):
    def __init__(self):
        super().__init__()
        # Computes the average value of each activation map
        self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
        # Selects the max value of each activation map
        self.max_pool = nn.AdaptiveMaxPool2d((1, 1))

    def forward(self, x):
        avg_pooled = self.avg_pool(x).flatten(1)
        max_pooled = self.max_pool(x).flatten(1)
        # Concatenates both poolings to produce a feature vector
        return torch.cat([avg_pooled, max_pooled], dim=1)
```

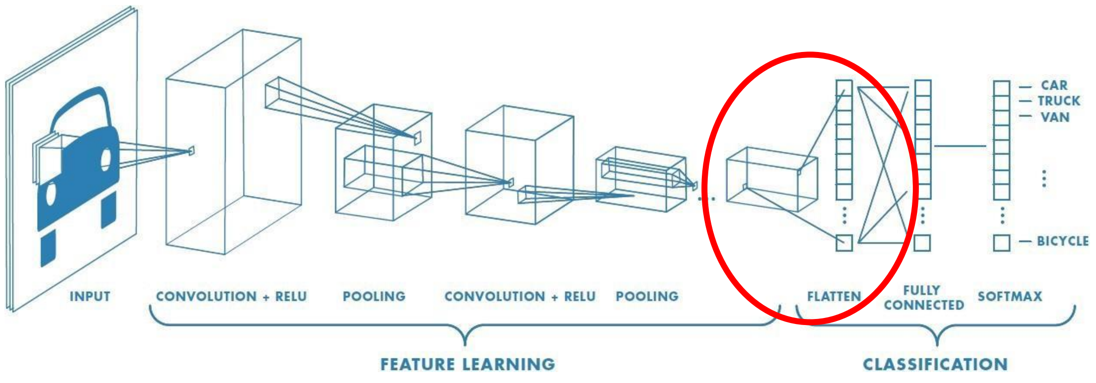


Feature vector

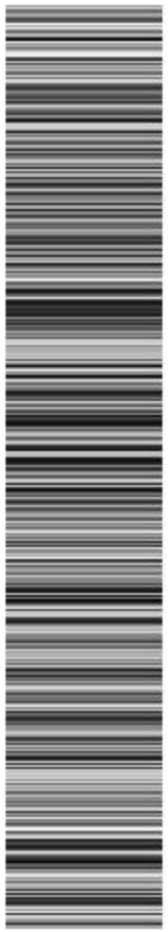


Using global pooling to reduce parameters

```python
import torch
import torch.nn as nn

model = nn.Sequential(
    nn.Conv2d(3, 16, kernel_size=5, stride=1, padding=2),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),

    nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=2),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),

    GlobalPooling(),  # This replaces the Flatten layer

    nn.Linear(64, 120),
    nn.ReLU(),
    nn.Linear(120, 84),
    nn.ReLU(),
    nn.Linear(84, 10)
)
```



Feature vector



Summary

Pooling is a way to compress information

- Pooling allows us to do lossy compression while retaining important visual features
- Convolutions with stride > 1 achieve a similar effect at the cost of a higher number of parameters

Global pooling is an alternative to flattening activation maps

- We can create feature vectors (aka embeddings) using the global mean and/or max pooling operations

SPONSORED BY THE

Federal Ministry
of Education
and Research

Further reading and references

A guide to convolution arithmetic for deep learning

- https://arxiv.org/abs/1603.07285

Network in network (1x1 convolutions)

- https://arxiv.org/abs/1312.4400

Hypercolumns for object segmentation and fine-grained localization

- https://openaccess.thecvf.com/content_cvpr_2015/papers/Hariharan_Hypercolumns_for_Object_2015_CVPR_paper.pdf

SPONSORED BY THE
Federal Ministry
of Education
and Research