Avoiding Over-Detection: Towards Combined Object Detection and Counting

Philip T.G. Jackson and Boguslaw Obara(✉)

School of Engineering and Computing Sciences,
Durham University, South Road, Durham DH1 3LE, UK
{p.t.g.jackson,boguslaw.obara}@durham.ac.uk

Abstract. Existing object detection frameworks in the deep learning field generally over-detect objects, and use non-maximum suppression (NMS) to filter out excess detections, leaving one bounding box per object. This works well so long as the ground-truth bounding boxes do not overlap heavily, as would be the case with objects that partially occlude each other, or are packed densely together. In these cases it would be beneficial, and more elegant, to have a fully end-to-end system that outputs the correct number of objects without requiring a separate NMS stage. In this paper we discuss the challenges involved in solving this problem, and demonstrate preliminary results from a prototype system.

Keywords: Object detection · Deep learning · Overlapping objects · Clustered objects · Non-max suppression

1 Introduction

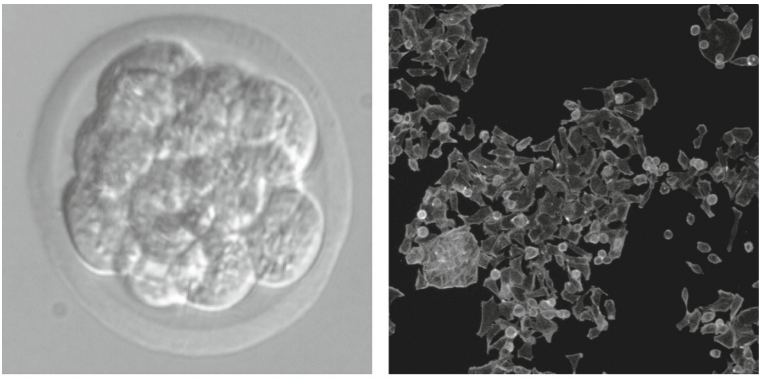
Object detection is the task of localising and classifying all objects present in an image [18]. While the field of deep learning has produced many object detection networks with excellent true positive rate, they tend to suffer from low precision, i.e. high false positive rate. Usually the network outputs many bounding boxes per object, and these over-detections are filtered by non-max suppression (NMS) [17], leaving one box per object. NMS is a fixed post-processing step that is not learnt from the data, and typically relies on a user-chosen overlap threshold (0.7 used in [16]). Furthermore, NMS is unaware of the contents of the boxes it prunes, and so has no way to know if the ground-truth boxes really do overlap.

The question arises of how it may be possible to train a deep neural network to output exactly one box per object, without the need for a separate non-learned filtering step. Aside from being more elegant, this approach may have potential for greater accuracy, particularly in the case of detecting many small, densely clustered objects. In these cases, traditional NMS may struggle to tell if two boxes overlap because they are localising the same object or if they are localising different objects which are very close. This is especially true when objects of the same class are not only close but genuinely do overlap. With very high numbers

L. Rutkowski et al. (Eds.): ICAISC 2017, Part I, LNAI 10245, pp. 75–85, 2017.
DOI: 10.1007/978-3-319-59063-9_7

of densely packed objects, another problem may also emerge: because detection networks emit a fixed number of boxes, it may become necessary to coordinate these boxes such that they are properly distributed among the many objects present. Over-detection in these cases may not only raise the false positive rate, but also lower the true positive rate; if there are only enough boxes to detect everything once then over-detecting one object may leave no boxes for another.

Close and overlapping objects occur in crowd footage, autonomous vehicle visual feeds, and histological images from biomedical microscopy, such as those in Fig. 1. In this paper we choose cell microscopy as a test case, and use the Simulating Microscopy Images with Cell Populations (SIMCEP) [10] system to generate large quantities of synthetic images with perfect ground-truth annotation for training and testing. The simplicity of this benchmark, which can be solved to reasonable accuracy without deep learning [19], allows us to focus solely on the over-detection problem. SIMCEP allows the user to generate artificial cell populations with varying degrees of clustering and overlap, and so makes an excellent testing ground for a dense object detection framework. Using simulated images allows us to generate essentially unlimited quantities of training data, bypassing the scarcity of labelled data that is normally the biggest constraint when training deep networks to solve bio-imaging problems. It is hoped that systems trained on SIMCEP images may still be applicable to real-world histological images via transfer learning. Fluorescence microscopy image analysis often requires objects to be counted as well as localised, so a one-box-per-cell system, which can be seen as combined localisation and counting, would be quite relevant in this field.



(a) (b)

Fig. 1. (a) mouse embryo, an extreme case of overlapping objects consisting of a ball of around 20 cells. (b) Human HT29 colon cancer cells, packed very closely. Both images from the Broad Bioimage Benchmark Collection [12].

2 Related Work

2.1 Deep Learning Methods for Object Detection

Object detection in deep learning is largely dominated by the Region Convolutional Neural Network (R-CNN) family of models. The original R-CNN [5] uses a selective search based method [22] to propose interesting-looking regions, only using the CNN to generate feature vectors for each region and a support vector machine (SVM) approach to then score them for each class. Fast R-CNN [4] is an iteration on this work, speeding the process up mostly by generating all convolutional features for the image in a single pass and pooling sub-sets of them for different region proposals, rather than running each proposed region through the CNN separately. Faster R-CNN [16] improves further by using the same convolutional network for both proposing regions and classifying their contents. This saves computational time and results in slightly more accurate bounding boxes, as well as being a more elegant system. Almost the whole pipeline is performed by the network, only the NMS is done separately.

Faster R-CNN is a fully convolutional network (FCN), so images of arbitrary size can be passed and the feature maps will grow or shrink accordingly. The final convolutional layer outputs feature vectors describing overlapping square regions in the image; these are used by the region proposal network (RPN) to predict a fixed number of bounding boxes per region. The RPN's output tensor consists of multiple "detectors": groups of neurons representing bounding box parameters and confidence levels. Each output box is described relative to a different fixed "anchor" box. The anchors are Faster R-CNN's answer to the problem of expressing an unordered set of boxes with a fixed-size tensor. The loss function must decide at training time which boxes from the RPN are to match with which ground-truth boxes, and which boxes should have high class probability (i.e. the RPN's confidence that box contains an object). In practice, all boxes whose anchors overlap sufficiently with a ground-truth box are trained to have high class probability and incur regression loss on their deviation from the ground-truth box. Output boxes whose anchors do not overlap sufficiently with any ground-truth box only incur loss for having high class probability. This can be seen as giving each detector a different "jurisdiction", in which it is responsible for matching any ground-truth box with a certain position, aspect ratio and size.

Another relevant detection framework is YOLO [15], which differs from Faster R-CNN principally in that is not an FCN. Although this requires that images are resized to a fixed dimension before processing, it also means that the feature maps are of constant size. This allows the final layer to be converted to a fixed-size vector that describes the entire image, in a similar manner to AlexNet [9]. This allows the classifier to make use of global image context, resulting in higher accuracy compared to Faster R-CNN, whose classifier only pools convolutional features from within the proposed bounding boxes. YOLO assigns responsibility to its output boxes in a different way to Faster R-CNN. Unlike Faster R-CNN, the jurisdictions of the detectors are not pre-defined, rather, responsibility for

detecting a given ground-truth box is assigned at training time to whichever detector outputs a box with the greatest intersection over union (IoU) with that box. The authors claim this leads to detectors learning to specialise in different sizes, aspect ratios and classes of object.

Although the above methods excel at detecting small numbers of large objects in datasets such as Pascal VOC [3], they are less well tested on large numbers of small objects. In particular, they all tend to over-detect objects, outputting many bounding boxes which must then be pruned by NMS to leave only one box per instance. The problem of learning to count has been explicitly investigated in [20], whose authors show that a network trained only on the multiplicity of a target object type will learn features that are also useful for classification and localisation of said objects. Although the results are encouraging, they do not tackle the problem of coordinating object detectors to output exactly one bounding box per ground-truth object.

2.2 Deep Learning Methods for Cell Detection

The greatest obstacle in applying deep learning approaches to biomedical image processing is the scarcity of labeled training data. Deep neural networks generally require many thousands of labeled images to train effectively, but individual problems in biomedicine tend to avail neither thousands of images nor enough trained experts to label them all. Many proposed methods [1,11,14] circumvent this problem by using CNNs to perform pixel-wise binary classification. These networks take small image patches as input and output the probability of the central pixel in the patch being part of a target object. Although this is a harder task than whole-image classification, it can yield thousands of training examples per image, since each pixel and its neighbourhood becomes an example in the training set. For example, [2] trains a CNN to identify the central voxels of zebrafish dopaminergic neurons in 3D images. This is part of a larger pipeline, which first uses an SVM to narrow down the set of potential voxels, so that the CNN need not be applied to every possible location in the image. The output probability map is then smoothed and individual cells are detected as local probability maxima. [14] uses a CNN to detect lipid deposits in retinal images, by classifying the central pixel of 65×65 image patches. Since these deposits are diffuse, amorphous objects, pixel-wise classification is appropriate here and there is no attempt to define the number of deposits present.

In [8], an FCN is trained to classify histological images at a whole-image level. Although it is only trained with whole-image labels, it is still able to localise individual cells by deriving class probability maps from the final convolutional layers, in a manner inspired by [21,23]. FCNs are particularly useful when processing histological images due to their ability to naturally scale to images of arbitrary size, without needing to downsample large images to a fixed size. [11] train a standard CNN to classify the central pixel of image patches, then convert it to an FCN to perform pixel-wise classification over a whole image in one pass. This has performance benefits over processing patches one-by-one, since computations can be shared among overlapping image patches.

A standard CNN based on the design of Krizhevsky [9] is used to count human embryonic cells in [6]. Since the cells in these images show very high overlap, the act of counting is treated as a classification task and the cells themselves are not localised.

3 Method

When attempting to design a network that produces output of variable length, one immediately hits two technical limitations:

- Existing deep learning frameworks process data in "tensors", N-dimensional arrays whose shape is always a hyperrectangle. This includes the output tensor. Outputting a different number of boxes for each image in a batch would be like outputting a matrix with variable length rows, which is not supported.
- In order for the network to learn the correct number of boxes, this number needs to be somehow differentiable. That means the number of boxes produced must vary smoothly with respect to the network parameters; a small parameter change should result in a small improvement in the number of boxes.

These constraints can be satisfied by outputting a fixed number of boxes with confidence scores attached - as is the case in existing detection frameworks. The problem now is how to assign confidence scores such that each object gets exactly one high confidence box that matches its corresponding ground truth box.

3.1 Loss Function

To train a network to behave in such a way, a loss function is required that is minimised if and only if the network outputs exactly one matching box with high confidence for each ground-truth box. This is difficult, because the order in which the boxes are emitted should not matter. Loss functions in supervised learning work by penalising deviation from some target output, but if a network emits N output boxes per image and an image has M objects, then there are $\frac{N!}{(N-M)!}$ possible correct outputs, corresponding to different orderings of the boxes. Faster R-CNN and YOLO solve this problem by establishing "jurisdictions" for their output boxes, whereby the loss function demands that a box should have high confidence if a ground-truth box falls into its jurisdiction.

Ideally, we would like the loss function to be minimised no matter which detectors are used to label the objects, so long as there is only one each. To this end, we define a loss function that assigns responsibility for ground-truth boxes based on both the output box parameters (centre coordinates, width and height) and confidence scores. We define a responsibility matrix R, where R_{ij} is the responsibility of detector i for object j, and

$$R_{ij} = \frac{C_i}{D_{ij}} \tag{1}$$

$$D_{ij} = (x_i - x_j^*)^2 + (y_i - y_j^*)^2 + (w_i - w_j^*)^2 + (h_i - h_j^*)^2 \quad (2)$$

where x, y, w, h are centre coordinates and width and height, normalised to $[0, 1]$ relative to the image dimensions and mean box size, respectively, $*$ denotes ground-truth, and C_i is the confidence of detector i.

At training time, each ground-truth box j selects the detector that is most responsible for it:

$$R_j^* = \arg\max_i R_{ij} \quad (3)$$

This chosen detector incurs a regression loss $D_{R_j^*,j}$, causing it to better localise the object for which it was responsible. All detectors also incur regression loss on their confidence, where target confidence C_i^* is 1 if detector i is responsible for an object, and 0 otherwise. The total loss is then:

$$L = \frac{1}{N}\sum_i (C_i - C_i^*)^2 + \frac{1}{M}\sum_j D_{R_j^*,j} \quad (4)$$

where M and N are the number of ground-truth objects and detectors, respectively. This responsibility scheme is similar to that used by [15], but differs in that ours takes into account box confidence, allowing it to penalise over-detection if too many high confidence boxes are emitted.

Using detector confidence to establish responsibility allows the network to choose for itself which detector will be responsible. If detectors $1 - 5$ localise object j, then their regression losses D_{ij} for $i = 1..5$ will be similar, and so the highest responsibility will go to detector k with highest confidence C_k. This chosen detector will get a target confidence C_k^* of 1 while the others get 0. This reinforces detector k as the detector responsible for that object; next time the same object is seen, C_k will be higher, while others will be lower. This can be seen as a kind of learnt NMS.

We found that if confidence is not used to determine responsibility ($R_{ij} = \frac{1}{D_{ij}}$), the network outputs many boxes per object which all have roughly equal confidence well below 0.5. This is because the network cannot predict which box will be closest to the ground-truth since they are all close, and so cannot predict which should have confidence 1 and which should have 0. Moving all but one box away from the object would be a solution, but this would only produce discontinuous, non-differentiable changes in loss as the responsibility assignment changes suddenly, so the network cannot learn to do this.

3.2 Model Architecture

A recurrent neural network (RNN) would be the obvious choice to minimise the loss function described above. If bounding boxes are emitted sequentially rather than simultaneously, then each one can be dependent on the ones that came before it. In this way, a detector can avoid outputting a high confidence box on an object that has already been detected. Despite this attractiveness though, our best results out of the many architectures trialled came not from an RNN but from an FCN. This architecture is specified in Table 1.

Everything from `conv1` to `conv7` is a relatively standard convolution/maxpooling stack, with some slightly unusual features (stride of 2 in `conv6`) which allow the stack to output feature maps whose effective receptive fields in the image overlap by half (effective receptive field size is 64×64 pixels, effective stride is 32×32). This overlap ensures that every object lies fully within at least one neuron's receptive field. `boxes` emits bounding box parameters and `boxes_global` is a custom layer that performs a simple transformation from local coordinate space global image space. `concat` joins the feature maps of `boxes_global` and `conv7`, allowing the remaining three layers to predict confidence scores based on both the boxes themselves and the image features they were predicted from. We observed a modest improvement in performance due to this addition. The final three layers, then, can be seen as a learnt filtering stage that replaces the traditional NMS post-processing. A Theano/Lasagne implementation is available at https://github.com/philipjackson/avoiding-overdetection.

Table 1. A specification of our network architecture. Unless otherwise stated, each layer takes the previous layer's output as input. Nonlinearities are leaky rectified linear [13] with $\alpha = 0.1$ unless otherwise stated. B is a hyperparameter denoting the number of detectors per "window" (i.e. position in the final feature map, `conv7`). $B = 9$ in our experiments.

Network layers		
Name	Type	Parameters
`conv1`	Convolution	`num_filters=32, filter_size=(5,5)`
`pool1`	Maxpool	`pool_size=(2,2)`
`conv2`	Convolution	`num_filters=48, filter_size=(3,3)`
`pool2`	Maxpool	`pool_size=(2,2)`
`conv3`	Convolution	`num_filters=64, filter_size=(3,3)`
`pool3`	Maxpool	`pool_size=(2,2)`
`conv4`	Convolution	`num_filters=86, filter_size=(3,3)`
`pool4`	Maxpool	`pool_size=(2,2)`
`conv5`	Convolution	`num_filters=128, filter_size=(1,1)`
`conv6`	Convolution	`num_filters=128, filter_size=(2,2), stride=(2,2)`
`conv7`	Convolution	`num_filters=128, filter_size=(1,1)`
`boxes`	Convolution	`num_filters=4*B, filter_size=(1,1), nonlinearity=identity`
`boxes_global`	Coord Transform	
`concat`	Concatenation	`inputs=boxes_global,conv7`
`filter1`	Convolution	`num_filters=16*B, filter_size=(3,3)`
`filter2`	Convolution	`num_filters=16*B, filter_size=(1,1)`
`confidence`	Convolution	`num_filters=B, filter_size=(1,1), nonlinearity=sigmoid`

4 Results

We trained our model on a set of 17000 SIMCEP images using the Adam optimizer [7], and validated against a set of 3000. The images were of size 224×224

pixels and contained anywhere from 1 to 15 cells. The parameters of SIMCEP were adjusted to randomise obfuscating features such as blur, Gaussian noise and uneven lighting, and the cells show varying levels of clustering and overlap.

A selection of results is shown in Fig. 2. We interpret any detection with a confidence above 0.5 as a positive, and so the number of such detections is the network's estimate of the number of cells present. Across our validation set, the root mean square of the deviation of this estimate from the true count was 2.28. Further quantitative results are shown in Table 2.

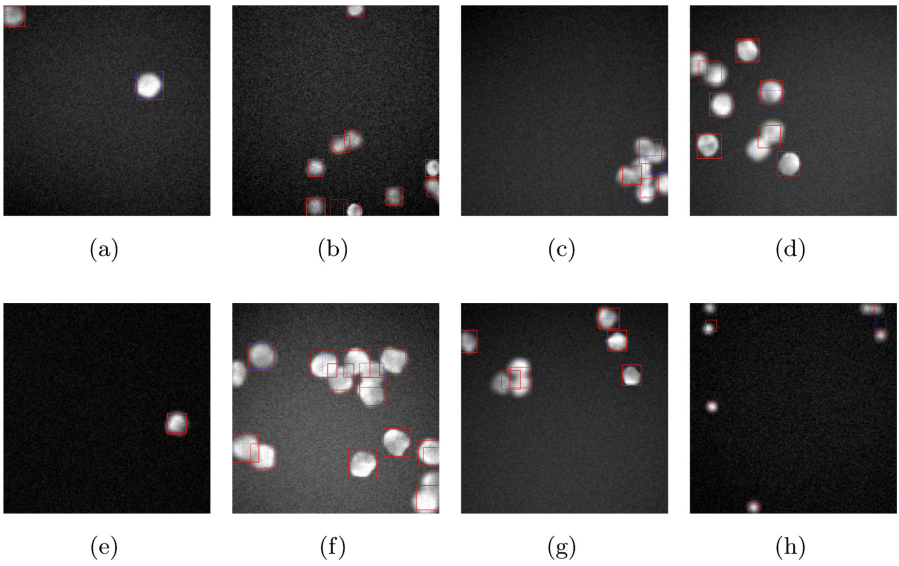


Fig. 2. A sample of detection results on SIMCEP images. Confidence is represented in the transparency of the boxes; all output boxes with confidence above 0.1 are shown. Instead of post-processing with NMS, we simply take boxes with confidence above 0.5 (shown in red) as positive detections. Boxes with confidence below 0.5 are shown in blue. (Color figure online)

Table 2. True and false positive rates on training and validation sets. A true positive is counted as any output box with an intersection over union (IoU) above 60% with a ground-truth box, but each ground-truth box can only be paired with a single output box. So if two output boxes cover the same object, then this counts as one true positive and one false positive. Output boxes with less than 60% IoU with any ground-truth box are always false positives.

	True positive rate	False positive rate	F_1-score
Training set	75.4%	19.2%	0.774
Validation set	75.3%	19.4%	0.773

5 Conclusion

For images containing objects whose bounding boxes overlap heavily due to occlusion or dense clustering, NMS cannot reliably remove excess bounding boxes emitted by the network, since ground-truth bounding boxes with identical classes may truly overlap significantly. An end-to-end system that outputs the correct number of boxes without the need for post-processing NMS is therefore preferable. In this paper, we discuss the problem and take some early steps towards solving it, demonstrating a system that can localise densely clustered objects and simultaneously approximate the correct number of boxes. Rather than performing regression directly on the number of objects, we encode this number implicitly in the number of high confidence boxes emitted by the network.

We propose that, unless an alternative output encoding can be found which shows a one-to-one mapping between output values and unordered sets of boxes, supervised learning itself is unsuitable for this task. There are many ways for a network to output the same set of boxes, depending on which box it places on which object (or for an RNN, which order it outputs them in), but supervised learning requires us to arbitrarily choose one of them, and penalise all the others. In this paper we partially solve this problem by choosing which box should have high confidence based partly on the confidence values themselves, however the results are far from perfect. We put forward three reasons for this, and suggest how they may be countered by applying reinforcement learning instead of supervised learning.

Firstly, because we assign responsibility for an object to the detector with the maximum responsibility for it, our loss function is discontinuous, due to the $\arg\max$ operation. This is likely to cause problems for supervised learning, which is based on direct optimization of the loss function by gradient descent. Reinforcement learning trains a network to optimize a reward function which may be related to the network's output in a complex, non-differentiable or even unknown way. In particular, reward functions do not prescribe a target output for every input, and so they completely bypass the problem of choosing which detector should label which object. This makes reward functions a much more natural way to express the goal of one box per object.

The second reason is that in order to teach a network to output the right number of boxes, that number must somehow be made smooth and differentiable, despite the fact that we ultimately want an integral number. To derive this hard number, we currently threshold the confidence levels at 0.5. Not only is this threshold somewhat arbitrary, but worse still, it is effectively a post-processing step that the network itself is unaware of, and indeed cannot be trained to optimize because it is non-differentiable. This too can be solved with reinforcement learning by building the thresholding step into the reward function, because discontinuous reward functions are alllowed.

The third reason is that our FCN architecture outputs all the boxes in parallel. This means that each detector is unaware of what the others are doing, so it is difficult for them to coordinate themselves so as to avoid over-detection. Using an RNN that outputs boxes in series would solve this problem, as the

output on one time step can be conditioned on that of previous ones. This also fits well with reinfrocement learning, since a reward signal can be administered on every time step; this would accelerate training compared to a single overall reward signal per image.

References

1. Ciresan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in Neural Information Processing Systems, vol. 25, pp. 2843–2851. Curran Associates, Inc. (2012)
2. Dong, B., Shao, L., Da Costa, M., Bandmann, O., Frangi, A.F.: Deep learning for automatic cell detection in wide-field microscopy zebrafish images. In: IEEE International Symposium on Biomedical Imaging, pp. 772–776 (2015)
3. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge 2012 (VOC2012) results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
4. Girshick, R.: Fast R-CNN. In: IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
5. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
6. Khan, A., Gould, S., Salzmann, M.: Deep convolutional neural networks for human embryonic cell counting. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9913, pp. 339–348. Springer, Cham (2016). doi:10.1007/978-3-319-46604-0_25
7. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
8. Kraus, O.Z., Ba, J.L., Frey, B.J.: Classifying and segmenting microscopy images with deep multiple instance learning. Bioinformatics **32**(12), 52–59 (2016)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1–9 (2012)
10. Lehmussola, A., Ruusuvuori, P., Selinummi, J., Huttunen, H., Yli-Harja, O.: Computational framework for simulating fluorescence microscope images with cell populations. IEEE Trans. Med. Imag. **26**(7), 1010–1016 (2007)
11. Litjens, G., Sánchez, C.I., Timofeeva, N., Hermsen, M., Nagtegaal, I., Kovacs, I., Hulsbergen-van de Kaa, C., Bult, P., van Ginneken, B., van der Laak, J.: Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. Sci. Rep. 6 (2016)
12. Ljosa, V., Sokolnicki, K.L., Carpenter, A.E.: Annotated high-throughput microscopy image sets for validation. Nat. Meth. **9**(7), 637 (2012)
13. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: International Conference on Machine Learning, vol. 30 (2013)
14. Prentai, P., Lonari, S.: Detection of exudates in fundus photographs using convolutional neural networks. In: International Symposium on Image and Signal Processing and Analysis, pp. 188–192 (2015)
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Computing Research Repository (2015)

16. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
17. Rothe, R., Guillaumin, M., Gool, L.: Non-maximum suppression for object detection by passing messages between windows. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9003, pp. 290–306. Springer, Cham (2015). doi:10.1007/978-3-319-16865-4_19
18. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge, vol. 1409 (2014)
19. Ruusuvuori, P., Manninen, T., Huttunen, H.: Image segmentation using sparse logistic regression with spatial prior. In: 2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO), pp. 2253–2257, August 2012
20. Seguí, S., Pujol, O., Vitria, J.: Learning to count with deep object features. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 90–96 (2015)
21. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps (2013)
22. Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. Int. J. Comput. Vis. **104**(2), 154–171 (2013)
23. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). doi:10.1007/978-3-319-10590-1_53