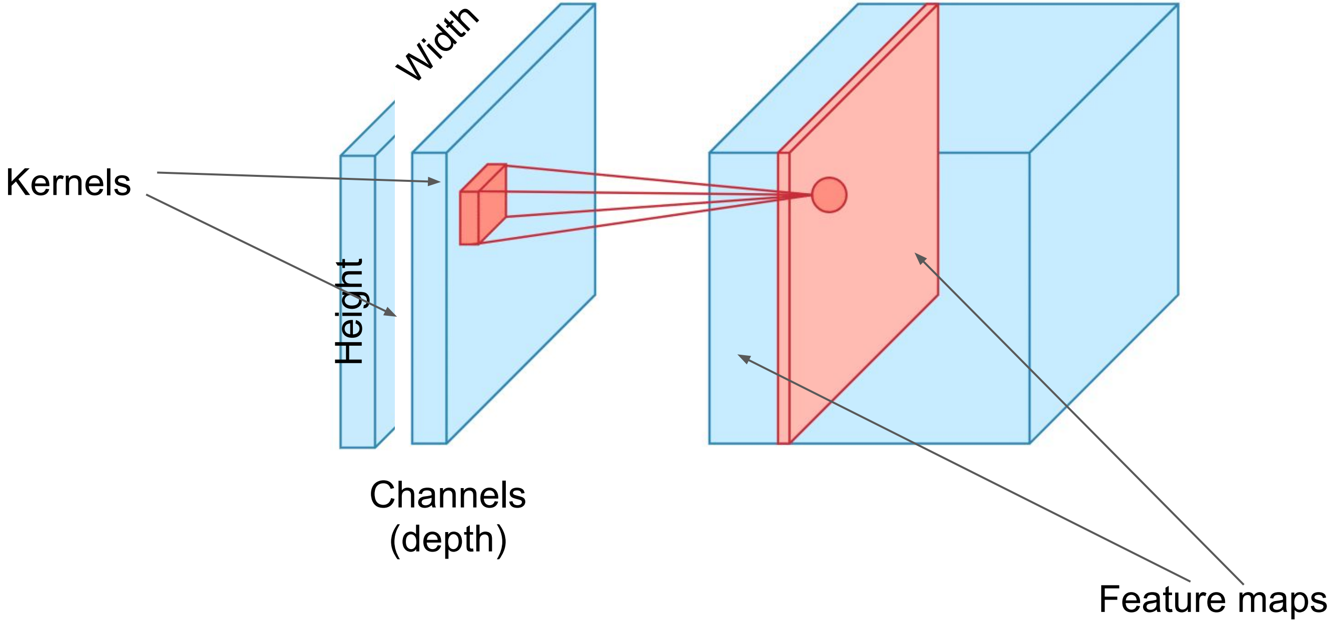
# CNNs

# Definition of CNNs

- One or more layers in the net use convolutions

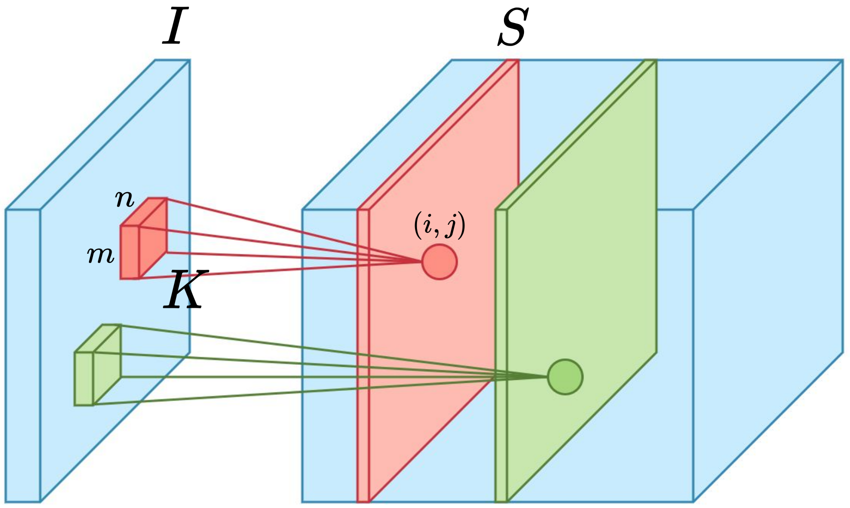



https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

# Convolutions

- Confusing terminology
  - Convnets actually perform *cross-correlation*
- Computing a single output pixel:

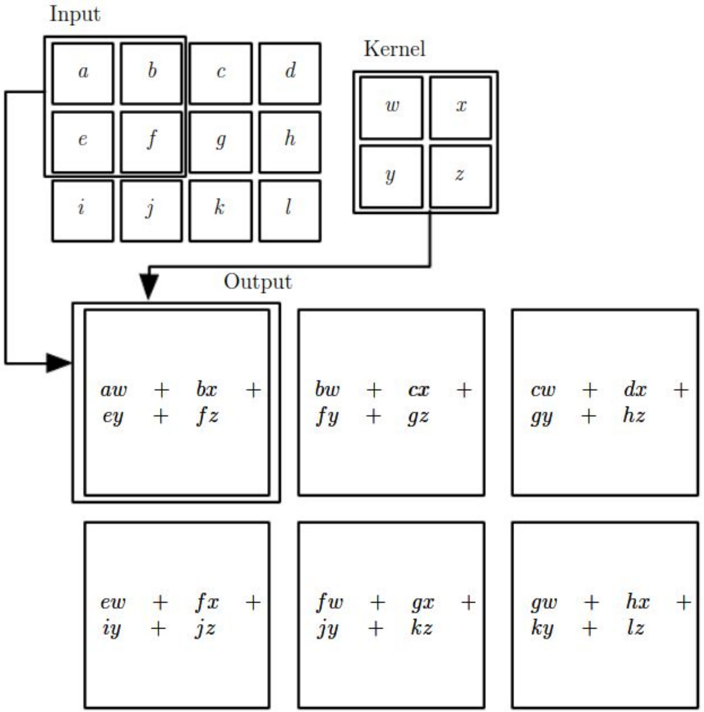
$$S(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$



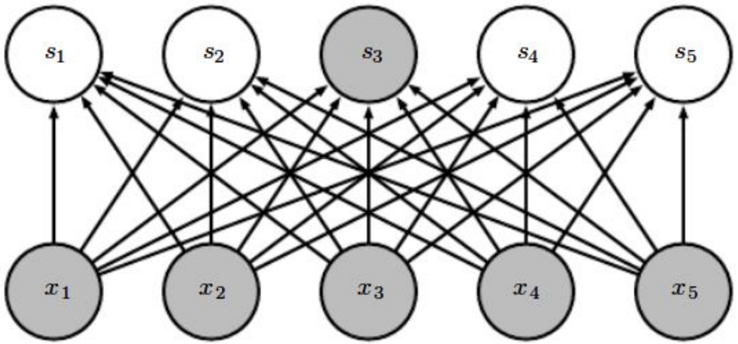

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

# Convolutions

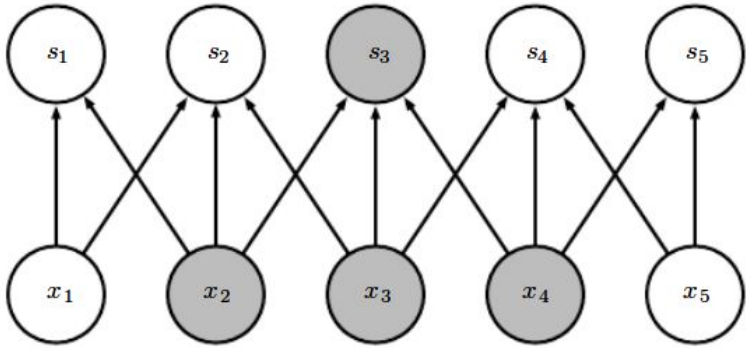
$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$



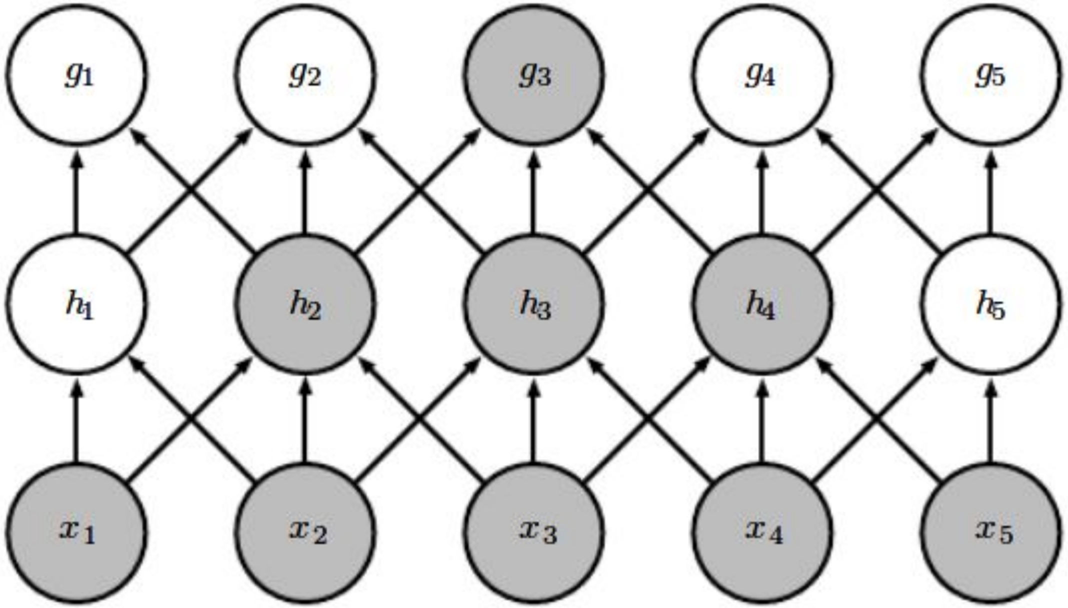
Full connectivity




Sparse connectivity

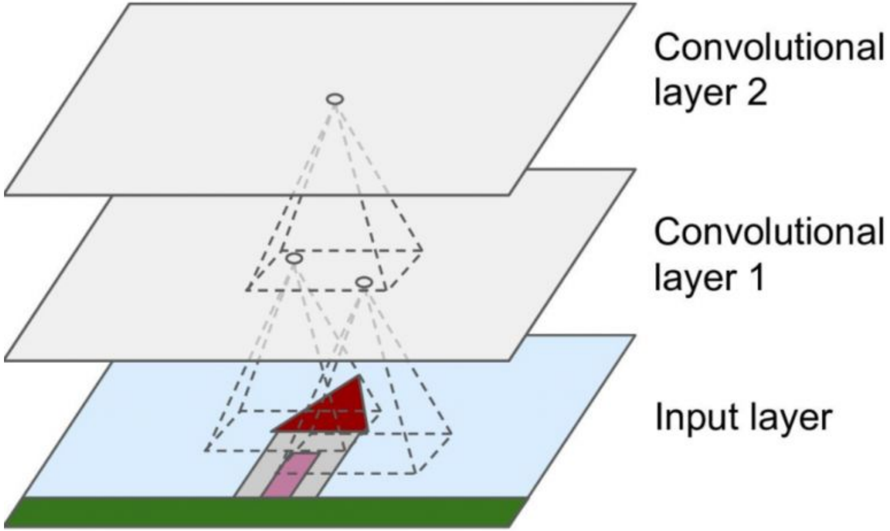


# Receptive field sizes



# Receptive field sizes

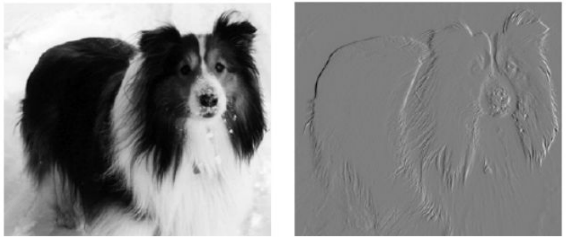
- As we go through higher layers, individual neurons “see” more of the input
- Add to this the increasing number of non-linearities
- This way, higher layers learn more abstract concepts

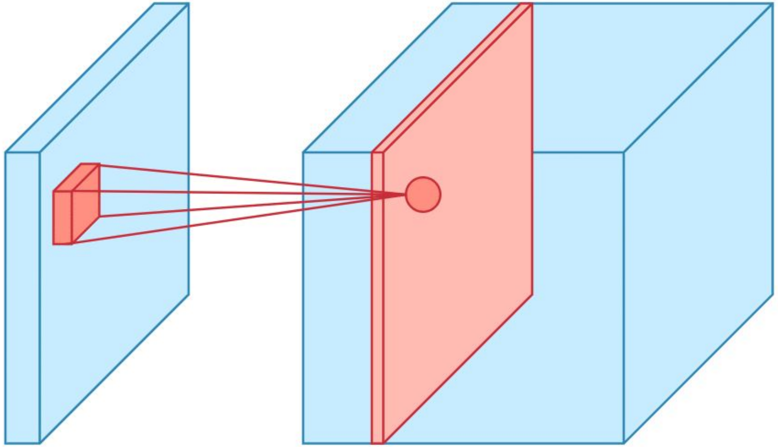


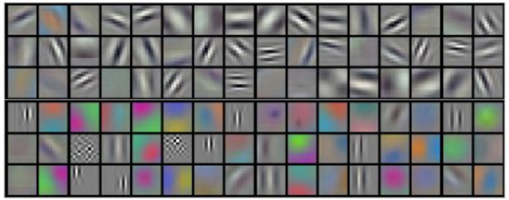

https://tekworld.org/2018/12/25/day-45-100-days-mlcode-convolutional-neural-networks-cnn/#page-content

# Weight matrices as filters

- Early layers in a CNN learn to extract primitive features
- 96 of the learned AlexNet filters (11x11x3) are shown here



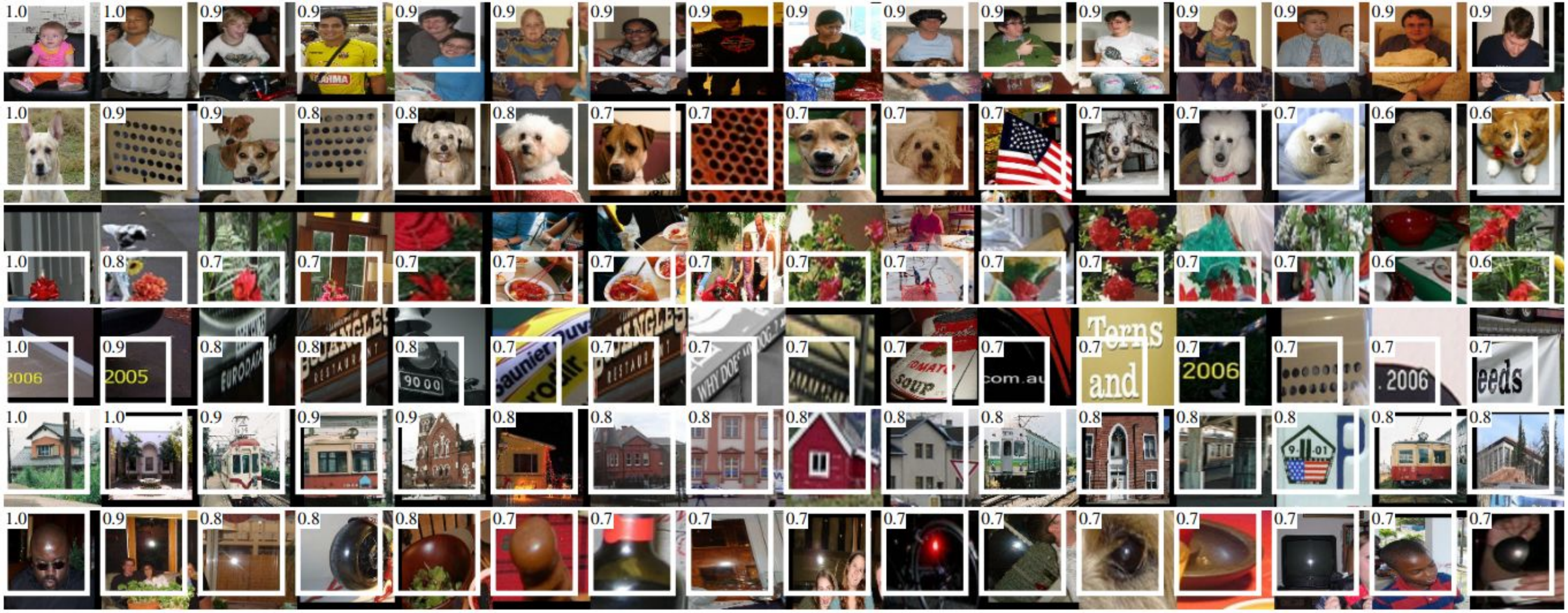




Krizhevsky et al., 2012

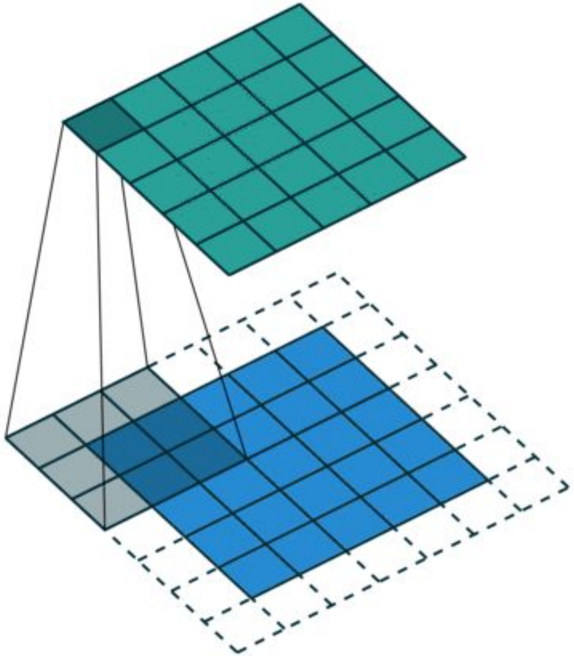
# Higher-level abstractions

- Girshick et al., 2014



# Padding

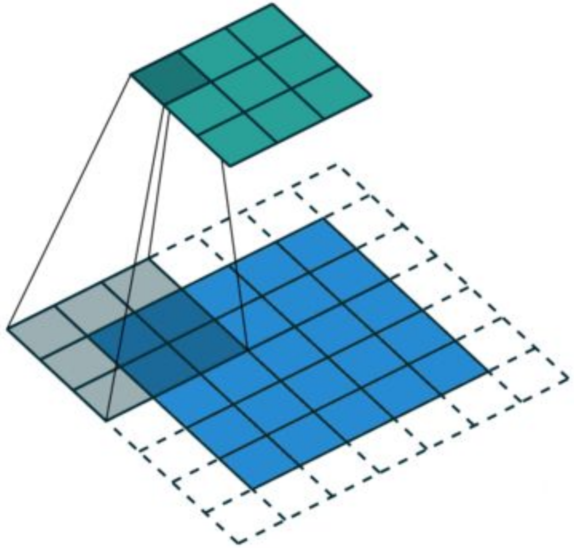
- **Input**
- **Filter**
- **Output feature map**
- Helps maintaining dimensions in the output feature map
- Kernel size:
  - 3
- Padding:
  - 1
- Stride:
  - 1



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

# Strided convolutions

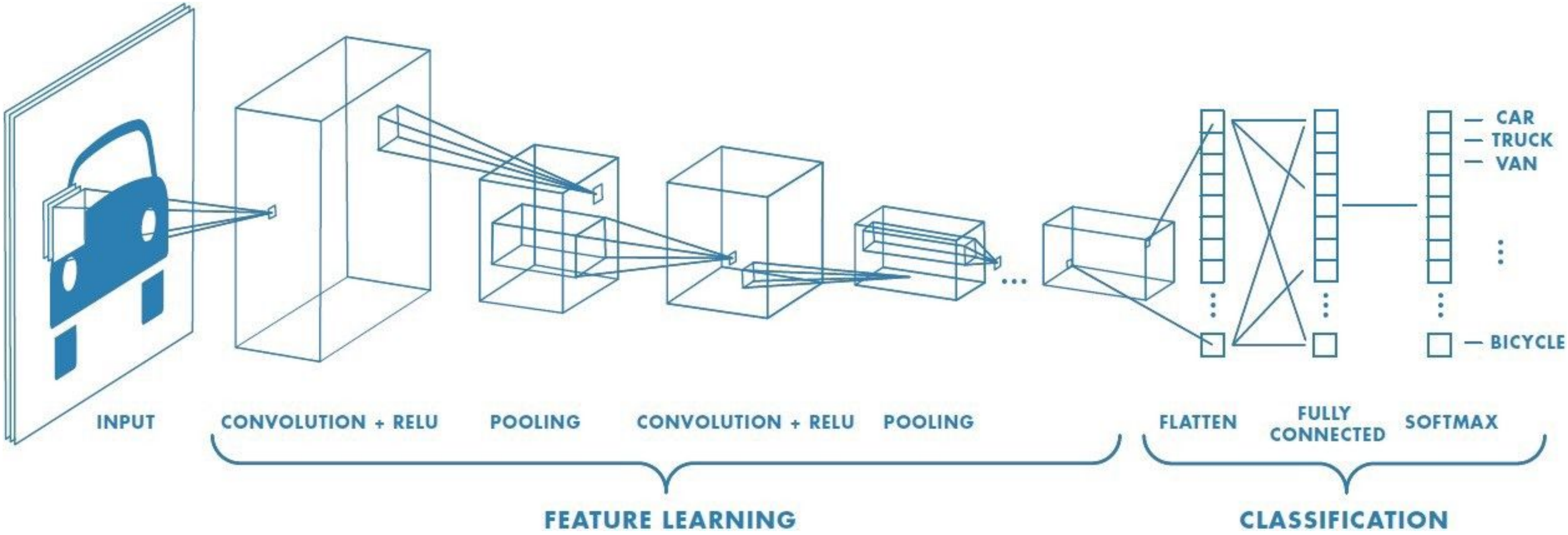
- **Input**
- **Filter**
- **Output feature map**
- Reduces data
- Maintains local invariance
- Kernel size:
  - 3
- Padding:
  - 1
- Stride:
  - 2



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

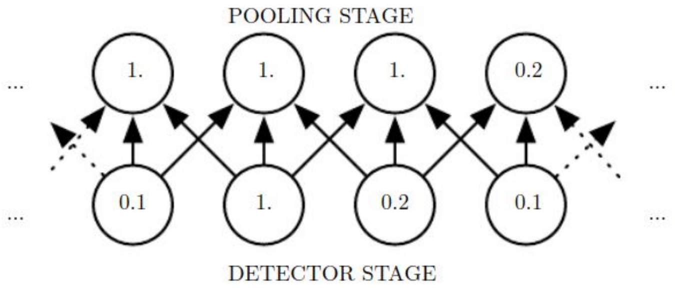
# Pooling

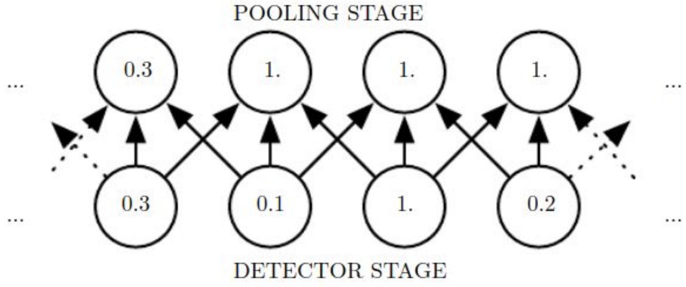
- Data reduction
- Local invariance




https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a5

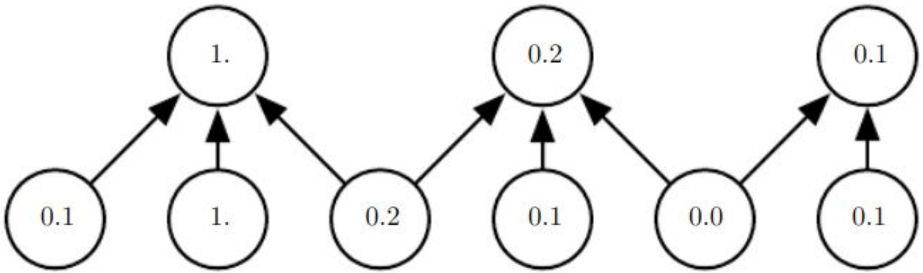
# Pooling (1D)


POOLING STAGE
1.
1.
1.
0.2
0.1
1.
0.2
0.1
DETECTOR STAGE


POOLING STAGE
0.3
1.
1.
1.
0.3
0.1
1.
0.2
DETECTOR STAGE

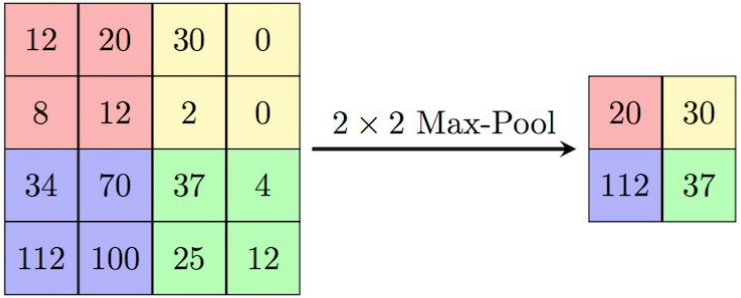
# Strided pooling

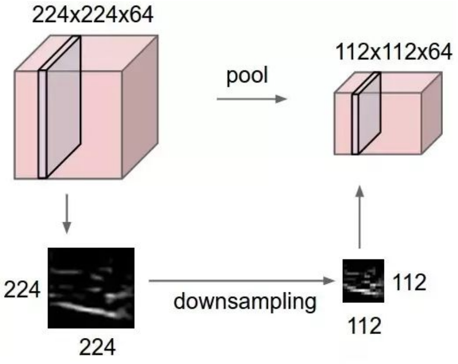
- Similar to strided convolutions, but no parameters
- The stride (here 2) controls how much to downsample
- The pool size controls the “locality”



# Pooling in 2D

- Here we use a stride of 2 and a pool size of 2x2

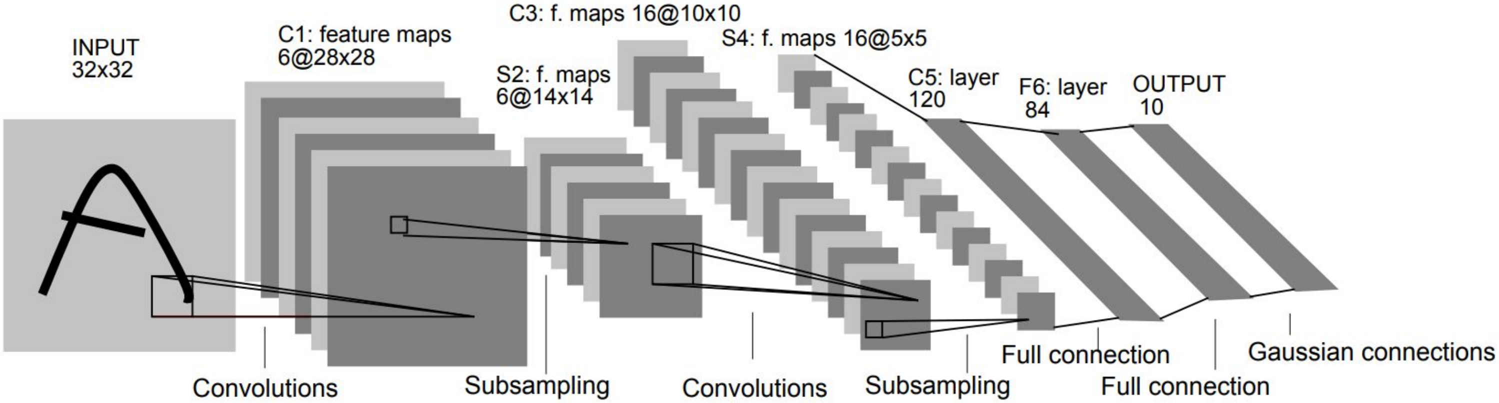


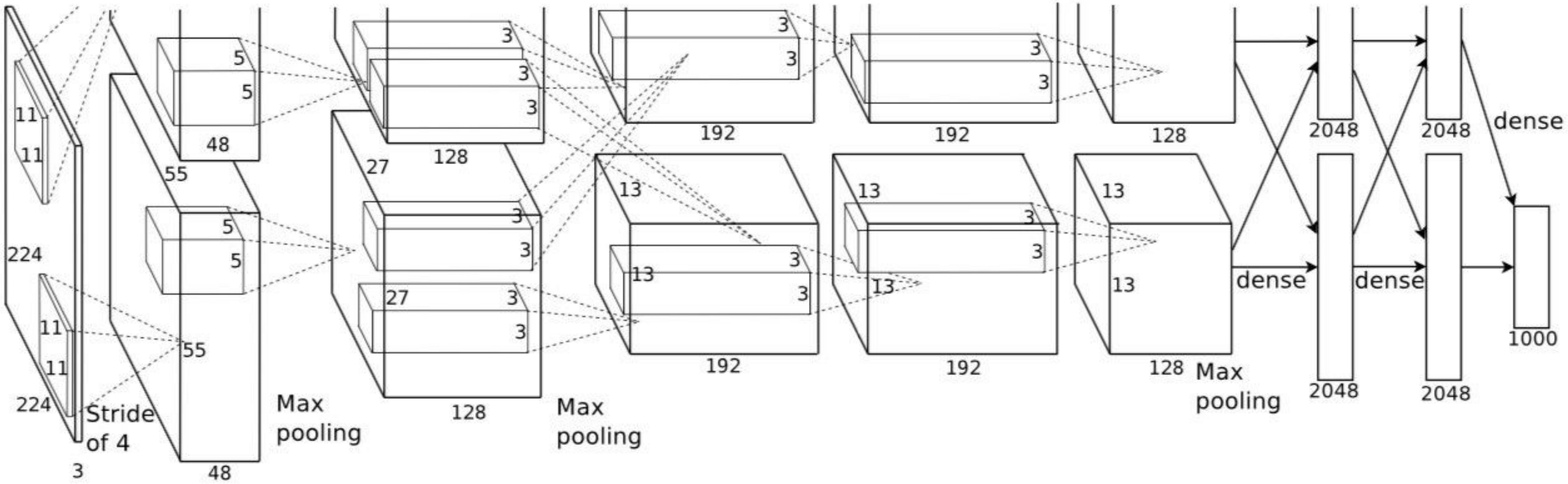

https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

# Some famous CNNs

# LeNet


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

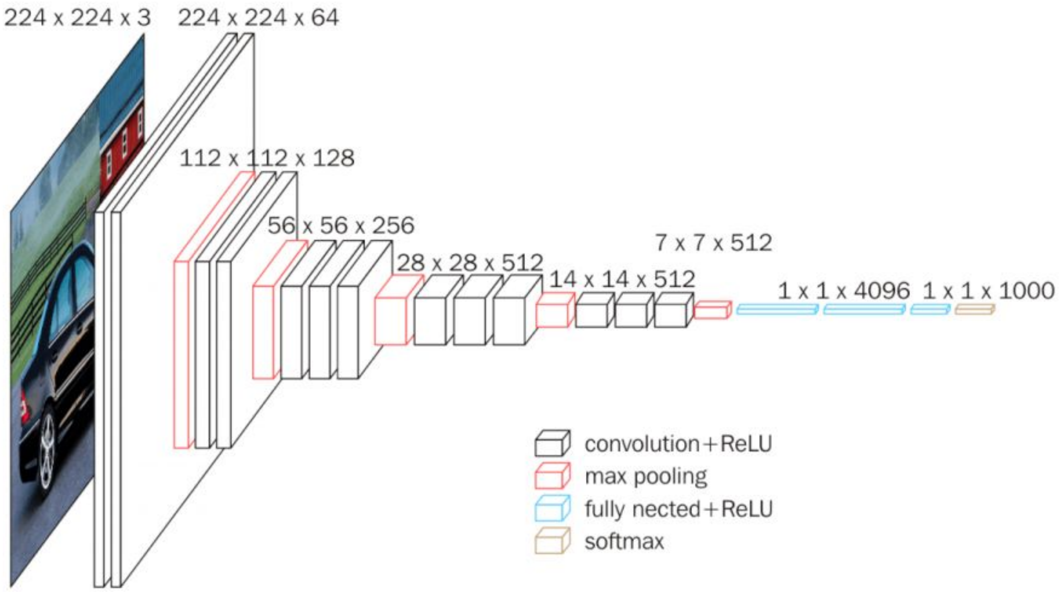
# AlexNet



224
224
3
11
11
11
11
Stride
of 4
55
55
48
48
5
5
5
5
Max
pooling
27
27
128
128
3
3
3
3
3
3
3
3
Max
pooling
13
13
13
192
192
3
3
3
3
13
13
13
192
192
3
3
3
3
13
13
128
128
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000

# VGGNet

- Simonyan & Zisserman, 2015




| 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 | | | | | |

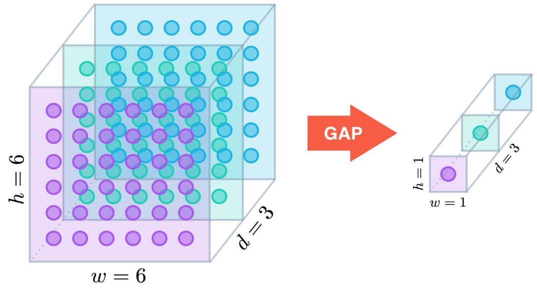
# The all convolutional net

- Springenberg et al., 2015
- Replace all pooling operations with stride-2 convolutions
- Use *global average pooling* in the output layer

**ImageNet model**

| Layer name | Layer description |
|---|---|
| input | Input $224 \times 224$ RGB image |
| conv1 | $11 \times 11$ conv. 96 ReLU units, stride 4 |
| conv2 | $1 \times 1$ conv. 96 ReLU, stride 1 |
| conv3 | $3 \times 3$ conv. 96 ReLU, stride 2 |
| conv4 | $5 \times 5$ conv. 256 ReLU, stride 1 |
| conv5 | $1 \times 1$ conv. 256 ReLU, stride 1 |
| conv6 | $3 \times 3$ conv. 256 ReLU, stride 2 |
| conv7 | $3 \times 3$ conv. 384 ReLU, stride 1 |
| conv8 | $1 \times 1$ conv. 384 ReLU, stride 1 |
| conv9 | $3 \times 3$ conv. 384 ReLU, stride 2, dropout 50 % |
| conv10 | $3 \times 3$ conv. 1024 ReLU, stride 1 |
| conv11 | $1 \times 1$ conv. 1024 ReLU, stride 1 |
| conv12 | $1 \times 1$ conv. 1000 ReLU, stride 1 |
| global_pool | global average pooling ($6 \times 6$) |
| softmax | 1000-way softmax |

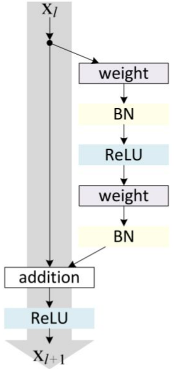
Global Average Pooling

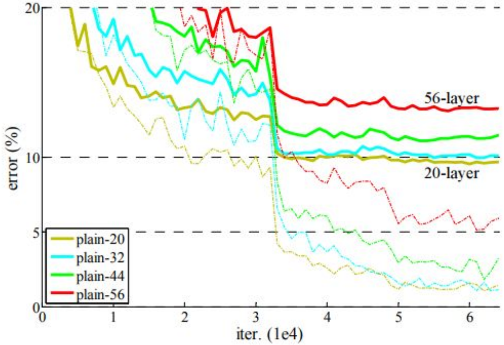


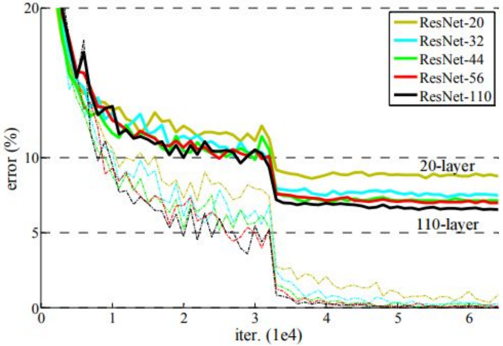

https://towardsdatascience.com/iclr-2015-striving-for-simplicity-the-all-convolutional-net-with-interactive-code-manual-b4976e206760

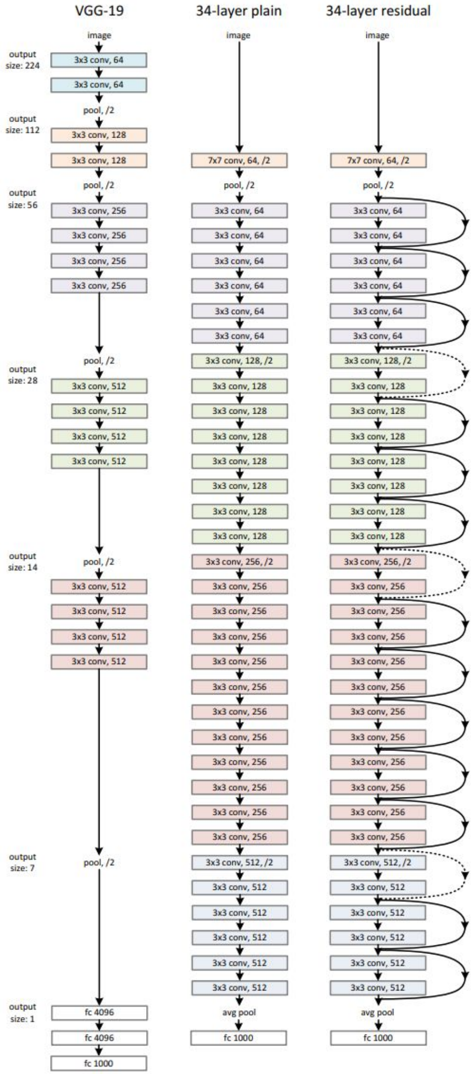
# ResNet

- He. et al., 2015
- Residual “shortcuts” allow for gradient flow
- 100-layer barrier overcome for the first time

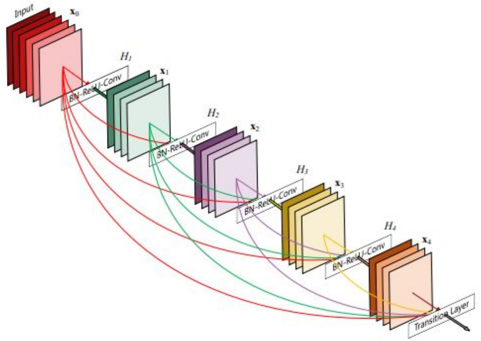


# Densely connected CNNs




- Huang et al., 2016
- In dense blocks, all feature maps are concatenated
- Downsampling only occurs between blocks

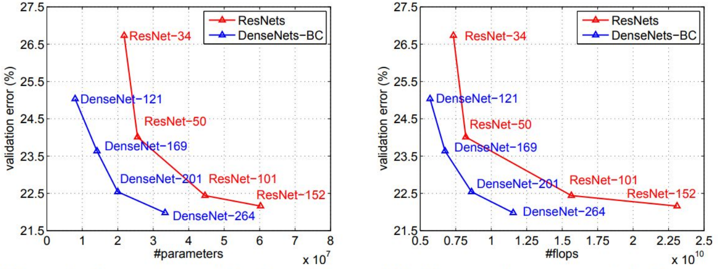
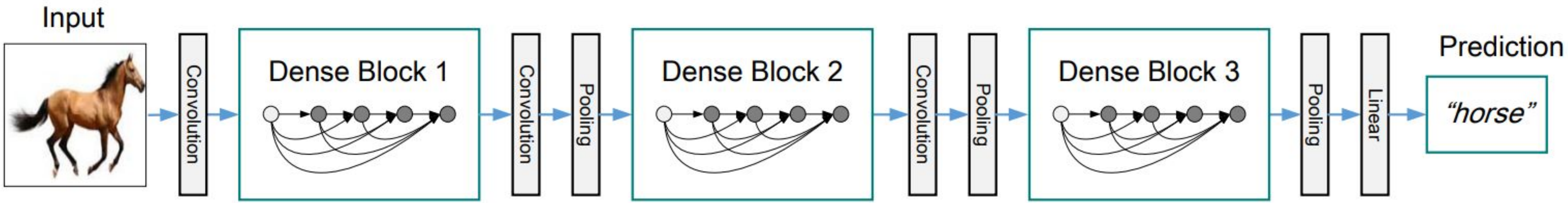
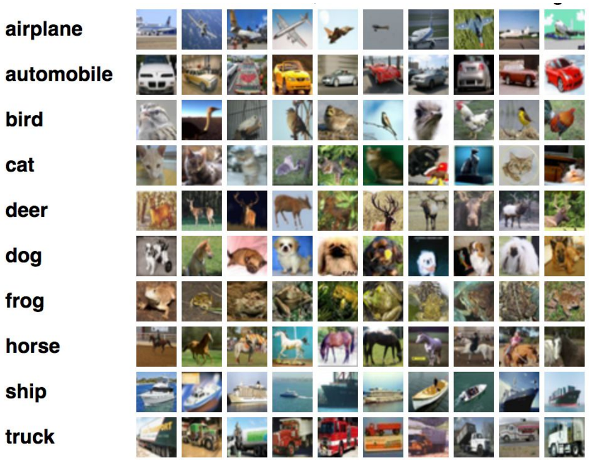



**Figure 3:** Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).



# Challenge - CIFAR10

Previously: KNN (30-40 %), MLP (50-60 %)




- Design and train a CNN on CIFAR10 (> 80 %)
  - Increase the channels while decreasing the resolution through pooling or strided convolutions
  - Potential help: ~ 4 conv layers, use Adam, batchnorm, data augmentation, batchsize: 128, num_workers: 4
- Bonus: Transfer Learning (> 90 %)
  - Load a pretrained resnet 18 model and replace the head with a new head to classify CIFAR10 images
  - Potential help:
    - resnet18.fc = [your new head]
    - Upscale the CIFAR10 images to the image size it was pretrained on (224 x 224)
    - Lower the learning rate to avoid destroying the pretrained weights

# CNN topologies (classification)

- Notice how many CNNs follow a similar pattern