Lesson 09

Convolutional Neural Network - Advanced Techniques

Ing. Marek Hrúz, Ph.D.

Katedra Kybernetiky
Fakulta aplikovaných věd
Západočeská univerzita v Plzni

Nesterov Momentum

- This method is already well established and recommended to use
- The idea is – first go to the point where we should end up (the momentum gradient direction v^t)
- Then correct the estimate by computing the gradient in the "look-ahead" point

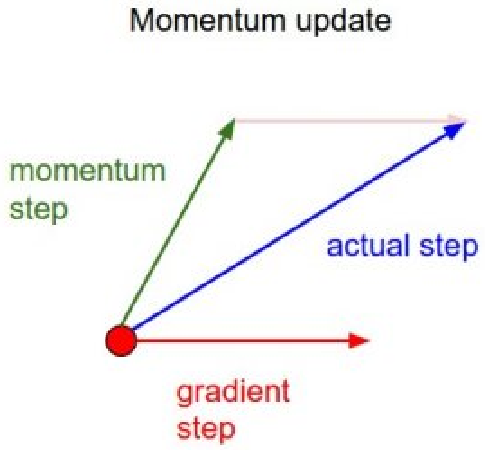
$$\omega^{t+1} = \omega^t + v^t - \epsilon \cdot \nabla L\left(\omega^t + v^t\right) \tag{1}$$

- where $v^t = \alpha \cdot v^{t-1} - \beta \cdot \epsilon \cdot \omega^{t-1} - \epsilon \cdot \nabla L\left(\omega^{t-1}\right)$ is the momentum
- For comparison with classical momentum:

$$\omega^{t+1} = \omega^t + v^{t+1} \tag{2}$$

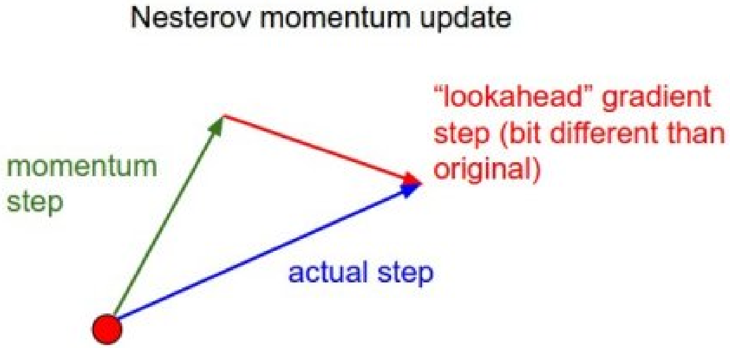
DEPARTMENT OF CYBERNETICS

Nesterov Momentum vs. Momentum



DEPARTMENT OF
CYBERNETICS

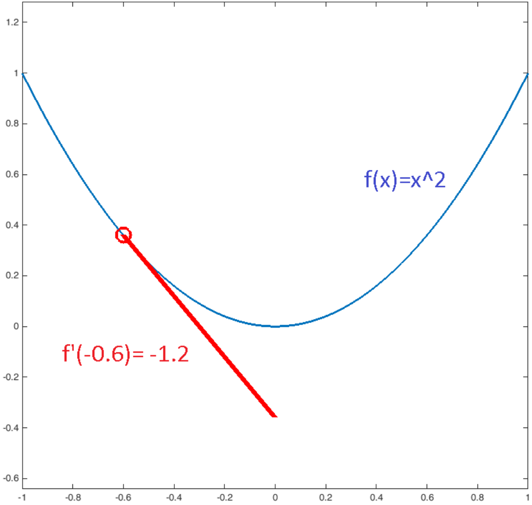
Nesterov Momentum vs. Momentum



Per-parameter adaptive learning rate methods

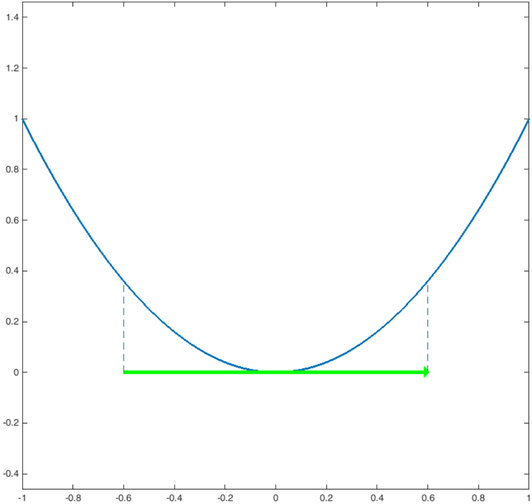
- There are drawbacks of static learning rates
- The learning rate is also global - bad
- There is a way to estimate the learning rate from the gradient estimate
- The gradient is the direction and rate of the largest growth of a function
- But we apply the gradient in the **domain** of the function

Gradient visualization



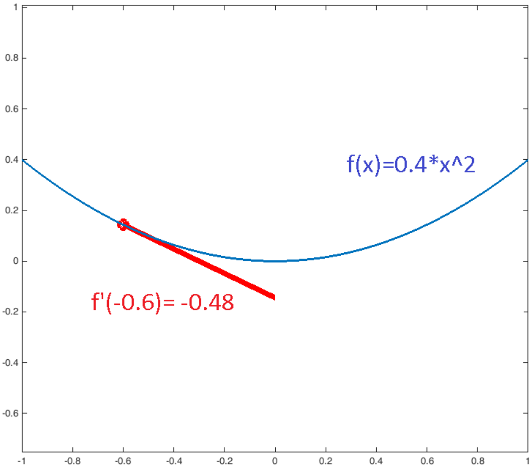
DEPARTMENT OF CYBERNETICS

Gradient application



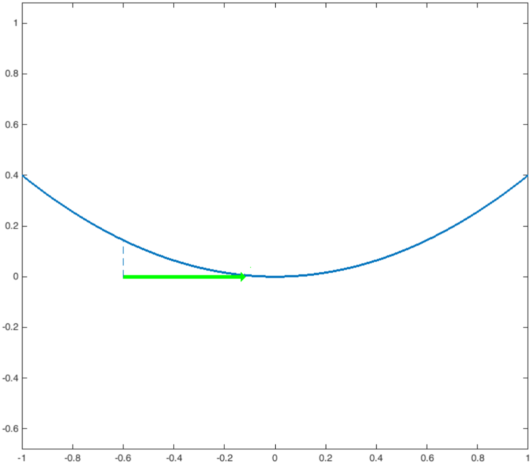
DEPARTMENT OF CYBERNETICS

Gradient visualization



DEPARTMENT OF CYBERNETICS

Gradient application



Per-parameter adaptive learning rate methods

- A steep function has a large gradient - but steepness (intuitively) means that we are close to local extreme - the step should be small
- A shallow function has a small gradient - but shallowness (intuitively) means that we are far from extreme - the step should be large
- We can make use of this - we apply an adaptive learning rate

Adagrad

- We introduce an parameter of cumulative squared gradient magnitudes:

$$\sigma_i = \sigma_i + \|\nabla J\left(\omega_i^t\right)\|_2^2 \tag{3}$$

$$\omega_i^{t+1} = \omega_i^t - \frac{\epsilon}{\sqrt{\sigma_i + \xi}} \nabla J\left(\omega_i^t\right) \tag{4}$$

- where ξ is a small constant, ω_i^t is a vector of weights of i^{th} neuron
- The method is aggressive and updates of gradients go to zero (since σ_i always grows)

Adadelta & RMSProp

- Adadelta is the reaction to weak point of Adagrad (dying updates)
- The always growing magnitude of gradient history is replaced by a running average

$$\sigma_i = \gamma\sigma_i + (1-\gamma)\left\|\nabla J\left(\omega_i^t\right)\right\|_2^2 \tag{5}$$

- The learning rate is also approximated (we want it to have the same hypothetical units as gradient) as a running average of parameter updates

$$\epsilon^{t+1} = \gamma\epsilon^t + (1-\gamma)\,\Delta\omega_i^t \tag{6}$$

- Then we can write the update as

$$\omega_i^{t+1} = \omega_i^t - \frac{\sqrt{\epsilon^t + \xi}}{\sqrt{\sigma_i + \xi}}\nabla J\left(\omega_i^t\right) \tag{7}$$

- Independently discovered RMSProp neglects the learning rate approximation

Adam

- Adaptive Moment Estimation (Adam) additionally approximates the running average of non-squared gradients (first moment)

$$m_i = \beta_1 m_i + (1 - \beta_1) \nabla J \left(\omega_i^t\right) \tag{8}$$

- The running average of squared gradient magnitudes is kept (second moment)

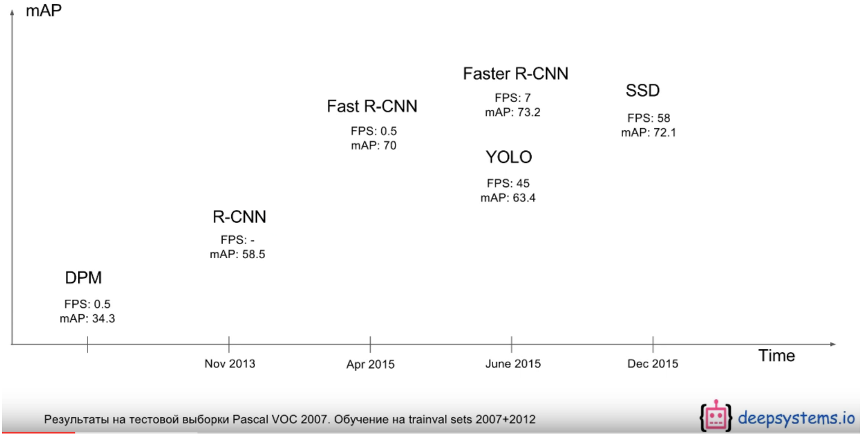
$$v_i = \beta_2 v_i + (1 - \beta_2) \|\nabla J \left(\omega_i^t\right)\|_2^2 \tag{9}$$

- The update becomes:

$$\omega_i^{t+1} = \omega_i^t - \frac{\epsilon}{\sqrt{v_i + \xi}} m_i \tag{10}$$

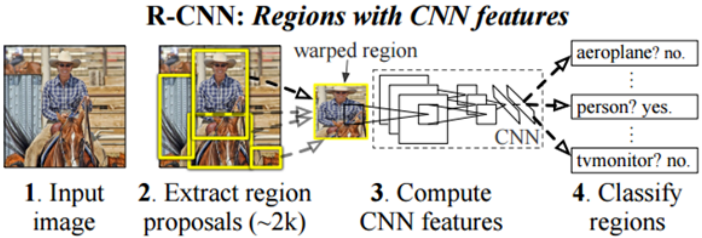
DEPARTMENT OF CYBERNETICS

Object Detecting Networks - time-lapse



Region based CNN

- R-CNN
- Propose regions - selective search, that is merging of super pixels (2 sec per image)
- Extracts features from warped proposals
- Train per class SVM on CNN features

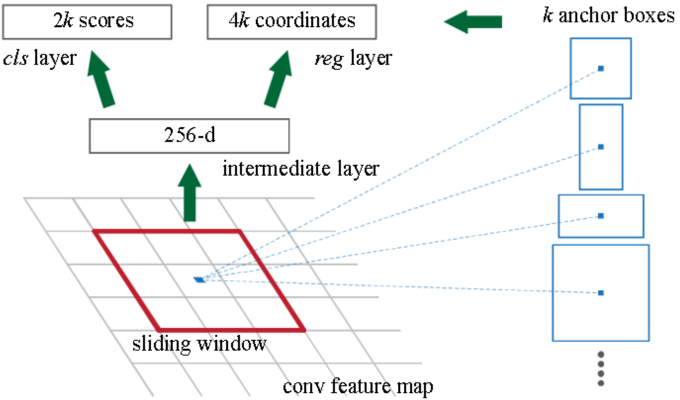


DEPARTMENT OF CYBERNETICS

Region Proposal Network

- Faster R-CNN
- Two modules that share parameters:
  - Region proposal model
  - Classification module
- Region proposal - a small CNN is applied to the last layer of a pre-trained CNN (VGG-16; $14 \times 14 \times 512$)
  - The small CNN is a convolutional layer (with $n \times n$ kernels) followed by a fully connected layer (512 dim)
  - After that there are two FC layers - $2k$ neurons for objectiveness and $4k$ neurons for location

Region Proposal Network



Region Proposal Network

- k is the number of anchor boxes
- The anchors have different sizes and aspect ratios
- In original paper they use 3 sizes and 3 ratios
- The outputs of the $2k$ and $4k$ FC layers are relative to the position and size of the appropriate anchor box

Region Proposal Network - learning

- The targets are deduced from the ground truth data (annotated regions of objects)
- Each anchor that has Intersection over Union (IoU) greater than 0.7 is considered a positive sample
- The objectiveness of such anchor (in $2k$ FC layer) is set to positive
- The regression of the anchor (position and size) is computed from the ground truth region
- Negative anchors are such that have IoU lower than 0.3
- The loss is defined as:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right) \tag{11}$$

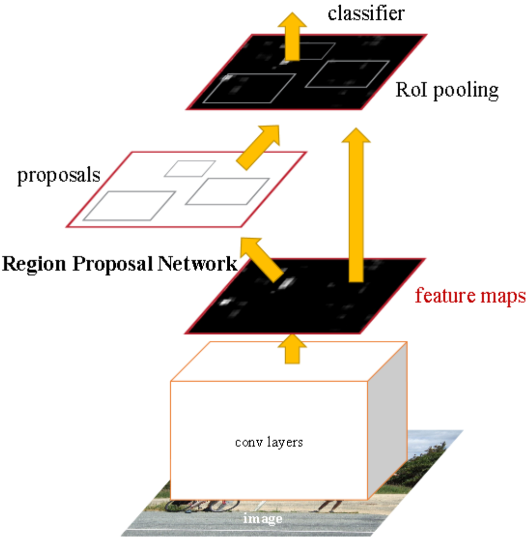
- L_{cls} is a 2 dim softmax, L_{reg} is a smoothed L^1-norm

DEPARTMENT OF CYBERNETICS

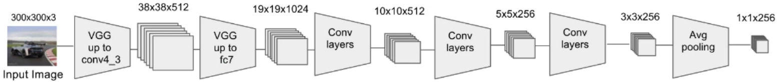
Region Recognition - Detection Network

- Uses Fast R-CNN
- The weights are shared - the same features are used for proposal and detection
- Each proposed region is recognized - but watch out! the regions have different sizes
- That is why you need to use the ROI max-pooling
- This is quite hard to implement and is deprecated by Single Shot Multi-Box Detector
- ... thus, need not to learn (YEAH!)

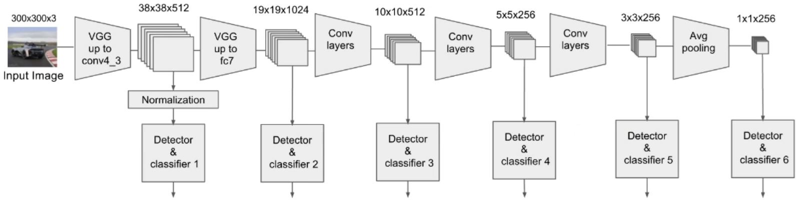
Region Recognition - Detection Network



Single Shot Multi-Box Detector (SSD)

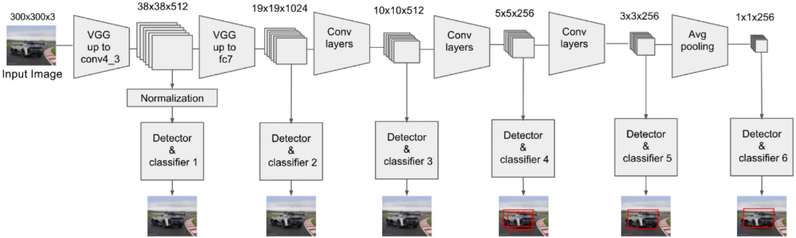


Single Shot Multi-Box Detector (SSD)

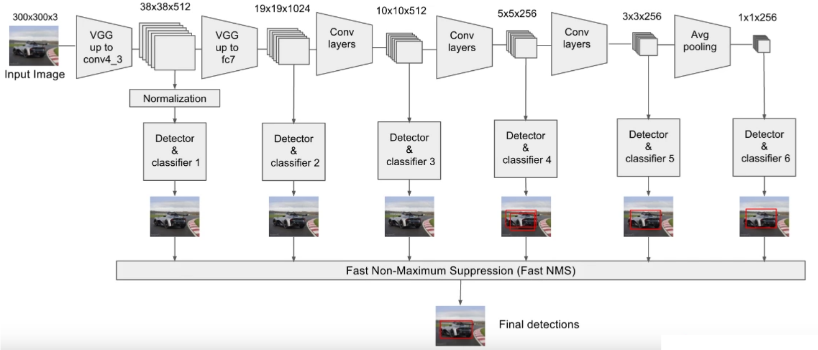


DEPARTMENT OF
CYBERNETICS

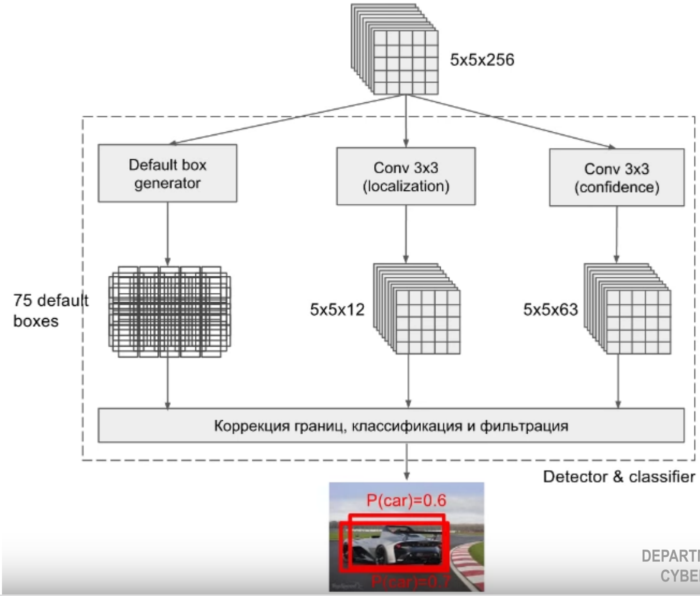
Single Shot Multi-Box Detector (SSD)



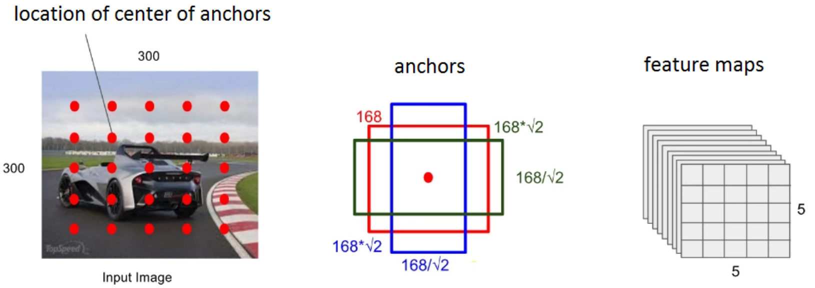
Single Shot Multi-Box Detector (SSD)



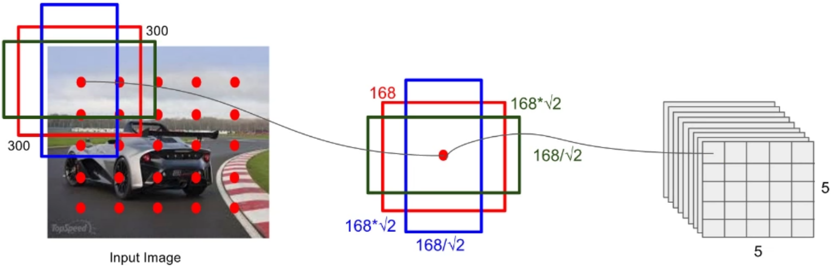
Detection and Classification



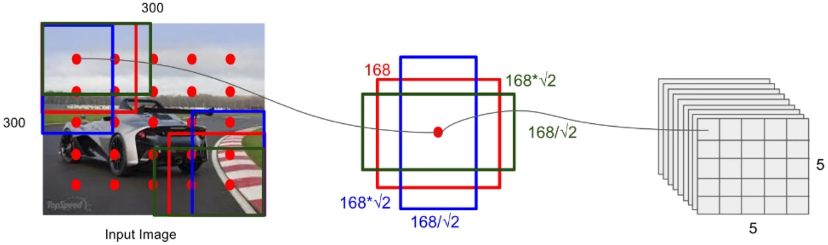
Default Boxes



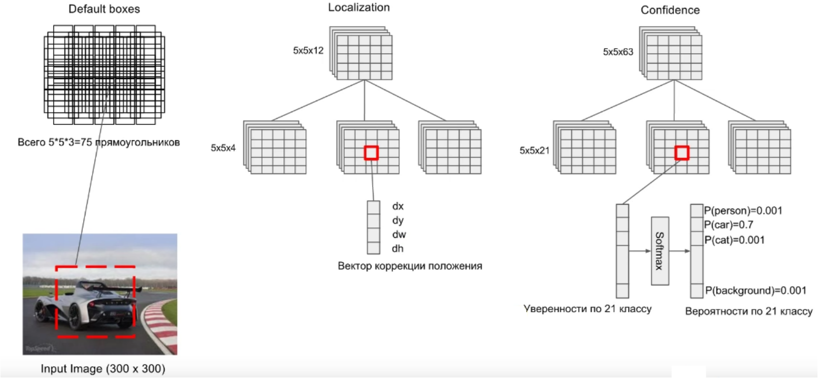
Default Boxes



Default Boxes

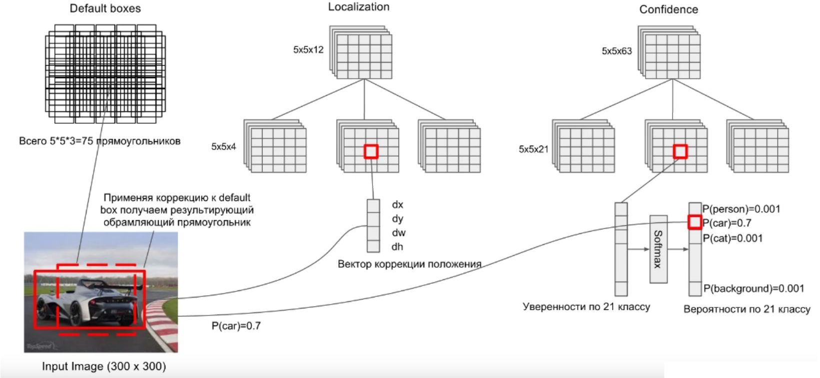


SSD - all together



DEPARTMENT OF CYBERNETICS

SSD - all together



Training of SSD

- Each ground truth box is matched to the default box (anchor) with the best Jaccard overlap (IoU)
- Each GT box thus has only one matching default box
- This is extended by adding default boxes with at least 0.5 Jaccard overlap
- From these matches the deltas of x, y, width, and height are computed and also the classifier gets its label

Training Objective

$$L(x, c, l, g) = \frac{1}{N}\left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right) \tag{12}$$

- x is the predicted class, c is the GT class, l is the regressed deltas of default boxes, g is the GT deltas
- N is the number of matched default boxes
- L_{loc} is a smooth L^1 loss, L_{conf} is softmax

$$L_{loc} = \sum_i \text{smooth}_{L1}(l_i - g_i) \tag{13}$$

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{14}$$

DEPARTMENT OF CYBERNETICS

Hard negative mining

- After matching a lot of default boxes will be negatives (background class)
- This gives an imbalanced training set with lots of default boxes
- The negatives are ranked according the confidence
- Only a portion of the negatives if chosen (3:1) for the gradient update
- Data augmentation:
  - Use the entire original image
  - Sample the original image so that the patches have IoU with the object at least 0.1, 0.3, 0.5, 0.7, or 0.9
  - Randomly sample the image into patches