

Slovak university of technology in Bratislava
Faculty of informatics and information technologies

Martin Schnürer

Deep neural network for human detection

DP1 report

Intelligent Software Systems

9.2.5 Software engineering, 9.2.8 Artificial intelligence

Ústav informatiky, informačných systémov a softvérového inžinierstva FIIT STU v Bratislave v spolupráci s
ÚI SAV Bratislava

Supervisor: Ing. Peter Malík PhD.

May 2018

Zadanie diplomovej práce

Meno študenta: **Bc. Martin Schnürer**

Študijný program: Inteligentné softvérové systémy

Študijný odbor: Softvérové inžinierstvo – hlavný študijný odbor
Umelá inteligencia – vedľajší študijný odbor

Názov práce: **Detekcia osôb hlbokými neurónovými sieťami**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujete diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvoríte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh hlbšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riadte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnete zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU v Bratislave v spolupráci s ÚI SAV Bratislava

Vedúci práce: **Ing. Peter Malík, PhD.**

Termíny odovzdania:

Podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti:
home > Informácie o > štúdiu > harmonogram štúdia > diplomový projekt.

V Bratislave dňa 12. 2. 2018

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky, informačných systémov
a softvérového inžinierstva

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce¹

Študent:

Meno, priezvisko, tituly: Martin Schnürer, Bc.
Študijný program: Inteligentné softvérové systémy
Kontakt: xschnuren@is.stuba.sk

Výskumník:

Meno, priezvisko, tituly: Peter Malík, Ing. PhD.

Projekt:

Názov: Detekcia osôb hlbokými neurónovými sieťami
Názov v angličtine: Deep neural network for human detection
Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava
Oblasť problematiky: Hlboké učenie a konvolučné neurónové siete

Text návrhu zadania²

Počas posledného desaťročia priťahla detekcia osôb výraznú pozornosť v sfére počítačového videnia a rozoznávania vzorov najmä v dôsledku širokej škály úloh, na ktoré môžu byť aplikovateľné. Schopnosť spoľahlivo rozoznať polohu osoby na obrázku alebo videu je náročnou úlohou, s ktorou sa stretávame pri kamerovej detekcii osôb v priestore alebo u autopilota pri jazde autom. Detekcia osôb je jedným z najťažších problémov v kategórii detekcii objektov, nehovoriac o rozoznávaní polohy osoby na obrázku. Problém detekcie osôb môže byť jednoducho poňatý ako lokalizácia subjektov na danom obrázku, pričom subjekty sú konkrétne označené ohraničenou oblasťou na obrázku.

Analyzujte súčasný stav a navrhnete riešenie problematiky detekcie osôb a klasifikácie ich polohy na obrázku pomocou hlbokých konvolučných neurónových sietí. Navrhnete efektívny model hlbokéj neurónovej siete a vhodnú topológiu. Redukujte počet parametrov tohto modelu a maximalizujte jeho presnosť. Vyhodnoťte a porovnajte experimentálne výsledky detekcie osôb a klasifikácie ich polohy a zhodnoťte príspevok k aktuálnej problematike.

¹ Vytlačíť obojstranne na jeden list papiera

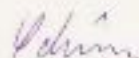
² 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra³

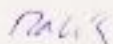
- Szegedy, C. - Toshev, A. - Erhan, D.: Advances in Neural Information Processing Systems: Deep Neural Networks for Object Detection :NIPS, 2013: 9s:
- Nguyen, D. - Li, W. - Ogunbona P.: Pattern Recognition: Human detection from images and videos: Elsevier March 2016, Pages 148-175.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Martin Schnürer, konzultoval(a) a osvojil(a) si ho Ing. Peter Malík, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 17.1.2018



Podpis študenta

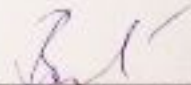


Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schvájený: áno / nie⁴

Dňa: 16. 2. 2018



Podpis garanta predmetov

³ 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uvedte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

⁴ Nehodiace sa prečiarknite

1. Plan for winter semester (Diploma thesis part 2)

Week	Plan
1.	Choose, download and prepare dataset, load to the working environment (First try COCO dataset)
2.	Download Mask R-CNN model and prepare environment
3.	<ul style="list-style-type: none">- Download pretrained weights, apply on Mask R-CNN.- Execute model, try to run and classify some images.
4.	Figure out how to adapt model to predict only persons (pedestrians).
5.	<ul style="list-style-type: none">- read papers, how to reduce classification set only to person detection.
6.	Reduce classification set - implementation on model.
7.	(Adding layers, configuring hyperparameters...)
8.	
9.	Choose a baseline, download and prepare (runnable version).
10.	Implement evaluation metrics for baseline, previous model of Mask R-CNN and modified model.
11.	
12.	Process of evaluation. Try different settings and hyperparameters.
13.	

2. Keywords

Convolutional neural network

Convolutional neural network is very similar to ordinary neural network. Layers of neurons with activation functions are present. Layers are connected with learnable weights and biases (neurons with constant output multiplied by learnable weight).

Difference between basic neural network is that convolutional neural network (CNN) makes the explicit assumption, that the input are images.

Both networks has layers, but their structure and mechanism of how they work are different. Pooling, convolutional and ReLu layer are the most common types of layers CNN has. Every type has different purpose and makes different transformations with the input. Advantage of CNNs is their image movement-insensitivity (translational invariant) - in case that the object on the image has significantly changed its position, CNN is still capable of successfully recognizing that object.

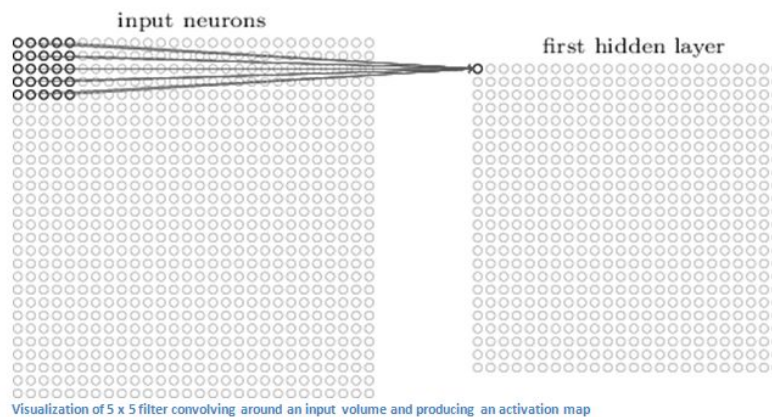
Convolution layer

In convolution layer we process image through the process called convolution. To execute process we need *convolution filter* with size $w \times h$, and the image to apply the filter on. It turns out, that near pixels create together important feature and it's more likely that two near pixels create important feature than the one with the pixel on the other side of the image. Therefore, convolution filter computes dot product on the set of neighbour pixels and the output of convolution is array of dot products of the convolution. Often more than one convolution filters are used.

Convolution process

On the input is usually presented three dimensional array - height, width and channel depth (rgb) of the image. Each individual filter in a filter bank is called a kernel and it has a fixed receptive field (window) that is scanned over a layer below it to compute an output feature map. The kernel does a simple dot product + bias computation as it scans the layer below it and then feed the result through an activation function, rectifier for example, to compute the output map. The output map is then subsampled using sum or max pooling, the later being more common in order to reduce sensitivity to distortions of stimuli in the upper layers.

A valid convolution process over the given input with window of size $W \times H$ produce output which has reduced dimensions. That is caused by the process, that we shift window right to the pixels, for which is process of applying filter valid. For the given window 5×5 , the output from given input $w \times h$ would be reduced to $(w-4) \times (h-4)$. This is often not desired and we want the output dimensions to remains the same. What we can do is “border” the input with zeros, this border then will have thickness of 2. This is also called **zero-padding**. We can also adjust window shift movement. The number of pixels we move window is named **stride**. As the stride grows, output dimensions shrink as well.



Pooling layer

Pooling layer in CNN is used to reduce variance, reduce computation complexity (as 2×2 max pooling/average pooling reduces 75% data) and extract low level features from neighbourhood.

Pooling layer is used to make a dot product from window of size $w \times h$ applied on the input array.

Dot product is computed as aggregation on the window which is located on the specific position on the image. There are several possible schemes used for aggregation, the most popular is max-pooling. (others are *min-pooling*, *average-pooling*). For example average-pooling is used for “smoothing” data and “max-pooling” for most important feature extraction line edges.

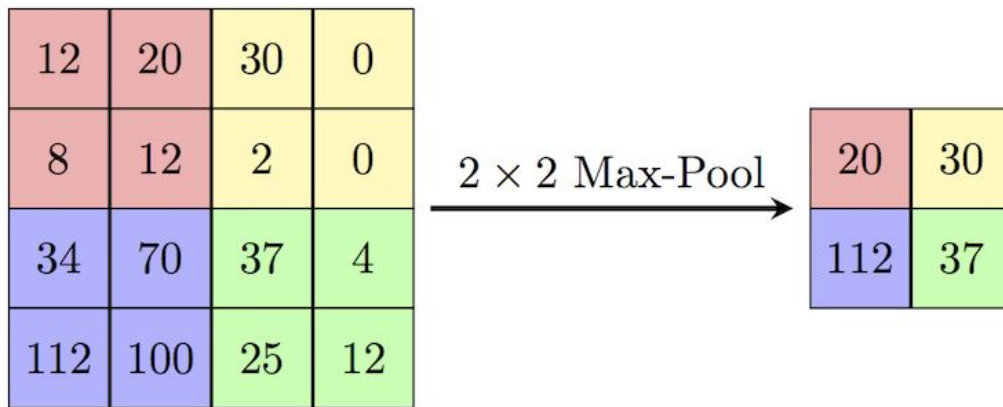


Figure 1 - Pooling layer with Max pooling (extract maximum value)

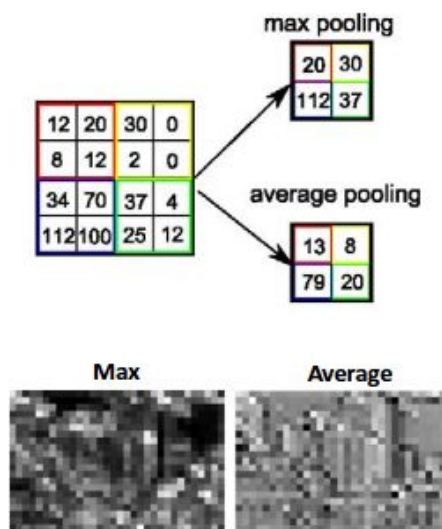


Figure 2 - Difference between max and average pooling

Learning (Backpropagation)

Backpropagation is used to adjust weights between layer to the state, where the model of neural network give us the best (generalised) results. In the output layer, the error is computed, then error is (back)propagated into previous layers - then adjust weights for a fraction correlated with error produced. This fraction is multiplied by *learning rate* to avoid to change the weight too much - so after the change, it would produce higher error than previously (then again, adjustment will be too high and weight values explode to infinity).

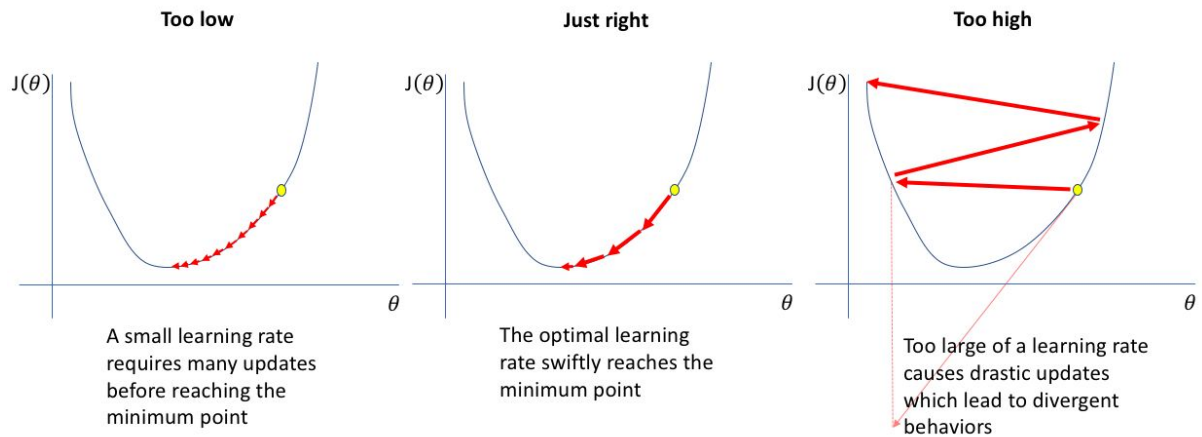


Figure 3 - learning process at different values of learning rate

Stochastic gradient descent

Stochastic gradient descent (SGD) is an iterative optimization algorithm. Algorithm is trying to find (local) minimum (or maxima), adjusting weights with the constant learning rate. Intuitive way to understand gradient descent optimisation is to imagine flow of the mountain river, which has the purpose to get to the lowest height as possible - in this case, produced error of neural network is the river height - we are trying to minimize error as much as possible.

Adam optimizer (*adaptive moment estimation*)

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data. The method is based on moving averages and dynamic change of the learning rate and momentum.

Overfitting

Overfitting is a problem in machine learning where the model trains very well on the data that you have, but performs poorly on data that it has not seen before.

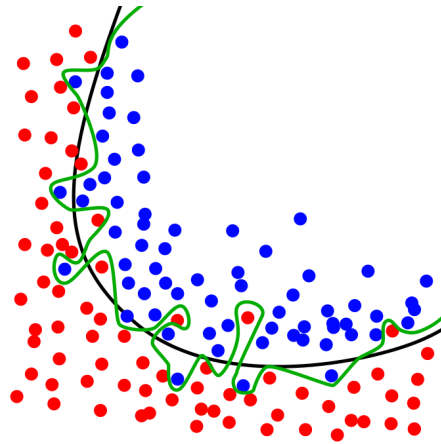


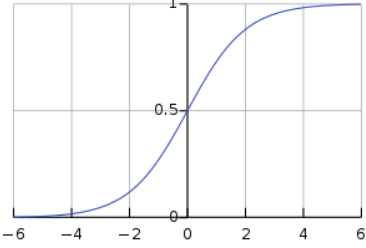
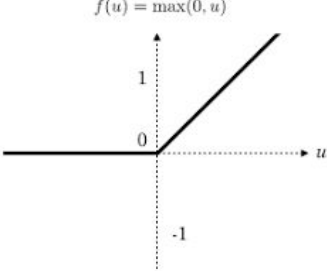
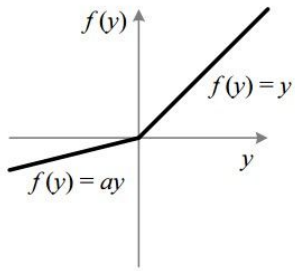
Fig. 4 - Green line is prediction from overfitted neural network (not well generalised). Black line is good generalisation (separation) of two classes (in example).

Overcoming Overfitting

1. Dropout - Invented by Geoffrey Hinton himself, the term "dropout" refers to dropping out units (both hidden and visible) in a deep learning model so that it doesn't create too much dependencies on one neuron for computing the output
2. Pooling - The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality - Example - in a 4x4 matrix of numbers, you can take the max of 4 numbers at a time and reduce it to a 2x2 matrix. You still preserve essential information and reduce overfitting
3. Batch Normalization - Similar to dropout in the sense that it multiplies each hidden unit by a random value at each step of training. In this case, the random value is the standard deviation of all the hidden units in the minibatch. Because different examples are randomly chosen for inclusion in the minibatch at each step, the standard deviation randomly fluctuates. Batch norm also subtracts a random value (the mean of the minibatch) from each hidden unit at each step. Both of these sources of noise mean that every layer has to learn to be robust to a lot of variation in its input, just like with dropout

Activation functions

Activation function of a neuron defines the output of that neuron given an input

Sigmoid function	ReLU function	Leaky ReLU
		
<p>Advantages:</p> <ul style="list-style-type: none"> - normalisation to range specific range (boundaries from bottom to top) <p>Disadvantages:</p> <ul style="list-style-type: none"> - vanishing gradient - expensive to compute 	<p>Advantages:</p> <ul style="list-style-type: none"> - easy to compute - Sparsity of layer (when $a < 0$) after activation - no vanishing gradient problem <p>Disadvantages:</p> <ul style="list-style-type: none"> - dead neurons (explained below) - because of (only) positive bias shift learning is slower - sometimes hard to learn (no top boundary) 	<p>Advantages:</p> <ul style="list-style-type: none"> - prevent dead neurons <p>Disadvantages:</p> <ul style="list-style-type: none"> - Not sparse anymore

Dead neuron at ReLU

If the output of a ReLU is consistently 0 (for example, if the ReLU has a large negative bias), then the gradient will consistently be 0. The error signal backpropagated from later layers gets multiplied by 0, so no error signal ever passes to earlier layers. The ReLU has died.

Some papers also claim that it is better to use ReLU and at the process of training monitor the fraction of “dead” neurons and possibly use leaky ReLU. It is worth to use Tanh activation function, but it is expected to work not as good as ReLU.

Vanishing gradient

During neural network training with backpropagation, the (local) minimum of the error function is found by iteratively taking small steps in the direction of the negative error derivative with respect to networks weights. With each subsequent layer the magnitude of the gradients gets exponentially smaller (vanishes) thus making the steps also very small which results in very slow learning of the weights in the lower layers of a deep network.

Passthrough layer

A Passthrough connection was first highlighted by the Residual Network to achieve higher accuracies for the 2016 Imagenet competition. This led to networks that were much deeper than previous networks. The clear distinguishing characteristic were connections from lower layers to higher layers that would bypass a layer or several layers. Residual Networks and other networks that have taken advantage of this technique have been shown to have higher prediction accuracy. This leads to the conclusion that Deep Networks should not necessarily maintain a strict hierarchy where information is always truncated as it flows from a lower to a higher layer (i.e. upstream). Rather, the added flexibility of flowing information that bypasses layers has a clear benefit. It seems that providing more pathways of information flow allows networks greater flexibility to more accurately model the domain.

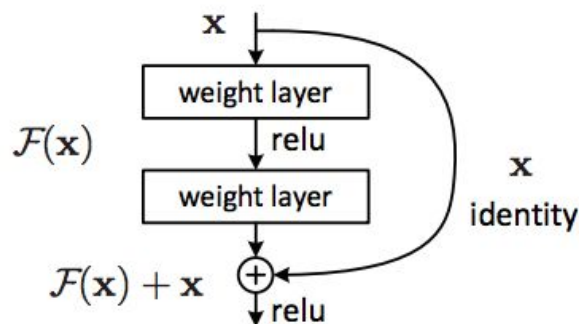


Fig. 5 - passthrough layer - jump over 2 layers

IoU - Intersection over Union

Intersection over union technique is widely used in image recognition as accuracy evaluators of predicted bounding box on object. It is a ratio between intersected area of ground-truth bounding box and predicted bounding box and union area between these two bounding boxes.

0.5 and higher score of IoU is considered a pretty good prediction of object position.

Region proposal network

A region proposal network (RPN) is a shallow fully connected neural network (NN) first introduced in the Faster R-CNN (Faster region convolutional neural network) for proposing regions with a high probability of containing an object of interest.

The RPN does not specify the class of the object in the proposed regions, it is merely trained to output a score that measures “objectness” as a probability measure. Then a threshold is applied to the objectness score and those regions associated with the score above that threshold are further processed.

Bounding box

Bounding box annotations are used for localization. In traditional sense convolutional neural networks are always used for image classification (although now unimaginable things are being done with it). In localization we not only classify the image but also finds the location of the dominant object present in the image by predicting the values of the bounding boxes (for eg top left corner coordinates and bottom right corner coordinates).

Region Proposal Network

The region proposal network (RPN) in the faster region-based convolutional neural network (Faster R-CNN) is used to decide “where” to look in order to reduce the computational requirements of the overall inference process. The RPN quickly and efficiently scans every location in order to assess whether further processing needs to be carried out in a given region. It does that by outputting k bounding box proposals each with 2 scores representing probability of object or not at each location.

Region of interest

Region of interest pooling (also known as RoI pooling) is an operation widely used in object detection tasks using convolutional neural networks. For example, to detect multiple cars and pedestrians in a single image. Its purpose is to perform max pooling on inputs of nonuniform sizes to obtain fixed-size feature maps (e.g. 7×7)

3. Previous work

In the last few years a lot of papers have been published. In the [1] the following architecture for pedestrian was proposed the architecture on the Figure X. A common pipeline for pedestrian detection is:

1. Region proposals (this also composed from *sliding window* and *selective search*)
2. Feature extraction (HOG, LDCF, Deep learning)
3. Region classification (AdaBoost, SVM)

For region proposal they (tried and) used:

- Sliding window
- Selective search [1]
- **Locally Decorrelated Channel Features [2]**

Their final model contain LDCF region proposal algorithm.

Lately, in the leightweight version they according to the paper replace region LDCF proposal algorithm with ACF [3]. The paper submitted different results in cases where neural network was used:

There were used both AlexNet and GoogleLeNet

- They claim, that general CNN neural network focused on general object recognition has much worse results for pedestrian detection, that trained CNN used only for pedestrian detection.
- Results show different results both for different CNN trained and proposal region algorithms.

Deep learning with deep network cascades - Deep network cascades was first used in [Ouyang et al. and Luo et al.]. The final model was capable of image processing (classification) with 15 fps.

[4, 5, 6] referenced DNN (deep neural networks) are slow for object detection. In the [7] *tiny deep network* was proposed - the lightweight version of the deep convolutional neural network. Use of tiny deep network from the work [3] made the classifier work at 2

FPS, thus speeding it up by 80x. Finally, we insert a fast feature cascade [4] with which our method achieves a real-time 15 FPS solution. Previous architecture were adopted from [Krizhevsky et. al.] and depth, size of receptive fields and other hyperparameters were reconfigured.

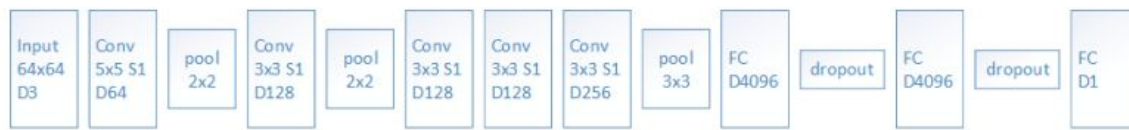


Fig. 6 - first attempt of lightweight neural network, this didn't produce desirable results.



Fig. 7 - Final attempt of lightweight version,

To the network was then passed regions of image with high confidence value (configurable threshold).

YOLO v1

In this work [8] new approach to object detection is presented.

Prior work on object detection repurposes classifiers to perform detection. Instead, this method frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities.

YOLO makes more localization errors but is less likely to predict false positives on background.

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

YOLO uses a single convolutional network which simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO achieves more than twice the mean average precision of other real-time systems. YOLO makes less than half the number of background errors compared to Fast R-CNN. YOLO still lags behind state-of-the-art detection systems in accuracy.

While it can quickly identify objects in images it struggles to precisely localize some objects, especially small ones.

Method: Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

The work was compared to other works, but mostly to R-CNN, which had the best precision but was not realtime (0.5 fps with 73.2 mAP accuracy).

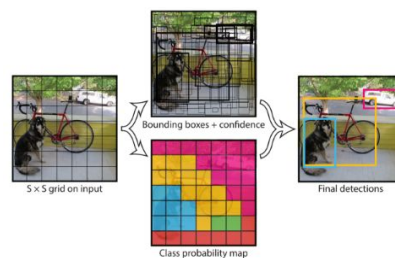


Fig. 8 - detection by YOLO

Every prediction from one cell contains 5 outputs x, y, w, h, c where:

x, y - BB center position

w, h - BB width and height

c - confidence value (probability of class)

Then the whole classifier outputs tensor:

$S \times S \times (B \times 5 + C)$

$S \times S$ - cells

B - Bounding boxes for one cell

5 - 5 outputs for each cell

C - number of classes to predict (cats, dogs, etc.)

YOLO is presented as one of the fastest classifiers, even Fast YOLO version was created and tested and it reached 155 fps, but with slightly less accuracy.

Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45

At comparison with R-CNN it turns out YOLO has worse results at object localisation.

19% localisation error was produced at YOLO and 8.6% at R-CNN. On the contrary, R-CNN got worse results at localising objects in the background (13.6%), where YOLO produced only 4.75%.

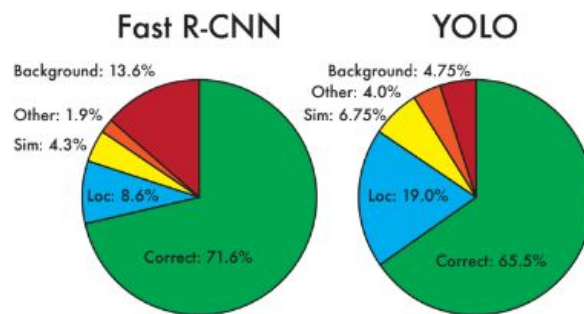


Fig 9 - comparison in correct predictions and errors between Fast R-CNN and YOLO

YOLO v2

YOLO v2 [9] has some improvements compared to the previous version:

Batch normalisation:

- Dropout layer removed, instead used batch normalisation - increased 2% mAP.

Training on high resolution images:

- Previous version was trained on an images with 256 x 256 resolution. In version 2 higher resolution images was used for training and model increased in 4% mAP

Convolutional with anchor boxes:

- YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. Instead of predicting coordinates directly Faster R-CNN predicts bounding boxes using hand-picked priors. Using only convolutional layers the region proposal network (RPN) in Faster R-CNN predicts offsets and confidences for anchor boxes. Since the prediction layer is convolutional, the RPN predicts these offsets at every location in a feature map. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn. In this version, fully connected layer was removed and bounding boxes was predicted by convolutional layer. Network was shrunk to operate on 416 input images instead of 448×448. YOLO's convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13×13 . Following YOLO, the objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object.

Without anchor boxes - 69.5% mAP, recall = 81%

With anchor boxes - 69.2% mAP, recall =88%

Dimension clusters

- Disadvantage of anchor boxes is that they have to be hand-picked. Therefore k-means clustering helps to appropriately adjust dimensions of the boxes. Paper found the best results at $k = 5$ with a good tradeoff between recall and complexity of the model

Direct location prediction

- When using anchor boxes, there is an issue with model instability, especially during early iterations. Most of the instability comes from predicting the x, y locations for the box. In region proposal networks the network predicts values t_x and t_y and the x, y center coordinates are calculated as:

$$\begin{aligned}x &= (t_x * w_a) - x_a \\ y &= (t_y * h_a) - y_a\end{aligned}$$

- This formulation is unconstrained so any anchor box can end up at any point in the image. With random initialization the model takes a long time to stabilize to predicting sensible offsets.
- In the version 2 center position of the bounding box is computed relative to the grid cell. With logistic activation function offset value falls in range 0 and 1.

Mask R-CNN

Mask R-CNN [10] is conceptually simple, flexible and generalised for object instance segmentation. Mask R-CNN is the extension of previous Fast R-CNN, where the new branch for segmentation is added. Segmentation is then applied for each Region of Interest (RoI) in parallel with the existing branch for classification and bounding box (BB) regression. The mask branch is a small fully convolutional network (FCN) applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. The model is considered to be flexible and simple to implement new architecture design.

RoIAlign is the method for preserving exact spatial locations between inputs and outputs - in the previous Fast R-CNN was presented RoIPool method, which performs coarse spatial quantization for feature extraction. Therefore RoIAlign was proposed as simple, quantization-free layer.

It turns out that RoIAlign had a large impact and improves mask accuracy by relative 10% to 50%, showing bigger gains under stricter localization metrics. They found essential to decouple mask and class prediction, so they predict binary mas for each class independently, without competition among classes and rely on the network's RoI classification branch to predict the category.

In some works segmentation precedes recognition, which is slow and less accurate. Instead, this method is based on parallel prediction of masks and class labels. Mark R-CNN is based on *instance-first* strategy, that means, it doesn't mix already segmented pixels into another categories like other *segmentation-first* strategies.

Faster R-CNN consists of two stages. In the first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage, which is in essence Fast R-CNN, extracts features using RoIPool from each candidate box and performs classification and bounding-box regression.

Mask R-CNN adopts the same method used at Faster R-CNN, but predicts label and bounding box in parallel with segmentation (binary mask) for each RoI.

Loss is then computed as $L = L_{cls} + L_{box} + L_{mask}$. The mask is Km^2 where K is number of encoded classes and m is $m \times m$ resolution of RoI (Region of Interest). Then Loss is only computer for k -th class - other mask outputs do not contribute to the loss. This definition of L_{mask} allows the network to generate masks for every class without competition among classes.

RoIAlign was introduced as the replacement for RoiPool, because of misalignment produced with quantization process of RoIPool feature extraction. Misalignment was presented between the RoI and the extracted features. While this may not impact classification, which is robust to small translations, it has a large negative effect on predicting pixel-accurate masks. In the RoIAlign bilinear interpolation is used to compute the exact values of the input features at four regularly sampled locatons in each RoI bin and then aggregate the result (using max or average).

4. References

- [1] Uijlings, J., van de Sande, K., Gevers, T., Smeulders, A., 2013. Selective search for object recognition. International Journal of Computer Vision.
- [2] [Nam, W., Dollar, P., Han, J. H., 2014. Local decorrelation for improved ´ pedestrian detection. In: 28th Annual Conference on Neural Information Processing Systems (NIPS).]
- [3] [Dollar, P., Appel, R., Belongie, S., Perona, P., Aug 2014. Fast feature pyramids for object detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on 36 (8), 1532–1545]
- [4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. <http://arxiv.org/pdf/1311.2524v4.pdf>, 2013.
- [5] A. Giusti, D. Ciresan, J. Masci, L. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. ICIP, 2013.
- [6] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. arXiv technical report, 2014.
- [7] Angelova et al. , Real-Time Pedestrian Detection With Deep Network Cascades
- [8] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, University of Washington,, Allen Institute for AI, Facebook AI Research - You Only Look Once: Unified, Real-Time Object Detection
- [9] Joseph Redmon, Ali Farhadi - YOLO9000, University of Washington, Allen Institute for AI
- [10] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, Facebook AI Research (FAIR) - Mask R-CNN