Lesson 08

Convolutional Neural Network

Ing. Marek Hrúz, Ph.D.

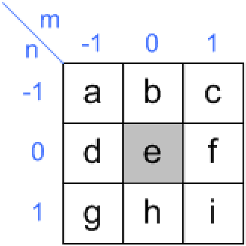
Katedra Kybernetiky
Fakulta aplikovaných věd
Západočeská univerzita v Plzni

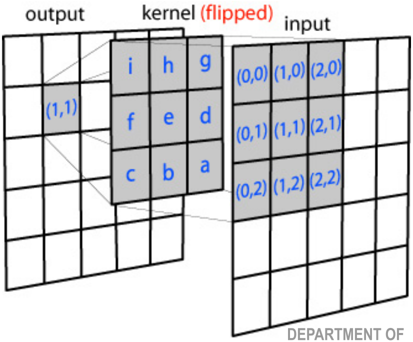
DEPARTMENT OF CYBERNETICS

Convolution

- we will consider 2D convolution
- the result of convolution: for each point it tells us the area under the multiplication of signal and kernel

$$(f * g)[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[m-i, n-j] \cdot g[i, j] \quad (1)$$





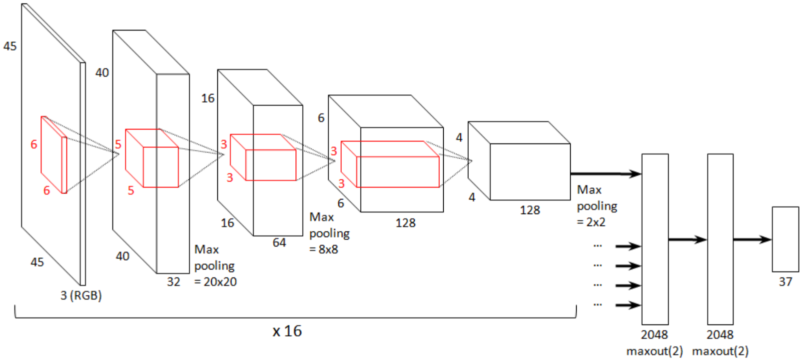
Architecture of Convolutional Neural Network

- There can be more variations of the architecture
- The standard architecture for image classification
- Convolution Layer $\rightarrow$ Max-pooling $\rightarrow$ Fully connected Layer

Convolutional Layer

- A special layer inside a Neural Network
- We define size and number of kernels (filters) to be learned, the stride and padding
- The layer uses the defined kernels to compute 'feature maps' over the input map as a convolution
- This dramatically reduces the number of parameters in the layer as opposed to fully connected layer
- Usually the input is the image $\textit{width} \times \textit{height} \times \textit{channels}$
- The kernels operate through channels $\rightarrow$ kernel of defined size 3×3 really has size $3 \times 3 \times \textit{channels}$
- This holds for all other convolutional layers $\rightarrow$ the number of kernels in a layer defines the number of channels of its output feature map

Output of Convolutional Layer - example

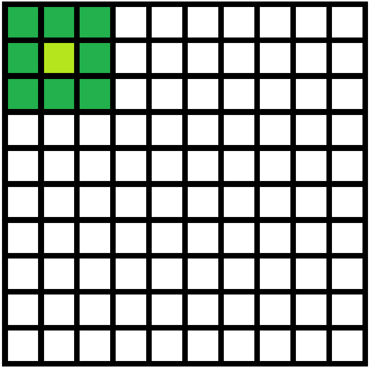


- The first convolutional layer has 32 kernels, thus the output feature map has depth (number of channels) equal to 32
- If we define the size of kernels in the consecutive layer to be 5×5 the kernels will have (really) the size of $5 \times 5 \times 32$

DEPARTMENT OF CYBERNETICS

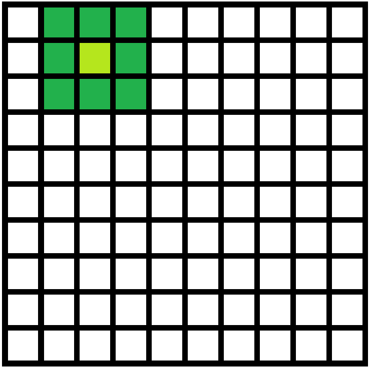
Output of Convolutional Layer - strides

- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride one:



Output of Convolutional Layer - strides

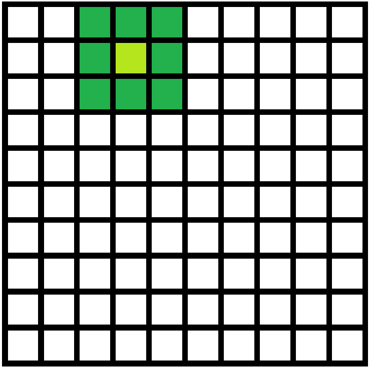
- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride one:



DEPARTMENT OF CYBERNETICS

Output of Convolutional Layer - strides

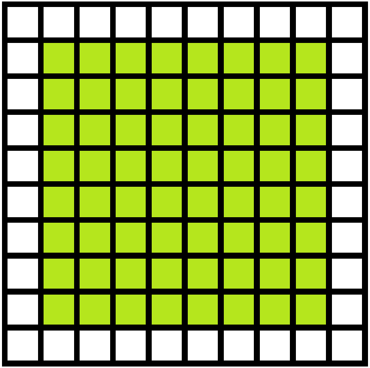
- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride one:



DEPARTMENT OF CYBERNETICS

Output of Convolutional Layer - strides

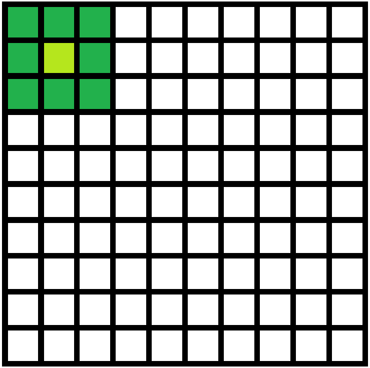
- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride one, visited locations:



DEPARTMENT OF CYBERNETICS

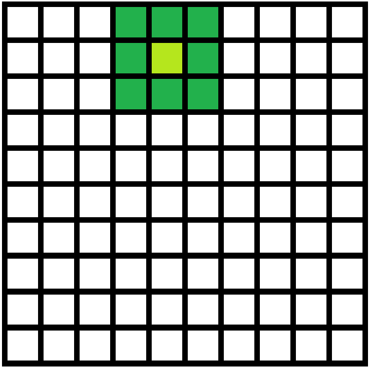
Output of Convolutional Layer - strides

- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride three:



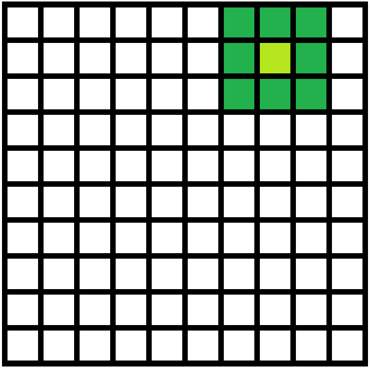
Output of Convolutional Layer - strides

- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride three:



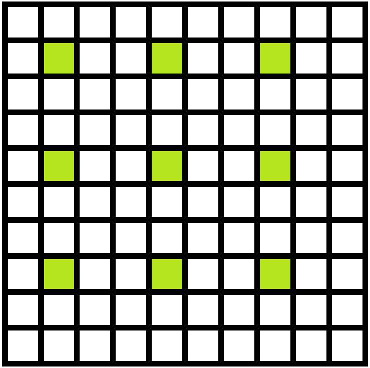
Output of Convolutional Layer - strides

- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride three:



Output of Convolutional Layer - strides

- The stride defines by how many pixels we move the kernel until we apply it next time
- Stride three, visited locations:



Output of Convolutional Layer - strides

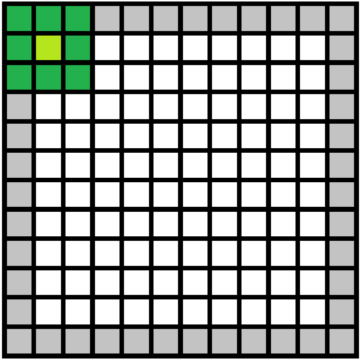
- The stride defines the size of the output feature map
- In the previous example we had an image 10×10
- With stride one, the output map will be of size 8×8
- With stride three, the output map will be of size 3×3
- The stride can be rectangular (eg. 3×1)
- There are several strategies for choosing strides
- Very often the strides are chosen so that consecutive kernels overlap

Output of Convolutional Layer - padding

- Padding is important for managing the shape of the output feature map
- It is a scalar parameter that determines the width of added boundary pixels to the input map
- Current implementations support zero valued boundaries

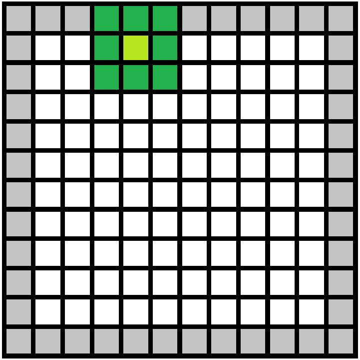
Output of Convolutional Layer - padding

- Example of padding equal to one, stride equal to three:



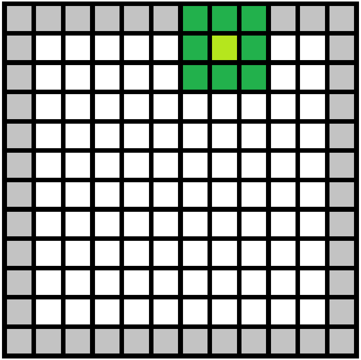
Output of Convolutional Layer - padding

- Example of padding equal to one, stride equal to three:



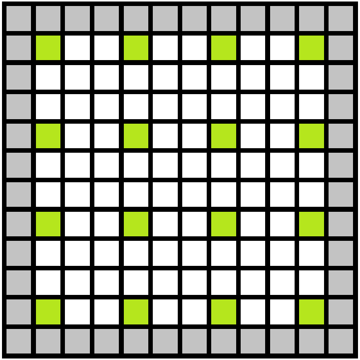
Output of Convolutional Layer - padding

- Example of padding equal to one, stride equal to three:



Output of Convolutional Layer - padding

- Example of padding equal to one, stride equal to three:
- Visited locations



Computing the convolution

- Let's consider M kernels $K_c, c = 1, \ldots M$ with size $k \times k$
- The size of the input map is $W^I \times H^I \times C^I$
- The depth of the output map C^O is the number of kernels M
- The width and height of the output map is determined by the size of the kernels, the strides, and the padding

$$W^O = \frac{W^I + 2 \cdot \mathrm{pad} - k}{\mathrm{stride}} + 1 \tag{2}$$

- For each output location (x, y, c) and each kernel K_c, where $c = 1, \ldots, M$ we compute the convolution:

$$O[x, y, c] = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \sum_{n=0}^{C^I-1} I[x_s - i, y_s - j, n] \cdot K_c[i, j, n] \tag{3}$$

- where (x_s, y_s, n) is the proper location in the input map given the stride and pool

DEPARTMENT OF CYBERNETICS

Activation function

- The output of the convolution is then 'activated' using activation function

$$O_{map}[x, y, c] = f\left(O[x, y, c] + b\right) \qquad (4)$$

- b is the bias term
- The choice of the activation function is arbitrary, up to the point of being differentiable (or at least have a defined derivative) on the whole domain
- The mathematical purpose of the activation function is to model the non-linearity
- But it is not necessary: $f(x) = ax$ is a proper activation function

Activation function - sigmoidal

- A family of S shaped functions
- Commonly used in past to model the activity of a neuron cell
- Sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{5}$$

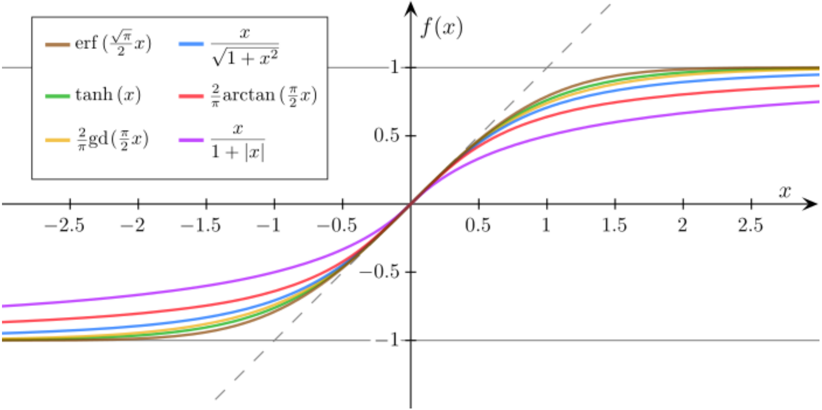
- Hyperbolic tangent:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6}$$

- There are more possibilities ...

DEPARTMENT OF
CYBERNETICS

Activation function - sigmoidal examples

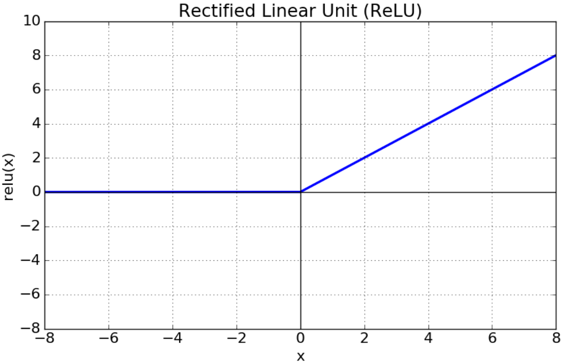


DEPARTMENT OF CYBERNETICS

Activation function - Rectified Linear Unit

- Most commonly used activation function in CNN

$$f(x) = \max(x, 0) \tag{7}$$

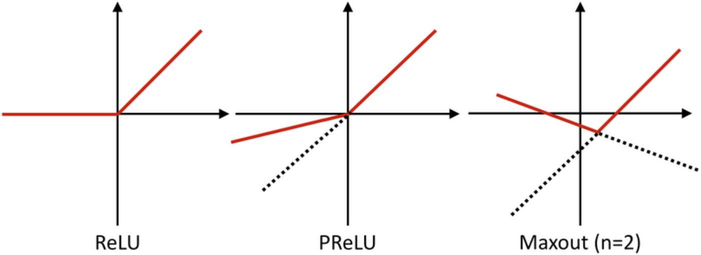


- non-linear, easy gradient

DEPARTMENT OF CYBERNETICS

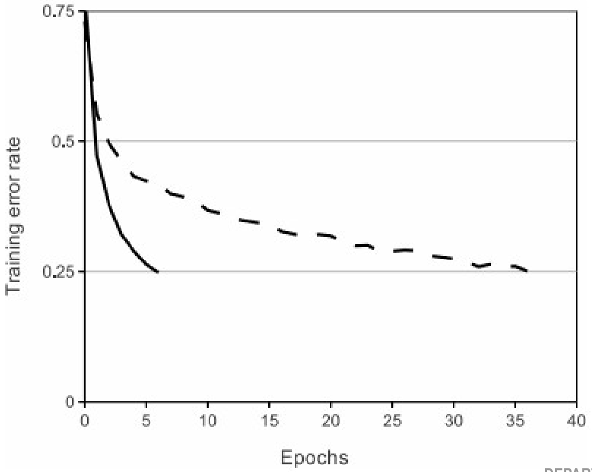
Activation function - Rectified Linear Unit Modifications

- PReLU - parametrized ReLU, where the slope of the negative part is handled as a parameter to be learned via backpropagation
- Maxout - several linear functions are being learned via backpropagation and the activation is the max of these



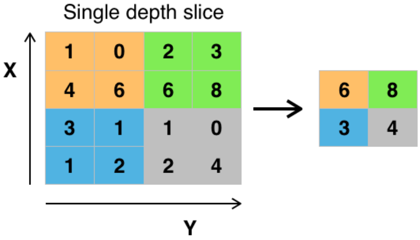
Activation function - Rectified Linear Impact on Training

- Krizhevsky (2012) reports a much faster learning with ReLU as opposed to tanh



Pooling

- Pooling is used to compress the information propagated to the next level of network
- In past average pooling was used
- More recently (2012) the max pooling was re-introduced and experiments show its superiority



- Overlapping pooling seems to be important
- Parameters: size of the pooling window, stride of the pooling window

Batch Normalization

- Any kind of normalization is important
- Batch Normalization is widely used and is the leading form of normalization in the means of performance
- The statistics of the output map are computed
- They are normalized so that they have zero mean and unit variance $\rightarrow$ well-behaved input for the next layer
- The normalization factors - scale and shift - are remembered as a running average through the training phase
- Further more the zero mean and unit variance statistics are scaled and shifted via learned parameters γ, β
- The main idea: decorrelation, any slice of the network has similar inputs/outputs, faster training

Classification layer - Softmax

- The best practice for classification is to use softmax
- Softmax is a function

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{8}$$

- that takes the input vector and transforms it to be between $\langle 0; 1 \rangle$ and to sum up to one
- It is a generalization of logistic function
- If j is the index of a class, then $\sigma(\mathbf{z})_j$ is the probability of the input belonging to class C_j
- The targets are so called 'one hot vectors' - a vector with one on the index j and zeros elsewhere

Learning - Objective function - Classification

- To be able to learn the parameters of the network we need a objective (criterion, loss) function to be optimized
- For the classification task with softmax layer we use so called categorical cross-entropy

$$L(p, q) = -\sum_{x} p(x) \log q(x) \tag{9}$$

- where x is the index of the class, p is the true distribution (one hot vector) and q is the approximated distribution (softmax)

Learning - Objective function - Regression

- Regression is a form of approximation when we provide inputs and outputs and are looking for parameters that minimize the difference between generated outputs (predictions) and provided outputs
- Mean squared error:

$$L(Y, \hat{Y}) = \frac{1}{N} \sum_{i} (y_i - \hat{y}_i)^2 \tag{10}$$

- Mean absolute error:

$$L(Y, \hat{Y}) = \frac{1}{N} \sum_{i} |y_i - \hat{y}_i| \tag{11}$$

- Hinge loss:

$$L(Y, \hat{Y}) = \frac{1}{N} \sum_{i} \max(1 - y_i \cdot \hat{y}_i, 0) \tag{12}$$

DEPARTMENT OF CYBERNETICS

Learning - Stochastic Gradient Descent

- To find the optimal values of the parameters ω of the network (weights and biases), we need to use backpropagation
- That is to compute the partial derivatives of the objective functions against individual parameters
- CNN has much less parameters than fully connected net $\rightarrow$ faster convergence
- The most widespread approach is to use **stochastic gradient descent**

$$\omega^* = \underset{\omega}{\operatorname{argmin}}\, L(\omega) \tag{13}$$

$$\omega^{t+1} = \omega^t - \epsilon \cdot \left\langle \frac{\partial L}{\partial \omega} \middle| \omega^t \right\rangle_{D_t} \tag{14}$$

- where t is the iteration step, ϵ is the learning rate, $\left\langle \frac{\partial L}{\partial \omega} \middle| \omega^t \right\rangle_{D_t}$ is th average over the t-th batch D_t with respect to ω evaluated at ω_t

Learning - Stochastic Gradient Descent

- The SGD uses mini-batches to optimize the parameters
- A mini-batch is an example of training data - not too small, not too big
- One run through a mini-batch is called an iteration, a run over all mini-batches in training dataset is called epoch
- It is very useful to use momentum in the computing of the gradient

$$v^{t+1} = \alpha \cdot v^t - \beta \cdot \epsilon \cdot \omega^t - \epsilon \cdot \left\langle \frac{\partial L}{\partial \omega} \Big| \omega^t \right\rangle_{D_t} \tag{15}$$

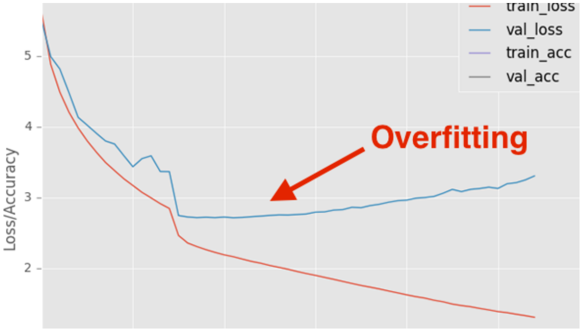
- where α is the momentum (0.9), β is the weight decay (0.0005) and then

$$\omega^{t+1} = \omega^t + v^{t+1} \tag{16}$$

DEPARTMENT OF CYBERNETICS

Overfitting

- Overfitting is a common phenomena when training neural networks
- Very good results on training data, very bad results on testing data



Reducing Overfitting - Data Augmentation

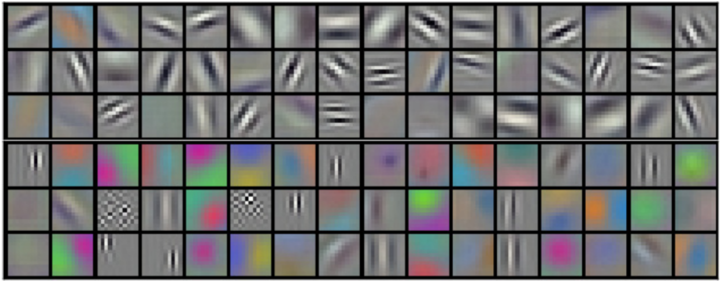
- It is the easiest way of fighting overfitting
- By applying label preserving transformations and thus enlarging the dataset
- Different methods (can be combined):
  1. Taking random (large) crops of the images (and resizing)
  2. Horizontal reflection
  3. Altering the RGB values of pixels
  4. Small geometric transformations

Reducing Overfitting - Dropout

- This method tries to make the individual neurons independent on each other
- Mostly used with fully connected layers
- We set a probability of dropout p_d
- For each training batch we set output of a neuron to be zero with probability p_d
- Is often implemented as a layer

DEPARTMENT OF CYBERNETICS

Examples - learned kernels



- These are the kernels of the first layer from AlexNet
- Trained no ImageNet 1000-classes, roughly 1.2 millions of training images

DEPARTMENT OF CYBERNETICS

Examples - results