Intelligent Detection Using Convolutional Neural Network (ID-CNN)

Intelligent Detection Using Convolutional Neural Network (ID-CNN)

Amish Jahangir Kapoor, Hong Fan and Muhammad Sohail Sardar

Donghua University, College of Information Science and Technology
201600 Shanghai, China
amishkapoor25@yahoo.com

Abstract. In this paper, we will study the basic flow and the principle of state-of-the-art object detection technique (i.e. Faster R-CNN) and improve it further with the inclusion of two strategies into it. Firstly, we propose a multi-layer features merging strategy by using a concatenation layer. Secondly, we introduce a contextual learning scheme for Faster R-CNN. Previously, Faster R-CNN just uses regional features. Contextual features are added with the regional features for the classification and detection task. Our improvement on Faster R-CNN shows promising results. We call our improved Faster R-CNN network as ID-CNN (Intelligent Detection Using Convolutional Neural Network) as its detection accuracy is better. Therefore, we call it as an intelligent detector. We use a deep VGG-16 model as our base model, as Faster R-CNN did. We evaluated our ID-CNN on Pascal VOC public datasets. Experimental results show that ID-CNN can effectively improve the object detection average precision to some extent. On VOC 2007 and 2012, we achieved a mean average precision (mAP) of 74.7% and 71.9%, respectively. ID-CNN is also end-to-end trainable with the same alternating fine-tuning optimization scheme of Faster R-CNN. Finally, we compared ID-CNN with Faster R-CNN on ImageNet object detection dataset and we achieved mAP of 48.1% compared with 46.2% for Faster R-CNN.

1. Introduction

Leveraging GPUs, the research on CNNs (like alexnet [12]) has been emerged swiftly and achieved state of- the-art results on image classification task. State-of-the-art object detection algorithms depends on the region proposal based methods [1],[2],[3],[4] and on the methods like SSD [8] and YOLO [5], that are feedforward in a single pass. SSD and YOLO, both deals detection as a regression problem but have less accuracy compared with the Faster R-CNN [3]. Region-based CNN [1] is great to hypothesize the object's location. Works on SPPnet [7] and Fast R-CNN [2], achieves nearly real-time rates when ignoring the time spent on region proposals. But the bottleneck of external region proposals was there, like selective search (SS) [4] which greedily merges super-pixels based on low-level features. So, it is computationally expensive and also is of slower magnitude.

Edge boxes [6] provides a better tradeoff between proposal's quality and the speed (at ~0.2s per image). However, still the region proposal step consumes as much running time as the detection network itself. Therefore, RPN (region proposal network) based network is combined with Fast R-CNN [2] to form a Faster R-CNN [3]. Faster R-CNN makes the whole object detection pipeline an end-to-end process. With a simple alternating optimization algorithm, RPN [3] and the Fast R-CNN [2] can be trained end-to-end to share convolutional features while exempting the use of external region proposals [4]. RPN is constructed by adding additional two layers on top of a Fast R-CNN [2] detection network. First layer converts each convolutional map position into a short feature vector (e.g. 512-d for VGG-16 in [3] and 640-d in our case). Second layer outputs an object-ness score and

regressed bounding-box offsets at each convolutional map position for k region proposals corresponding to various scales and aspect ratios at that location (k=9 as used in [3]). Faster R-CNN achieved great results in object detection pipeline with mAP of 69.9% than the great baseline of (SS) [4] with Fast R-CNN [2] with mAP of 66.9% on VOC 2007 dataset. However, space for improvement is still there in average precision.

In this paper, we show that with the slight modifications in the network of Faster R-CNN [3], an elegant and an effective result could be achieved. Our observations found that the low and the high-level features of the convolutional layers of a CNN are complementary to each other in a way as high-level features having semantic features are actually formed on low-level features. Also, the capability of localization in low-level features and the semantic information in the high-level features can be fused together [25]. Keeping this in mind, we proposed a multi-layer features merging strategy. We down-sample the low-level features to the same size of the high-level features by using convolution with a bottleneck structure. After that, we merge them together with the help of a concatenation layer. This lets for a more effective feature representation of the original image. Knowing the fact, that the Faster R-CNN [3] only uses regional features to classify and locate an object, we introduced a context-based learning structure to it. The contextual features are extracted from the image's convolution feature maps by using the context pooling layer, and then we merge it with the regional features for the classification and localization task of an image.

With the inclusion of multi-layer features merger and the contextual learning strategy, the accuracy for the detection and classification of RPN-based network with Fast R-CNN architecture has been improved. We evaluated our method on Pascal VOC 2007 [15] and VOC 2012 datasets, which shows an impressive mAP of 74.7% and 71.9%, respectively. Besides Pascal, we compared ID-CNN with Faster R-CNN on the ImageNet object detection dataset (ILSVRC) [24] and achieved a mAP of 48.1% compared with mAP of 46.2% for Faster R-CNN.

2. Literature Review

Outdated methods for object detection involved using a block-wise orientation of histogram (e.g. HOG [11] or SIFT [14]) features which could not attain high accuracy in standard datasets (e.g. PASCAL VOC 2007). A couple decades ago, CNN was implemented when LeNet architecture [16] was used for recognizing digits. That was the first successful CNN application that was used in practice. Then, a hibernation period was there for CNN as the computers were not fast yet. CNNs were rejuvenated with the work of Alexnet [12] for ILSVRC challenge to classify objects, and that paved the way for deeper networks for the classification tasks. By this time, GPUs were around.

Overfeat [10] (a class specific regression) method uses sliding window approach to classify and localize the object. In Overfeat, a fully-connected layer is trained to predict the box coordinates for the localization task which presumes a single object. Then FC layer turns into a convolution layer for detecting multiple class-specific objects. For R-CNN [2] object detection, Multi-box methods [9],[17] generates region proposals from a network whose last fully connected (FC) layer simultaneously predicts multiple number of boxes for being an object.

The goal of generating region proposals is to create a relatively small set of bounding boxes for the candidate to cover all objects in an image. These proposals are most commonly used with the complex and expensive classifiers to allow efficient object detection [1],[2],[4]. Faster R-CNN [3] is a two-staged cascade architecture, consisting of class-agnostic proposals (RPN) and a class-specific detector (Fast R-CNN). RPNs are like fully connected network that they can be trained end-to-end for the task of generating proposals. To combine RPN with fast R-CNN [2], an alternating fine-tuning optimization scheme is used. This fine-tuning scheme converges quickly and produces a unified network where RPN and Fast R-CNN share same convolution features.

Table 1. Different object detection architectures. '+' denotes employed while '-' denotes not considered. Note that, R-CNN [1] and SPPnet [7] are not end-end trainable with a multi-task loss whereas other architectures are based on multi-task joint training.

Framework	Proposal	Multi-scale	Learning Method	End-to-End	Loss Function
R-CNN	Selective Search	-	SGD, BP	-	Hinge Loss, B. box reg.
SPPnet	Edge Boxes	+	SGD	-	Hinge Loss, B. box reg.
Fast R-CNN	Selective Search	+	SGD	-	Log Loss, B. box reg.
Faster R-CNN	RPN	+	SGD	+	Log Loss, B. box reg.
SSD	-	-	SGD	+	Softmax loss + B. box reg.
Yolo	-	-	SGD	+	Sum-Squared error loss + B. box reg. + Obj. Conf. + Background Conf.

RPNs have so-called translation invariant anchors unlike the Multi-box method's [17] anchors. Anchors of RPN uses 3 scales for box area of 128^2, 256^2, 512^2 pixels with 3 aspect ratios of 1:1, 1:2, 2:1. Therefore, with 3 scales and 3 aspect ratios the anchors at each sliding window position on the feature map yields k=9. On the other hand, Multi-box [17] uses k-means to generate 800 anchors, which are not translation invariant. It requires a (4+1)×800 dimensional output layer, whereas RPN-based method requires a (4+2)×9 dimensional output layer. RPN-based proposal layers have fewer parameters compared with Multi-box, and thus have less risk of overfitting on small datasets comparatively like PASCAL VOC [15]. Anchors with an IoU (Intersection over Union) greater than 0.7 and the highest IoU overlap with a ground-truth box considered as positive anchors and whereas IOU less than 0.3 considered to be as negative anchors. 50% ratio is being used for positive and negative anchors in a mini-batch. For training RPN, a binary class label of being an object or not is assigned to each anchor.

3. Proposed Method

In order to improve the performance of Faster R-CNN, the proposed network ID-CNN consists of multi-layer features merging scheme and context-based learning scheme. We will take VGG-16 as our base model as [3] did and improves the feature extraction of it, as it plays a vital role for the classification and detection tasks. For an easier example, we take 224×224 as an input image to describe our proposed method. The two efficient inclusions to [3] are:

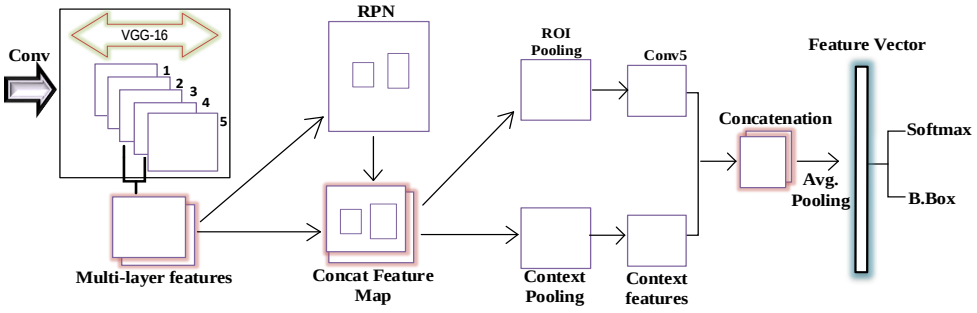


Figure 1. Intelligent Detection using Convolutional Neural Network (*ID-CNN*). The overall improved version of Faster R-CNN.

3.1. Multi-Layer Features Merging

With observations, we found that low and high-level features of a CNN are complementary to each

other. As high-level features having complex and semantic features are actually formed on low-level features that contains basic information like edges, curves, lines and texture etc. In addition, the capability of localization in low-level features and the semantic information in the high-level features can be fused together [25] as we also need to localize the object in the image as well. Therefore, multi-level merging strategy bridges the gap between high and low-level features. In this scheme, we merge low-level features with the high-level features by concatenation. As from papers [18] and [19], they use deconvolution to visualize the features of different layers in a network. As a reference from [19], it can be seen the role played by the features from different layers of a CNN. The shallow layers keeps most details of the input image, also small-structured objects barely have any response in the deeper layers. Multiple layers fusion is a great way to boost the performance of detection as CMS-RCNN [20] did.

High-level feature maps loses many important low-level features while undergoing through several down-samplings. By this, feature maps becomes more abstract to the input image as CNN always shrinks its feature maps by using max pooling operation after convolution layer in order to reduce the number of parameters in the network. Which means, one pixel in the feature map corresponds to several pixels in the input image. Therefore, the output feature maps from different convolution layers have different sizes. Due to a down-sampling, the size of the feature maps of two adjacent convolution layers are generally twice the same. In order to merge these feature maps, it is necessary to adjust the output feature maps to the same size of each other. That is to say, the large feature maps needs to be down-sampled to the same size of the smaller ones. We down-sample the output of conv3_3 and merge it with the output of conv4_3 by using concatenation to compare their features.

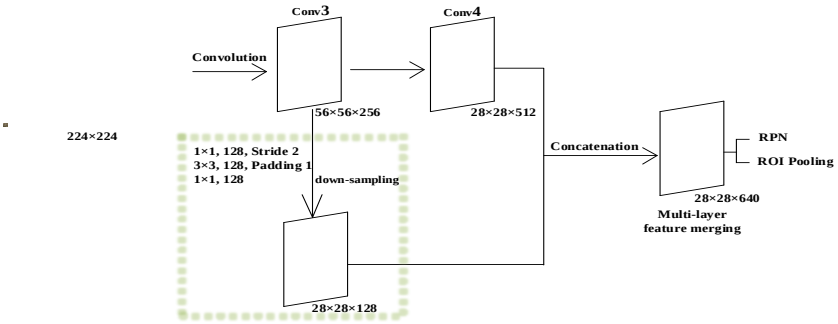


Figure 2. Multi-Layer Features Merging to merge Conv3_3 and Conv4_3 via a Bottleneck structure

A convolution layer with a stride of 2 is being used to down-sample the conv3_3 (i.e. from 56×56×256 to 28×28×128) with carefully chosen 128 filters. On the other hand, conv4_3 is 28×28×512 as shown in Fig. 2. If not maintained the dimension, the output of conv3_3 would increase the feature depth of the concatenation layer and hence the parameters as well after merging with conv4_3. Therefore, to improve the expression ability of the merged features and to reduce the feature depth (dimension) we would use a “Bottleneck structure” like the one used in GoogleNet [21]. As from [21] we come to know, the use of convolution involves a lot of parameters which makes the network computationally expensive. On the other hand, the problem becomes more pronounce when pooling operation is added as it preserves the feature depth (dimension) which means the total feature depth after concatenation can only grow. Hence, a bottleneck structure can be applied before the convolution operation to reduce the parameters and the 1×1 convolution in bottleneck also helps to reduce the feature depth. The third layer of Bottleneck does not double the number of convolution filters. Hence, the number of newly introduced parameters can be maintained at a lower level. Also, merged features’ expression ability would be improved and the number of feature maps of low and high-level would be of ratio (1:4). Since, the high-level features still plays a major role.

After that, we insert RPN between concatenation and the conv5 of the model to generate regions of interest for the input image. The RPN proposes 200-300 proposals based on the ignorance of all cross-

boundary anchors and applying NMS on the proposal regions based on their class score. For a typical input image of size e.g. (224×224), the output feature map can be obtained after the first four convolutional blocks of a deeper network like VGG16 to generate regions of interest from RPN. The candidate regions are then projected onto the concatenation layer's output feature map in a certain proportion. Since, VGG16 has gone under three down-sampling operations from conv1 to conv4, the feature map is reduced by 1/8 times and hence the projection ratio is also $1/8^{th}$ of the original image. The projected regions are then pass through ROI Pooling layer to become of certain size. Here, conv5 of VGG16 model is used to extract these local features of the candidate regions from ROI pooling. The size of which is 7×7 whereas the input of 5^{th} convolution layer is 14×14. This is because, the 5^{th} convolution layer begins with conv5_1 and then after pooling with a stride of 2 it would be down-sampled to give 7×7 at the output.

3.2. Context-Based Learning

In order to improve the accuracy for classification and localization of our network, then we want our network to know the surroundings within an image better. For this, context information of the whole image is required. The global context can be useful when making predictions for local image regions. The context can be regarded as a background information of the whole image. At present Faster R-CNN only utilizes the features from the local regions of interest rather than the context information of the whole image. In our network contextual learning is possible. The features we get from multi-layer features merging scheme can paved way for contextual information of the image. The context of concatenation layer's output feature map is pooled in the same way as that of ROI pooling is done. The only difference is, that the context is pooled with the feature map of the entire image as the regions of interest. Here, we divide it by 14×14 grids and pool the maximum value in each grid to get a fixed size of 14×14.

In the context environment, again we use a bottleneck structure to get better expression ability for the output features and to make them to be in a certain proportion (e.g. 1/4), as shown in Fig. 3. With 128 filters to maintain its proportion the convolutions of 1×1 are applied with stride of 2 whereas, 3×3 with a pad of 1 and then the final 1×1 to extract the context features while reducing its size to 7×7. The feature maps extracted by the convolution operation can be regarded as the context features. Then, we merge these context features with the regional features (conv5) by concatenation to give our network the contextual information.

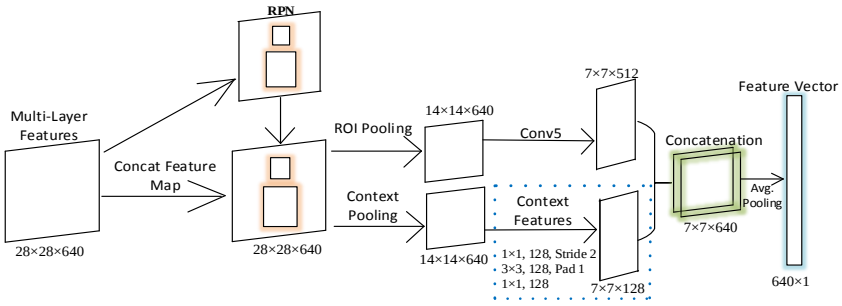


Figure 3. A comprehensive example-based diagram to let you know the insight workflow of Context-Based Learning Structure

Finally, the concatenation result is average pooled out with a 7×7 pool. This global average pooling (GAP) [23] layer helps to minimize the overfitting by reducing the total number of parameters in our model. It reduces the dimension to 1×1×640. That means, it generates one feature map for each corresponding category of the classification task in the last layer. It allows us to have the input image of any size. After-words, it mapped to a feature vector by fully connected layers. The network has two output vectors: softmax probabilities for classification with object-ness score and per class based

bounding-box regression offsets for the detection of box coordinates.

4. Loss Functions

An objective function is to be minimized and the RPN loss from [3] for an image is as follows:

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

Here, i is the index for an anchor in a mini-batch, p_i is the predicted probability of anchor i being an object whereas, p_i^* is ground-truth label (1 for positive anchor, 0 for negative). t_i and t_i^* represents predicted and ground-truth boxes' 4 parameterized coordinates for anchor i. L_{cls} is a classification's log loss (Softmax) over two classes (object or not object) and $L_{reg}(t_i, t_i^*)$ is a (smooth L1) regression loss from [2]. Both terms are normalized with N_{cls} and N_{reg} whereas λ is a balancing weight. After minimizing RPN loss, following multi-task loss in Fast R-CNN [2] is to be minimized:

$$L(p,u,t^u,v) = L_{cls}(p,u) + \lambda[u \geq 1] L_{loc}(t^u, v) \quad (2)$$

Here, p and u are predicted class and true class scores, respectively. Whereas, t^u is true box coordinates and v is predicted box coordinates. The first term (L_{cls}) represents a log loss (softmax) for classification and the second term (L_{loc}) represents a regression loss (smooth L1). The term [u ≥ 1] evaluates to 1 when u ≥ 1 and 0 otherwise. The hyper parameter λ controls the balance between the two task losses of Fast R-CNN. For bounding-box regression, the parameterization of the four coordinates are taken from R-CNN [1].

$$\begin{aligned} t_x &= \frac{(x - x_a)}{\omega_a}, \quad t_y = \frac{(y - y_a)}{h_a}, \quad t_w = \log\left(\frac{\omega}{\omega_a}\right), \quad t_h = \log\left(\frac{h}{h_a}\right) \\ t_x^* &= \frac{(x^* - x_a)}{\omega_a}, t_y^* = \frac{(y^* - y_a)}{h_a}, t_w^* = \log\left(\frac{\omega^*}{\omega_a}\right), t_h^* = \log\left(\frac{h^*}{h_a}\right) \end{aligned} \quad (3)$$

Here, x and y are the center coordinates of the box. Whereas, w and h are width and height, respectively. Variables x, x_a and x^* are for the predicted, anchor and ground-truth boxes, respectively (likewise for y, w and h). It is like a bounding-box regression from an anchor box to a nearby ground-truth box.

The overall multi-task loss for this network is the sum of RPN and Fast R-CNN loss. RPN and Fast R-CNN needs to be trained jointly with four losses (two each for RPN's proposals and for Fast R-CNN's detection). An algorithm is introduce in [3] that allows the sharing of convolution layers between the RPN and Fast R-CNN. From [3], we used a four step training algorithm to learn for shared convolution features via alternating fine-tuning optimization scheme. We refer readers to [3] for more details about the alternating fine-tuning optimization algorithm.

5. Optimization of Parameters

RPN, that was basically implemented as a fully convolutional network in [22] can be trained end-to-end by back-propagation and stochastic gradient descent (SGD). All new layers are randomly initialized by drawing weights from a zero-mean Gaussian distribution of $N(0,0.01^2)$. All other layers (i.e. the shared convolution layers) are initialized by pre-training a model for ImageNet classification [14]. We use a learning rate of 0.001 for first 50k mini-batches and then reduced it to 0.0001 for the next 20k mini-batches on the dataset. Instead, of 60k and 20k as used in the literature of [3]. Weight decay of 0.0005 and a momentum of 0.9 is being used with caffe framework for the implementation.

Our method also provides an end-to-end learning environment like in [3] even after the inclusion of both multi-layer features merging and the contextual learning strategies into it. For the fine-tuning of parameters, the parameters of conv1 and conv2 of VGG16 are fixed. Whereas, the parameters from conv3 to conv5 and the FC layers are fine-tuned. It is feasible to fix the parameters of first few layers and adjust only the latter ones when the network is large like VGG16. As the characteristics features of first few layers have certain generality and they tend to change less.

6. Experiments and Results

We evaluated our ID-CNN on PASCAL VOC 2007 [15], VOC 2012 and on ImageNet ILSVRC [24] dataset. PASCAL dataset consists of (~5k trainval images) and (~5k test images) for over 20 object categories. For the ImageNet pre-trained network, we use the VGG-16 model [13] that has 13 convolution layers and 3 FC layers. The purpose of using the same model as used by [3] is to make a fair comparison. Faster R-CNN is our baseline.

Table 2 shows the effects of Multi-layer features merging strategy on object detection results using VOC 2007 test set and 07+12 (union set of VOC 2007 trainval and VOC 2012 trainval) as training set. In 1st experiment, we use Faster R-CNN for training and testing. In 2nd, we use max pooling layer (M_P) for the down-sampling of conv3_3 and then merged its output feature maps with the output feature maps of conv4_3. In third experiment, convolution layer (M_C) is use for the down-sampling. And in our fourth experiment, convolution layer (M_C) with a bottleneck structure (M_B) is use for down-sampling. The result of 2nd experiment that uses max pooling is lower than that of 1st. Contrary, the results of experiments 3rd and 4th are fairly increased by 0.5% and 0.9%, respectively than that from baseline.

Table 2. Detection results with different Multi-layer features Merging Strategies

Experiment No.	Network	Model	Training Set	Test Set	mAP (%)
1	Baseline	VGG-16	07+12	07	73.2
2	Baseline + M_P	VGG-16	07+12	07	73.0
3	Baseline + M_C	VGG-16	07+12	07	73.7
4	Baseline + M_C + M_B	VGG-16	07+12	07	74.1

Table 3 shows the effects of context-based learning strategy on object detection results. Here again, we will try different ways to down-sample our context features. So that, in a better way features' expression ability would be possible. First experiment uses baseline network for training and testing. Second experiment uses max pooling layer (C_P) for the down-sampling of context features. As happened before, with the use of pooling operation our detection accuracy sinks. In third experiment, convolution layer (C_C) is used for down-sampling and a progress can be seen of 0.4%. In fourth one, convolution layer (C_C) with a bottleneck structure (C_B) is used for down-sampling and a clear 0.6% improvement is made here by utilizing the contextual image features.

Table 3. Detection results with different Context-Based learning strategies

Experiment No.	Network	Model	Training Set	Test Set	mAP (%)
1	Baseline	VGG-16	07+12	07	73.2
2	Baseline + C_P	VGG-16	07+12	07	72.9
3	Baseline + C_C	VGG-16	07+12	07	73.6
4	Baseline + C_C + C_B	VGG-16	07+12	07	73.8

In order to test the overall improved performance, we add this contextual learning strategy with our multi-layer features merging strategy. The experimental result is shown in the Table 4. It gives us the final result of our ID-CNN network that includes both strategies into Faster R-CNN and gives us an impressive 1.5% of overall improvement.

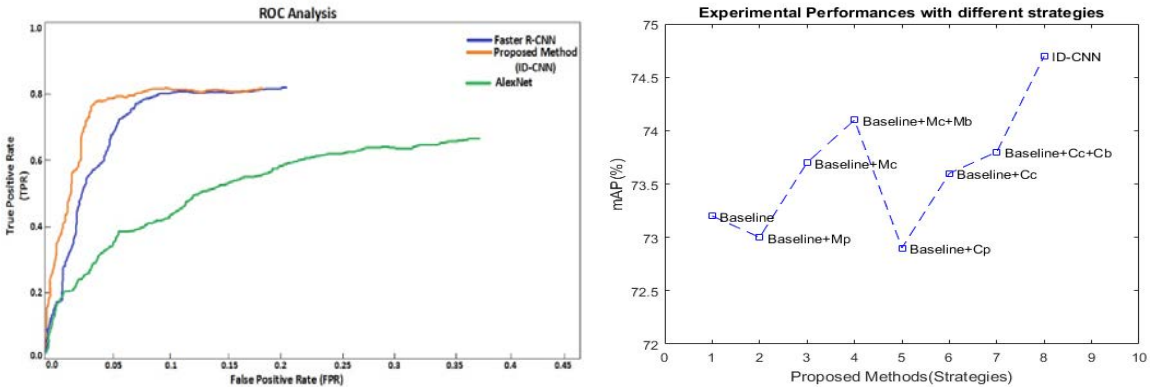
Table 4. Detection results on Pascal VOC 2007 test set

Network	Model	Training Set	Test Set	mAP (%)
ID-CNN	VGG-16	07+12	07	74.7

Table 5 shows the overall performance of our ID-CNN network compared with the state-of-the-art object detection network [3]. Detection results are on VOC 2012 test set and the networks are trained on 07++12 (union set of VOC 2007 trainval + test and VOC 2012 trainval).

Table 5. Detection results on Pascal VOC 2012 test set

Network	Model	Training Set	Test Set	mAP (%)
Baseline	VGG-16	07++12	12	70.4
ID-CNN	VGG-16	07++12	12	71.9



(a) (b)

Figure 4. (a): A ROC-based analysis between Faster R-CNN with our ID-CNN and clearly it performs better than its predecessors. **(b):** Graphical representation of our experiments with different proposed schemes as mentioned in Tables from 2-4, respectively.

Besides PASCAL datasets, we compared ID-CNN with Faster R-CNN on ImageNet object detection dataset [24]. It has 1000 categories with over 1.28 million training, 100,000 test and 50,000 validation images. We divide the validation set into two parts (val1 and val2). For training, we use the training set plus val1. Whereas, val2 for testing.

Table 6: Object detection results on ILSVRC

Method	Model	Training Set	Test Set	mAP (%)
Faster R-CNN	VGG-16	Train+val1	val2	46.2
OUR1	VGG-16	Train+val1	val2	47.3
OUR2	VGG-16	Train+val1	val2	48.1

“OUR1” represents the inclusion of multi-layer features merging strategy. Whereas, “OUR2” represents the inclusion of both multi-layer features merging and the contextual learning. . The average detection accuracy of VGG-16 with multi-layer strategy is improved by 1.1%. Then, we add the contextual learning strategy to the experiment of “OUR1” and it add-ons 0.8% additional

improvement. The overall mAP is improved by 1.9%.

7. Conclusion

In this work, we have presented "Intelligent detection using CNN" (ID-CNN) for high-quality object detection that is simple, efficient and practical to implement. The proposed framework experimentally justifies that multi-layer features merging via a bottleneck structure and the use of contextual learning (also via a bottleneck structure) improves the overall detection accuracy. ID-CNN prevents from losing any important features and also keeps alive the small-structured object's features by taking them out from shallow layers and merge them with the deeper ones.

Improvements in context modeling and the inclusion of intermediate level features merging resulted in modest, but significant cumulative gains on classification and localization tasks. ID-CNN also enables an end-to-end deep learning based object detection framework with the same alternating fine-tuning optimization scheme. Finally, its implementation on further deeper network (like Resnet) may improve it further.

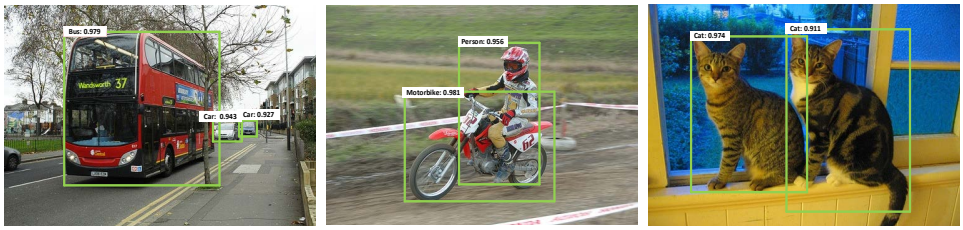


Figure 5. Detection examples using "ID-CNN" on PASCAL VOC 2007. An IoU threshold of 0.7 was used for correctness.

8. References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik.: Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 580--587 (2014)
[2] Girshick R.: Fast R-CNN[C]. IEEE International Conference on Computer Vision, Santiago. 1440--1448 (2015)
[3] Ren S, He K, Girshick R, et al.: Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In NIPS, 91--99 (2015)
[4] Uijlings J R R, van de Sande K E A, Gevers T, et al.: Selective Search for Object Recognition. International Journal of Computer Vision, 154--171 (2013)
[5] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi.: You Only Look Once: Unified, Real-Time Object Detection. In IEEE Conference CVPR, (2016)
[6] C. L. Zitnick and P. Doll´ar.: Edge boxes: Locating object proposals from edges. In ECCV, 391--405 (2014)
[7] K. He, X. Zhang, S. Ren, and J. Sun.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 1904--1916 (2014)
[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan.: SSD: Single Shot MultiBox Detector. In ECCV, (2014)
[9] D. Erhan, C. Szegedy and A. Toshev.: Scalable object detection using deep neural networks. In CVPR, (2014)
[10] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, (2014)
[11] Navneet Dalal, Bill Triggs.: Histograms of Oriented Gradients for human detection. In CVPR, (2005)
[12] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. hinton.: ImageNet classification with Deep Convolutional Neural Networks. In NIPS, (2012)
[13] K. Simonyan and A. Zisserman.: Very deep convolutional networks for large-scale image recognition. In ICLR, 1409--1556 (2015)

[14] David G. Lowe.: Object Recognition from Local Scale-Invariant Features. In ICCV, (1999)
[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. In ICJV, 303--318 (2010)
[16] Y. Lecun, L. Bottou, Y. Bengio.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 2278--2324 (1998)
[17] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov.: Scalable, high-quality object detection. In CVPR, (2015)
[18] M. D. Zeiler and R. Fergus.: Visualizing and understanding convolutional neural networks. In ECCV, (2014)
[19] Jason Yosinski, Jeff Clune, Anh Nguyen.: Understanding Neural Networks through Deep Visualization. International Conference on Machine Learning, (2015)
[20] Chenchen Zhu, Yutong zheng, Khoa Luu.: CMS-RCNN: Contextual Multi-Scale Region-based CNN for Unconstrained Face Detection. Deep Learning for Biometrics, 57--79 (2017)
[21] Christian Szegedy, Wei Liu, yangqing Jia.: Going Deeper with Convolution. In CVPR, (2015)
[22] J. Long, E. Shelhamer, and T. Darrell.: Fully Convolutional networks for semantic segmentation. In CVPR, (2015)
[23] Min Lin, Qiang Chen, Shuicheng Yan.: Network In Network. In ICLR, (2014)
[24] Russakovsky O, Deng J, Su H, et al.: Imagenet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, vol. 115(3), 211--252 (2015)
[25] B. Hariharan, P. Arbel´aez, R. Girshick, and J. Malik.: Hypercolumns for object segmentation and fine-grained localization. In CVPR, 447--456 (2015)