TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

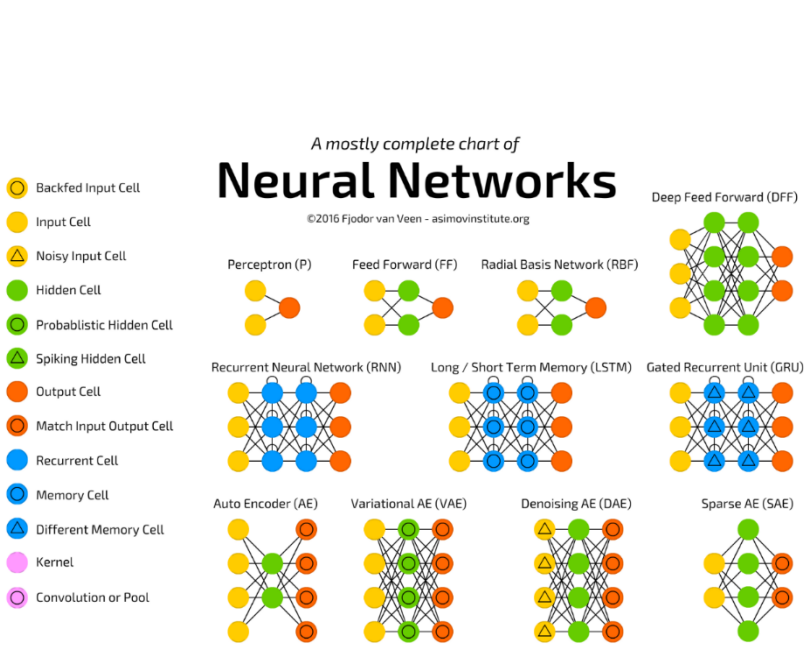
Neural Networks

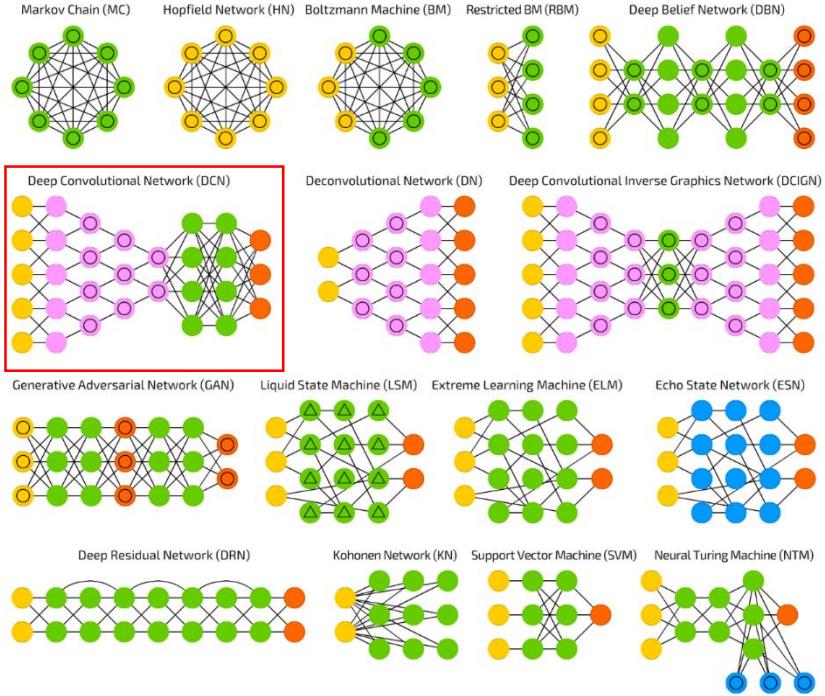
Programming for Data Science

Neural Networks

Dresden Database Systems Group

Architectures



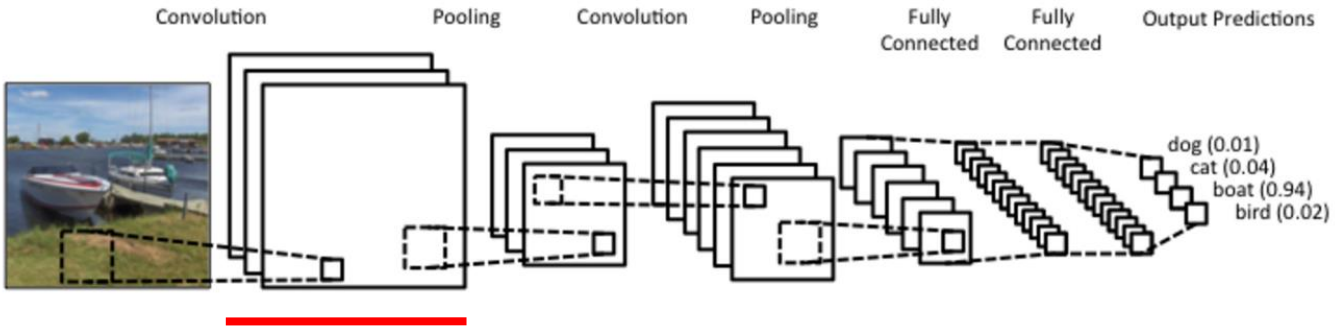


Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks (CNN)

- Processing of Image & Audio data
- Convolutional and pooling layers
- Biological paradigm: receptive field → all sensory receptors that transmit to one subsequent neuron

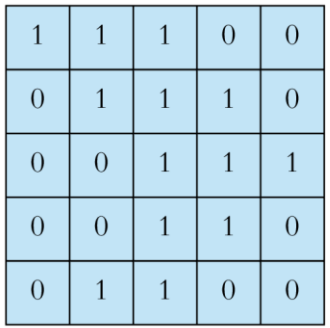


Neural Networks

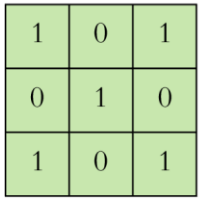
Dresden Database Systems Group

Convolutional Neural Networks

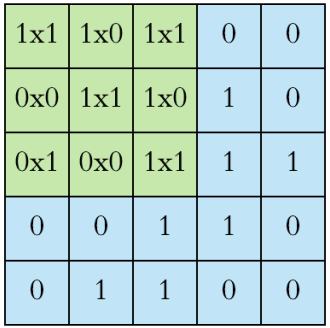
- Convolutional Layer: “small“ convolution matrix / kernel is moved over the input
- Input to the neuron is scalar product of kernel → neighboring neurons have overlapping input
- Most common activation function: ReLu



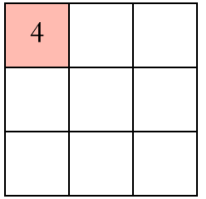
Input



Filter / Kernel



-



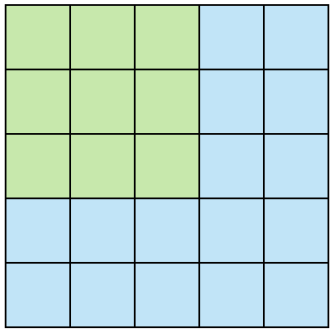
-

Neural Networks

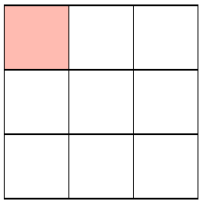
Dresden Database Systems Group

Convolutional Neural Networks

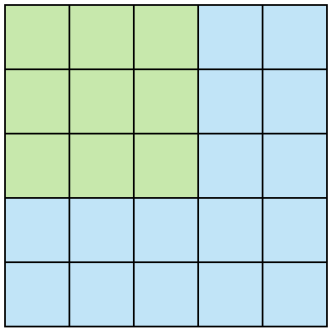
- Parameter: window size, stride, number
- Window size: typically 3x3 (5x5, 7x7, 1x1,...)
- Stride: step size for moving the kernel (default 1)
- Stride influences overlap and number of created convolutions
- Number of filters / kernels : typically power of two



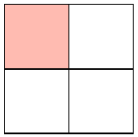
Stride 1



Feature Map



Stride 2



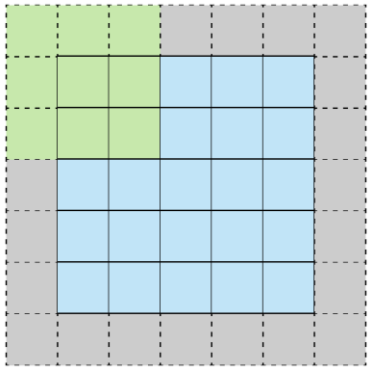
Feature Map

Neural Networks

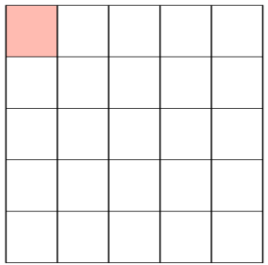
Dresden Database Systems Group

Convolutional Neural Networks

- Parameter: padding
- Kernel has to fit input → reduces dimensionality of feature map
- Padding “frames“ input → feature map retains dimensionality of input
- Padding with 0 or edge values of input



Stride 1 with Padding

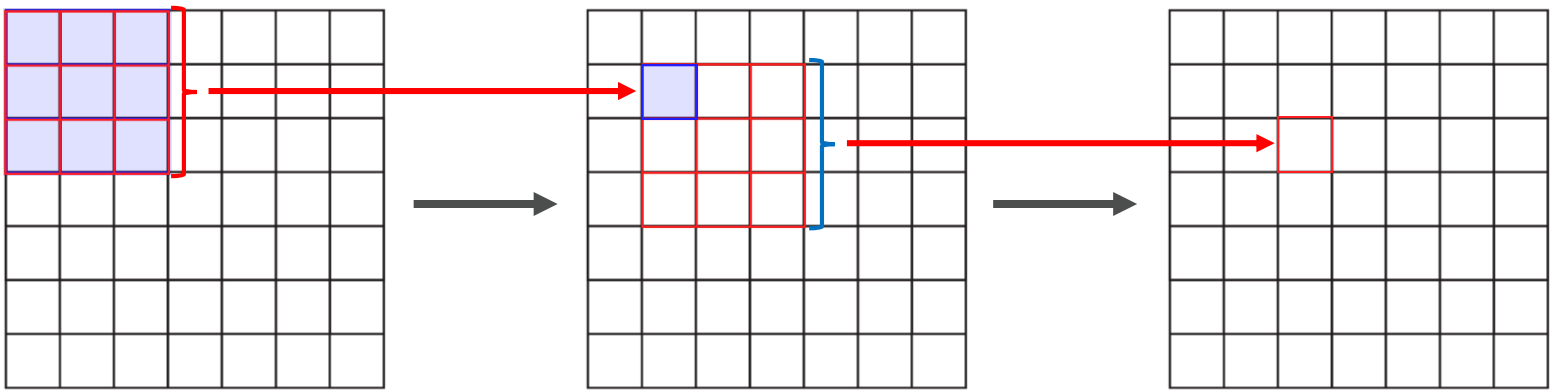


Feature Map

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

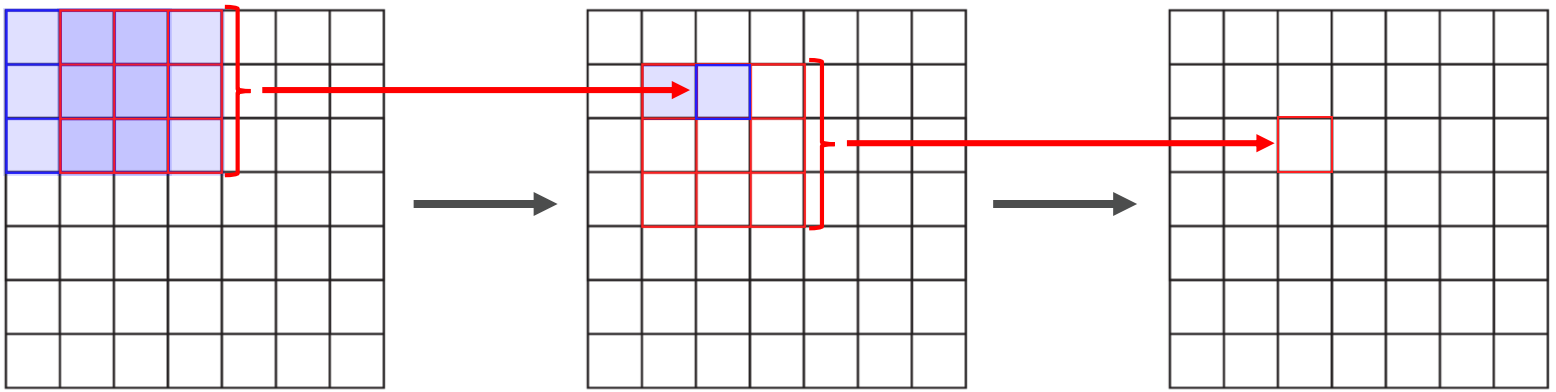
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

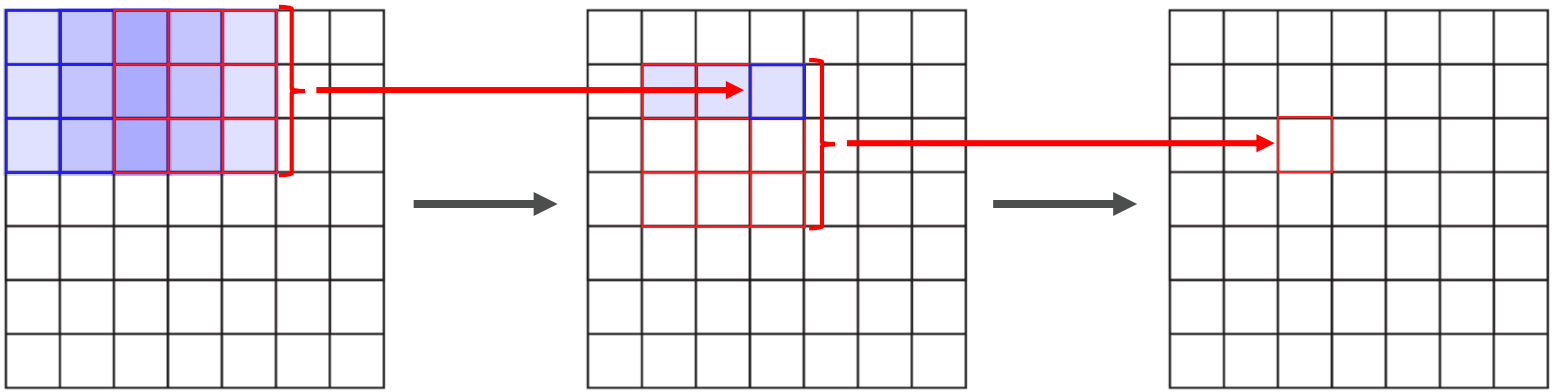
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

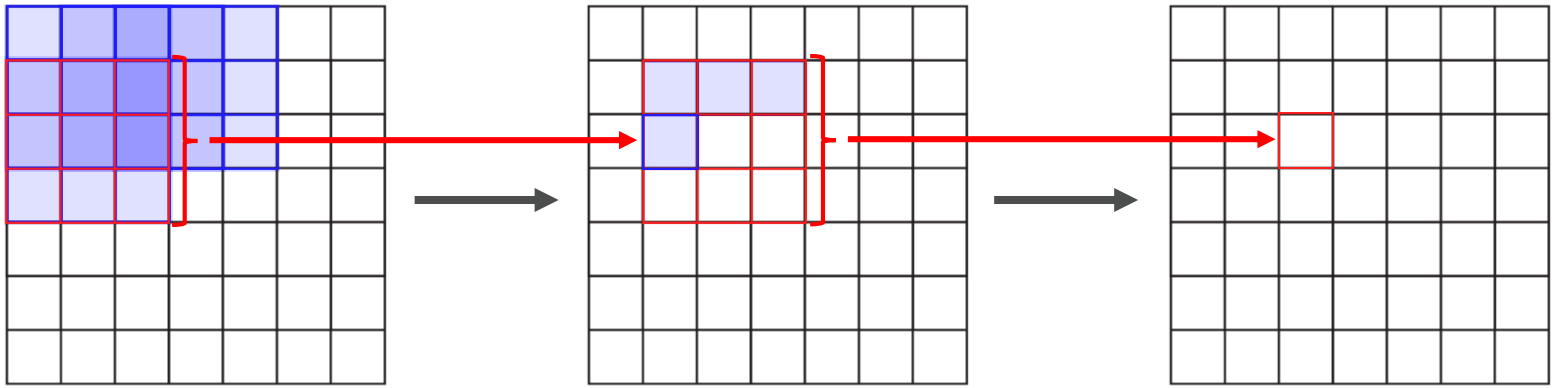
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

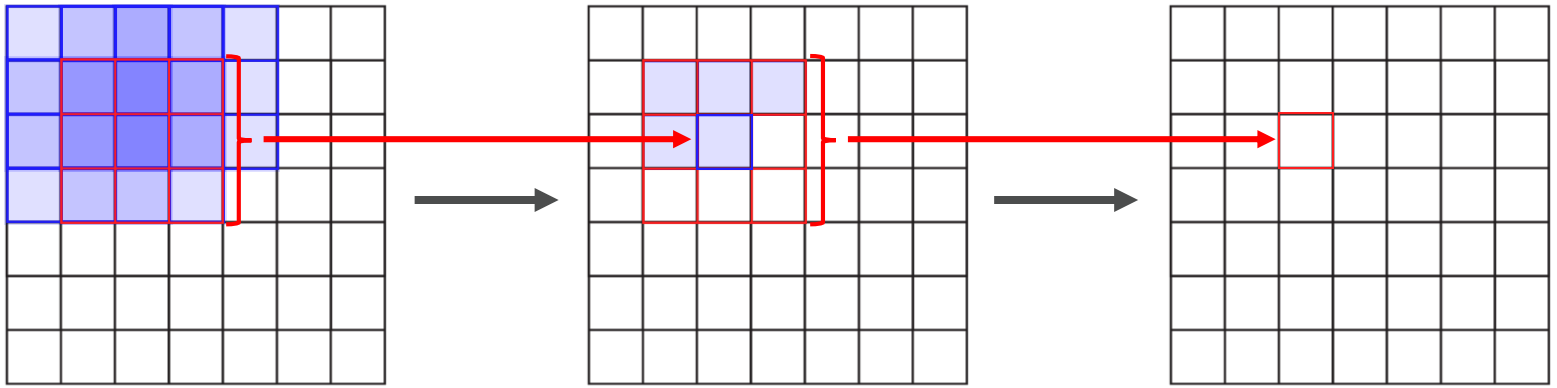
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

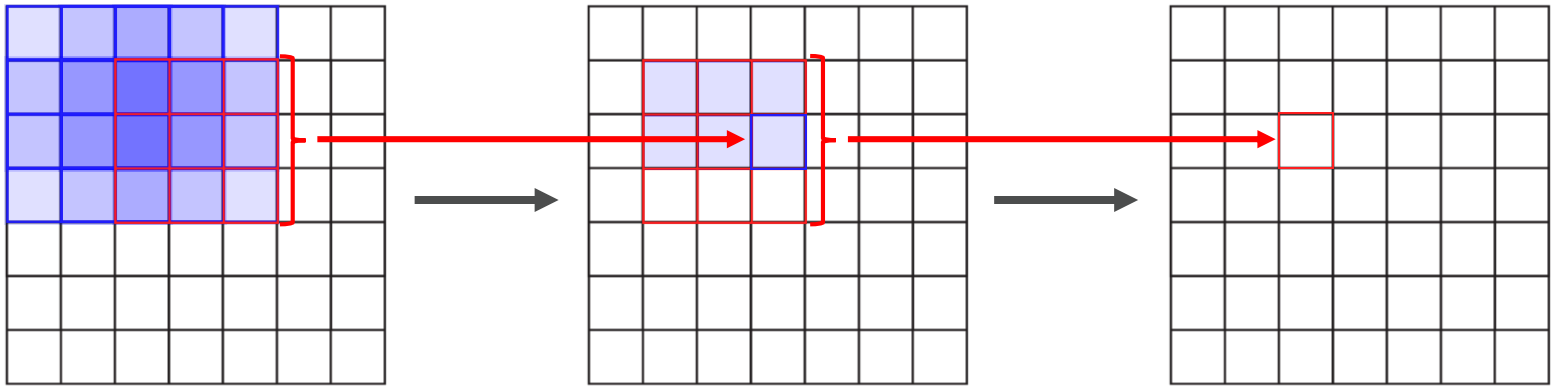
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

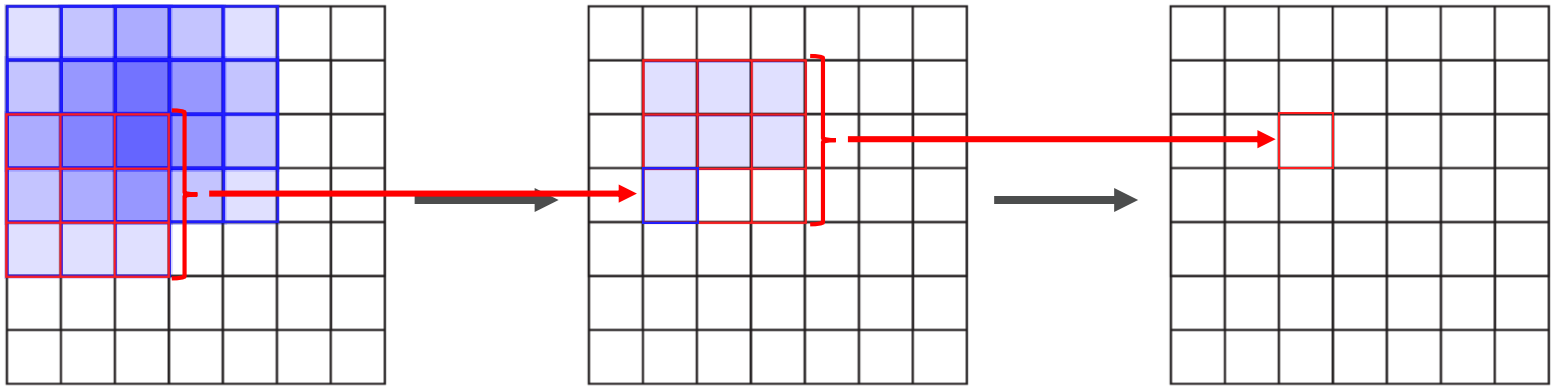
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

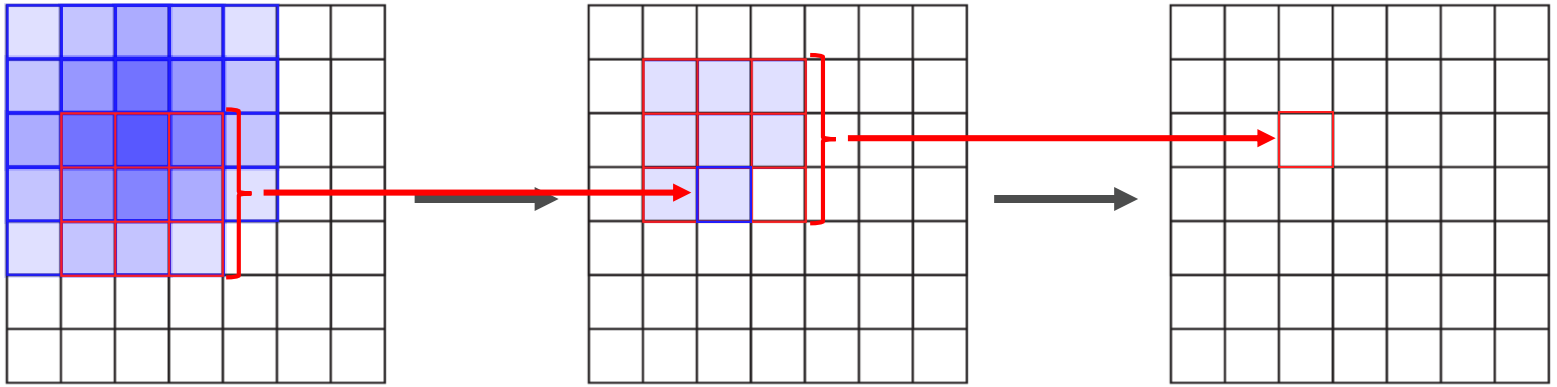
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

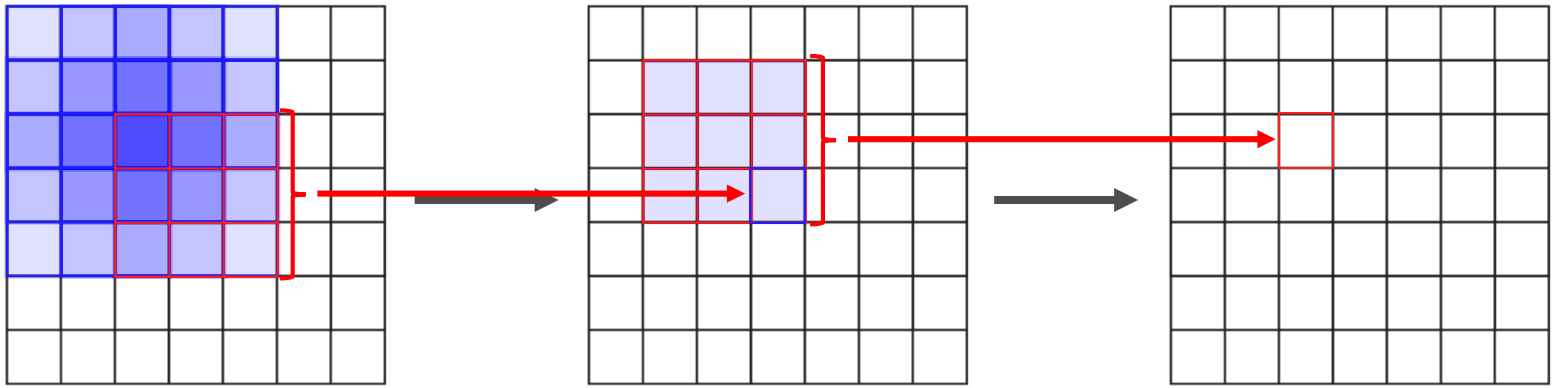
Layer 2

Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks



Layer 1

(Input)

Layer 2

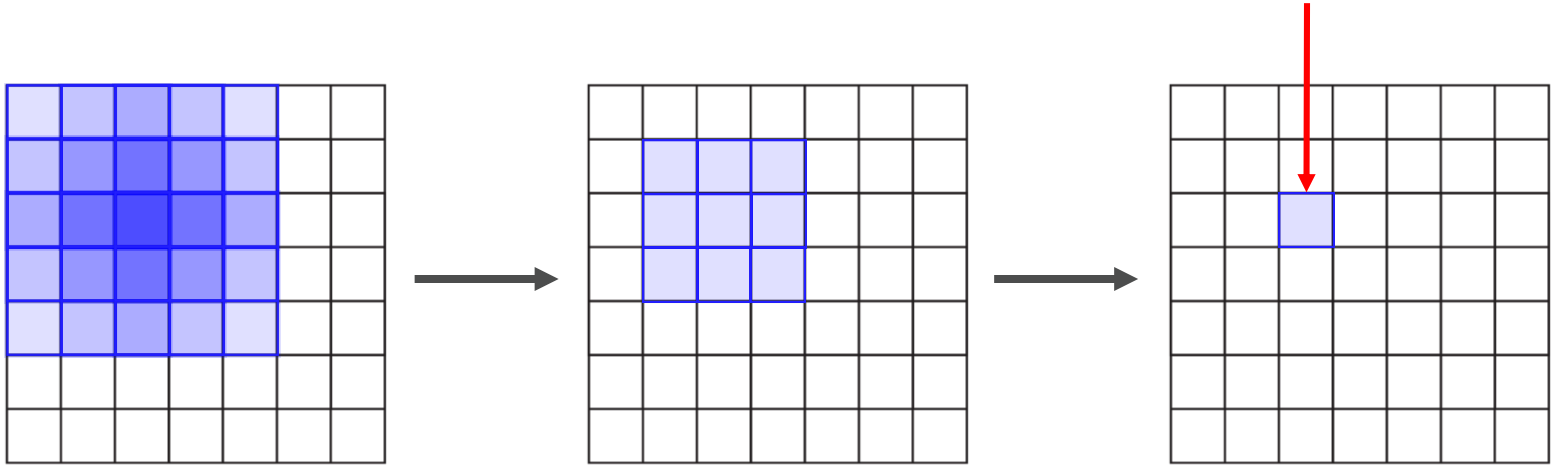
Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks

- Each neuron in layer 3 “sees“ 5x5 neurons from layer 1



Layer 1

(Input)

Layer 2

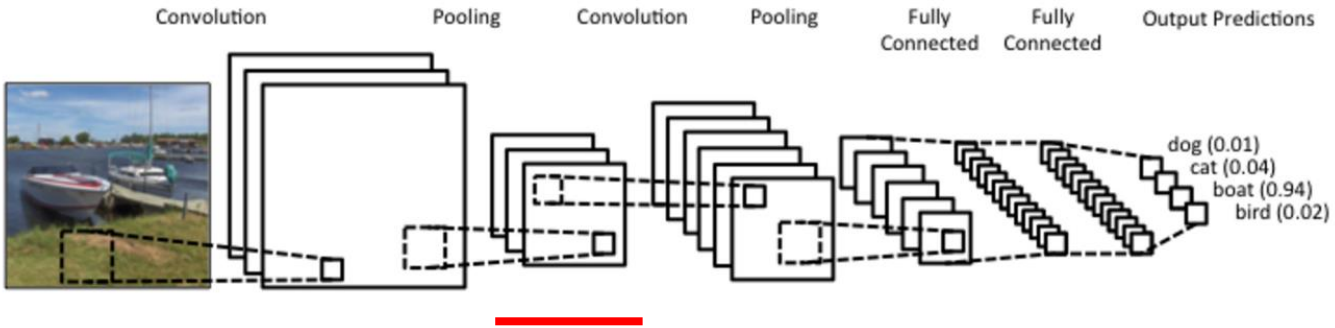
Layer 3

Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks

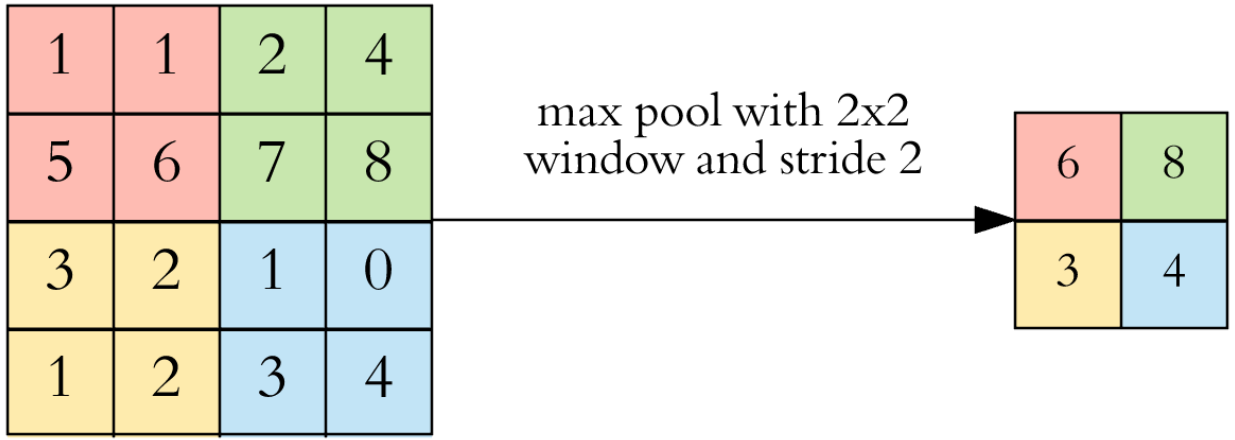
- Subsequent Pooling Layer combine outputs and remove redundant information
- Example: exact position of an edge < approximate position
- Reduces storage requirements, prevents overfitting
- Biological paradigm: lateral inhibition → excited neuron inhibits surroundings → spread of activation is suppressed → creates contrast and better perception



Dresden Database Systems Group

Pooling - Convolutional Neural Networks

- Most common type of pooling layer: max
- 2x2 window with stride 2
- Reduction by 75%

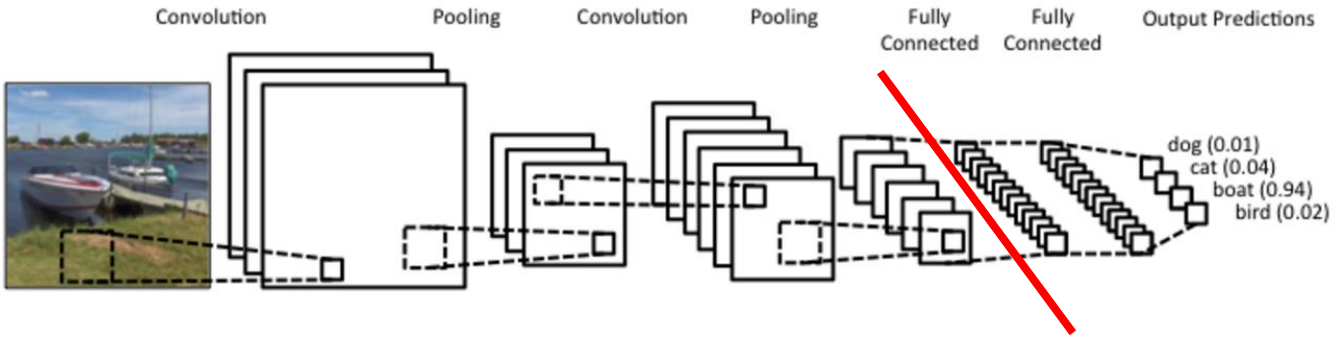


Neural Networks

Dresden Database Systems Group

Convolutional Neural Networks

- After convolution and pooling layers, “standard“ NN layers are added
- Convolution & pooling extract and filter features
- Later feedforward neuron layers use these features for classification
- Flattening: for NN layers, the 2d feature map must be reduced to 1d vector
- Transfer Learning: keep trained convolution and pooling layers, switch NN layers



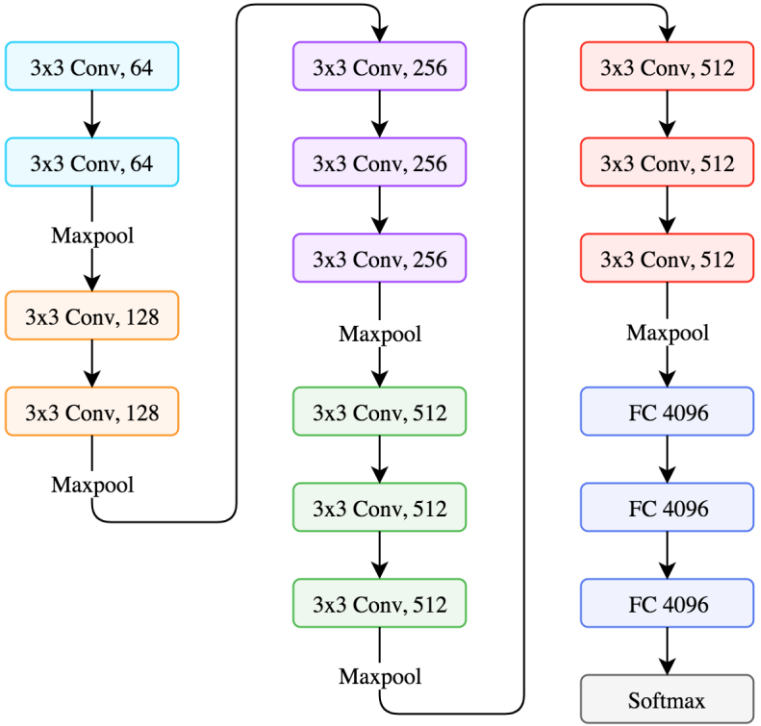
Neural Networks

Dresden Database Systems Group

VGG16 – State-of-the-art CNN

- Oxford's Visual Geometry Group, 2014
- ImageNet classification challenge (2nd place , 7.3% error rate)
- ImageNet complements nouns of WordNet hierarchy (118K) with images (ca. 500 images per noun)
- Stacked convolutional layers:
  - Collection: 2(3x3) = 5x5, 3(3x3) = 7x7
  - Multiple ReLU→ more non linearity
- Training time: 3 weeks on 4 GPUs

- 16 layer → simple and compact!
- Winner 2015: Microsoft ResNet (3.6% error rate) → 152 layer !!



Neural Networks

Dresden Database Systems Group

Conclusion

- Simple basic concept
- Arbitrarily complex application
- Versatile and powerful

- BUT

- Complex / costly
- Opaque
- Hype

Task

Step 0

- Get the MNIST dataset from keras.
- Use the predefined split into training and test data.
- Reshape and scale your data.

Step 1

- Use a CNN* with two convolutional layers (32 and 64 filters, kernel size 3x3, same padding), one max pooling layer (pool size 2x2) and at least one dense layer to classify all digits from their images.
- Other parameters: RELU activations, categorical cross entropy, accuracy metric, the Adam optimizer, 12 epochs and a batch size of 32.
- Evaluate the NN on the test data.

[Step 2]

- Find a better structure for your CNN.

*use your own implementation

Package suggestions

Dresden Database Systems Group

R

- (data.table)
- keras

python3

- numpy
- pandas
- keras

Exercise Appointment

Dresden Database Systems Group

We compare and discuss the results

- Tuesday, 21.01.2020,
- Consultation: 16.01.2020,
- Please prepare your solutions! Send us your code!

If you have questions, please mail us:

claudio.hartmann@tu-dresden.de Orga + Code + R
lucas.woltmann@tu-dresden.de Tasks + Python