DeepID-Net: Object Detection with Deformable Part Based Convolutional Neural Networks

Wanli Ouyang*, Member, IEEE, Xingyu Zeng*, Student Member, IEEE, Xiaogang Wang, Member, IEEE, Shi Qiu, Member, IEEE, Ping Luo, Member, IEEE, Yonglong Tian, Student Member, IEEE, Hongsheng Li, Member, IEEE, Shuo Yang, Student Member, IEEE, Zhe Wang, Student Member, IEEE, Hongyang Li, Kun Wang, Junjie Yan, Chen-Change Loy, Member, IEEE, Xiaoou Tang, Fellow, IEEE

Abstract—In this paper, we propose deformable deep convolutional neural networks for generic object detection. This new deep learning object detection framework has innovations in multiple aspects. In the proposed new deep architecture, a new deformation constrained pooling (def-pooling) layer models the deformation of object parts with geometric constraint and penalty. A new pre-training strategy is proposed to learn feature representations more suitable for the object detection task and with good generalization capability. By changing the net structures, training strategies, adding and removing some key components in the detection pipeline, a set of models with large diversity are obtained, which significantly improves the effectiveness of model averaging. The proposed approach improves the mean averaged precision obtained by RCNN [18], which was the state-of-the-art, from 51% to 50.3% on the ILSVRC2014 detection test set. It also outperforms the winner of ILSVRC2014, GoogLeNet, by 6.1%. Detailed component-wise analysis is also provided through extensive experimental evaluation, which provides a global view for people to understand the deep learning object detection pipeline.

Index Terms—CNN, convolutional neural networks, object detection, deep learning, deep model

1 INTRODUCTION

Object detection is one of the fundamental challenges in computer vision and has attracted a great deal of research interest [11], [50]. Intra-class variation in appearance and deformation are among the main challenges of this task.

Because of its power in learning features, the convolutional neural network (CNN) is being widely used in recent large-scale detection and recognition systems [60], [54], [21], [27], [82], [81]. Since training deep models is a non-convex optimization problem with millions of parameters, the choice of a good initial point is a crucial but unsolved problem.

Of a good initial point is a crucial but unsolved problem, the choice...
The performance of deep learning systems depends significantly on implementation details [4]. However, an evaluation of the performance of the recent deep architectures on the common ground for large-scale object detection is missing. As a respect to the investigation on details in deep learning [4], [16], this paper compares the performance of recent deep models, including AlexNet [25], ZF [75], Overfeat [52], and GoogLeNet [60] under the same setting for different pretraining-finetuning schemes. We also provide experimental analysis on the properties that cause the accuracy variation in different object classes.

In this paper, we propose a deformable deep convolutional neural network for object detection; named as DeepID-Net. In DeepID-Net, we jointly learn the feature representation and part deformation for a large number of object categories. We also investigate many aspects in effectively and efficiently training and aggregating the deep models, including bounding box rejection, training schemes, context modeling, and model averaging. The proposed new framework significantly advances the state-of-the-art for deep learning based generic object detection, such as the well known RCNN [16] framework. This paper also provides detailed component-wise experimental results on how our approach can improve the mean Averaged Precision (AP) obtained by RCNN [16] from 31.0% to mean AP 50.3% step-by-step on the ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014) object detection task.

The contributions of this paper are as follows:

1) A new deep learning framework for object detection. It effectively integrates features representation learning, part deformation learning, context modeling, model averaging, and bounding box location refinement into the detection system. Detailed component-wise analysis is provided through extensive experimental evaluation. This paper is also the first to investigate the influence of CNN structures for the large-scale object detection task under the same setting. By changing the configuration of this framework, multiple detectors with large diversity are generated, which leads to more effective model averaging.

2) A new scheme for pretraining the deep CNN model. We propose to pretrain the deep model on the ImageNet image classification and localization dataset with 1000-class object-level annotations instead of with image-level annotations, which are commonly used in existing deep learning object detection [16], [60]. Then the deep model is fine-tuned on the ImageNet/PASCAL-VOC object detection dataset with 200/20 classes, which are the target object classes in the two datasets.

3) A new deformation constrained pooling (def-pooling) layer, which enriches the deep model by learning the deformation of object parts at any information abstraction levels. The def-pooling layer can be used for replacing the max-pooling layer and learning the deformation properties of parts.

4) Analysis on the object properties that influence the variation in object detection accuracy for different classes. Preliminary version of this paper is published in [38]. This paper include more analysis on the proposed approach and add experimental investigation on the properties that influence the accuracy in detecting objects.

The models pretrained by both image-level annotation and object-level annotation for AlexNet [25], ZF [75], Overfeat [52] and GoogLeNet [60] and the models after fine-tuning on ILSVRC2014 are provided online 1.

2 Related work
Since many objects have non-rigid deformation, the ability to handle deformation improves detection performance. Deformable part-based models were used in [12], [83], [41], [70], [65] for handling translational movement of parts. To handle more complex articulations, size change and rotation of parts were modeled in [13], and mixture of part appearance and articulation types were modeled in [3], [72]. A dictionary of shared deformable patterns was learned in [20]. In these approaches, features were manually designed.

Because of the power on learning feature representation, deep models have been widely used for object recognition, detection and other vision tasks [25], [52], [75], [21], [53]. However, [36] and [85]. 

In this paper, we extend it to general object detection on ImageNet. In [36], the deformation layer was constrained to be placed after the last convolutional layer. In this work the def-pooling layer can be placed after all the convolutional layers to capture geometric deformation at all the information abstraction levels. In [36], it was assumed that a pedestrian only had one instance of a body part, so each part filter only had one optimal response in a detection window. In this work, it is assumed that an object has multiple instances of a part (e.g. a car has many wheels), so each part filter is allowed to have multiple response peaks in a detection window. Moreover, we allow multiple object categories to share deformable parts and jointly learn them with a single network. This new model is more suitable for general object detection.

The use of context has gained attention in recent works on object detection. The context information investigated in literature includes regions surrounding objects [5], [8], [14], object-scene interaction [9], [22], and the presence, location, orientation and size relationship among objects [2], [64], [66], [7], [44], [14], [56], [9], [74], [8], [71], [40], [6], [51], [61]. In this paper, we use whole-image classification scores over a large number of classes from a deep model as global contextual information to refine detection scores.

Besides feature learning, deformation modeling, and context modeling, there were also other important components in the object detection pipeline, such as pretraining [16], network structures [52], [75], [25], refinement of bounding box locations [16], and model averaging [75], [25], [21]. While these components were studied individually in different works, we integrate them into a complete pipeline and take a global view of them with component-wise analysis under the same experimental setting. It is an important step to understand and advance deep learning based object detection.

![Diagram](image)

**Figure 2. Overview of our approach. Detailed description is given in Section 3.1. Texts in red highlight the steps that are not present in RCNN [16].**

## 3 Method

### 3.1 Overview of our approach

An overview of our proposed approach is shown in Fig. 2. We take the ImageNet object detection task as an example. The ImageNet image classification and localization dataset with 1,000 classes is chosen to pretrain the deep model. Its object detection dataset has 200 object classes. In the experimental section, the approach is also applied to the PASCAL VOC. The pretraining data keeps the same, while the detection dataset only has 20 object classes. The steps of our approach are summarized as follows.

1. Selective search proposed in [55] and edgeboxes proposed in [84] are used to propose candidate bounding boxes.
2. An existing detector, RCNN [16] in our experiment, is used to reject bounding boxes that are most likely to be background.
3. An image region in a bounding box is cropped and fed into the DeepID-Net to obtain 200 detection scores. Each detection score measures the confidence on the cropped image containing one specific object class. Details are given in Section 3.2.
4. The 1000-class whole-image classification scores of a deep model are used as contextual information to refine the detection scores of each candidate bounding box. Details are given in Section 3.6.
5. Average of multiple deep model outputs is used to improve the detection accuracy. Details are given in Section 3.7.
6. Bounding box regression proposed in RCNN [16] is used to reduce localization errors.

### 3.2 Architecture of DeepID-Net

DeepID-Net in Fig. 3 has three parts:

(a) The baseline deep model. The ZF model proposed in [75] is used as the default baseline deep model when it is not specified.

(b) Branches with def-pooling layers. The input of these layers is the conv5, the last convolutional layer of the baseline model. The output of conv5 is convolved with part filters of variable sizes and the proposed def-pooling layers in Section 3.4 are used to learn the deformation constraint of these part filters. Parts (a)-(b) output 200-class object detection scores. For the cropped image region that contains a horse as shown in Fig. 3(a), its ideal output should have a high score for the object class horse but low scores for other classes.
1) Pretrain the deep model with object-level annotations of 1,000 classes from ImageNet Cls-Loc train data.

2) Fine-tune the deep model for the 200-class object detection task, i.e., using object-level annotations of 200 classes from ImageNet Det train and val\(_1\) (validation set 1) data. Use the parameters in Step (1) as initialization. Compared with the training scheme of RCNN, experimental results show that the proposed scheme improves mean AP by 4.5\% on ImageNet Det val\(_2\) (validation set 2). If only the 200 target classes (instead of 1,000 classes) from the ImageNet Cls-Loc train data are selected for pretraining in Step (1), the mean AP on ImageNet Det val\(_2\) drops by 5.7\%.

Another potential mismatch between pretraining and fine-tuning comes from the fact that the ImageNet classification and localization (Cls-Loc) data has 1,000 classes, while the ImageNet detection (Det) data only targets on 200 classes, most of which are within the 1,000 classes. In many practical applications, the number of object classes to be detected is small. People question the usefulness of auxiliary training data outside the target object classes. Our experimental study shows that feature representations pretrained with 1,000 classes have better generalization capability, which leads to better detection accuracy than pretraining with a subset of the Cls-Loc data only belonging to the 200 target classes in detection.

3.4 Def-pooling layer

3.4.1 DPM and its relationship with CNN

In the deformable part based model (DPM) [12] for object detection, the following steps are used at the testing stage:

1) Extract HOG feature maps from the input image.
2) Obtain the part detection score maps by filtering the HOG feature maps with the learned part filters/detectors. The part filters are learned by latent SVM.
3) Obtain deformable part score maps by subtracting deformation penalty from part detection score maps.
4) Sum up the deformable part score maps from multiple parts to obtain the final object detection score map. Denote the convolutional layer at the \(l\)th layer by \(\text{conv}_l\). Denote the output maps of \(\text{conv}_l\) by \(M_l\). The steps above for DPM have the following relationship for CNN:

1) The HOG feature map in DPM corresponds to the output of a convolutional layer. Consecutive convolutional layers and pooling layers can be considered as extracting feature maps from input image.

2) The part detection maps in DPM correspond to the output response maps of the convolutional layer. For example, the output of \(\text{conv}_{l-1}\) is the feature maps \(M_1\), which are treated as input feature maps of \(\text{conv}_l\). Filtering on HOG feature maps using part filters in DPM is similar to filtering on the feature maps \(M_1\) using the filters of \(\text{conv}_l\) in CNN. Each output channel of \(\text{conv}_l\) corresponds to a part detection map in DPM. The filter of \(\text{conv}_l\) for an output channel corresponds to a part filter in DPM. The response map in CNN is called part detection map in the following of this paper.

3) The deformation penalty in DPM for each part corresponds to the deformation penalty in our proposed def-pooling layer for CNN. Details are given in Section 3.4.2.
3.4.2 Definition of the def-pooling layer

Similar to max-pooling and average-pooling, the input of a def-pooling layer is the output of a convolutional layer. The convolutional layer produces \( C \) maps of size \( W \times H \). Denote \( M_c \) as the \( c \)th map. Denote the \((i,j)\)th element of \( M_c \) by \( m_{c,i,j} \), \( i = 1, \ldots, W; j = 1, \ldots, H \). The def-pooling layer takes a small block with center \((x,y)\) and size \((2R+1) \times (2R+1)\) from the \( M_c \) and produce the element of the output as follows:

\[
\hat{b}^{(x,y)} = \max_{\delta_x, \delta_y \in (-R, R)} \tilde{m}_c(z_{\delta_x, \delta_y}, \delta_x, \delta_y),
\]

where \( x = 1, \ldots, W; y = 1, \ldots, H \).

\[
\tilde{m}_c(z_{\delta_x, \delta_y}, \delta_x, \delta_y) = m_{c,x,y}^{\delta_x, \delta_y} - \phi(\delta_x, \delta_y)
\]

\[
\phi(\delta_x, \delta_y) = \sum_{n=1}^{N} a_{c,n} d_{c,n}^{\delta_x, \delta_y}.
\]

\[ (x, y) \] denotes the assumed anchor location of object part.

\[ (\delta_x, \delta_y) \] denotes the translation/displacement of object part from the anchor position.

\[ z_{\delta_x, \delta_y} \] as defined in (4) is the deformed location from the assumed anchor position.

\[ m_{c,x,y}^{\delta_x, \delta_y} \] in (3) is the element in \( M_c \) at the location \( z_{\delta_x, \delta_y} \).

It is considered as the score of matching the \( c \)th filter with the features at the deformed location \( z_{\delta_x, \delta_y} \).

\[ \phi(\delta_x, \delta_y) \] in (3) and (5) is the deformation penalty of placing the part from the assumed anchor position \((x,y)\) to the deformed location \( z_{\delta_x, \delta_y} \). \( a_{c,n} \) and \( d_{c,n}^{\delta_x, \delta_y} \) in (5) are parameters of deformation that can be pre-defined or learned by back-propagation (BP). \( N \) denotes the number of parameters \( a_{c,n} \) and \( d_{c,n}^{\delta_x, \delta_y} \).

\[ \tilde{m}_c(z_{\delta_x, \delta_y}, \delta_x, \delta_y) \] in (1) and (3) is the deformable part score. It is obtained by subtracting the deformation penalty \( \phi(\delta_x, \delta_y) \) from the visual matching score \( m_{c,x,y}^{\delta_x, \delta_y} \).

\[ \hat{b}^{(x,y)} \] is the \((x,y)\)th element of the output of the def-pooling layer. For the anchor location \((x,y)\), \( \hat{b}^{(x,y)}_c \) is obtained by taking the maximum deformable part score \( \tilde{m}_c(z_{\delta_x, \delta_y}, \delta_x, \delta_y) \) within the displacement range \( R \), i.e. \( \delta_x, \delta_y \in \{-R, \ldots, R\} \).

The def-pooling layer can be better understood through the following examples.

Example 1. If \( N = 1, a_{n_1} = 1, a_{n_2}^{\delta_x, \delta_y} = 0 \) for \( |\delta_x|, |\delta_y| \leq k \) and \( a_{n_3}^{\delta_x, \delta_y} = \infty \) for \( |\delta_x|, |\delta_y| > k \), then this corresponds to max-pooling with kernel size \( k \). It shows that the max-pooling layer is a special case of the def-pooling layer. Penalty becomes very large when deformation reaches certain range. Since the use of different kernel sizes in max-pooling corresponds to different maps of deformation penalty that can be learned by BP, def-pooling provides the ability to learn the map that implicitly decides the kernel size for max-pooling.

Example 2. If \( N = 1 \) and \( a_{n_1} = 1 \), then \( a_{n_2}^{\delta_x, \delta_y} = 1 \) for \( (\delta_x, \delta_y) \in \{(0,0)\} \). It allows to assign different penalty to displacement in different directions. If the part has penalty 2 moving leftward and penalty 1 moving rightward, then we have \( a_{n_1}^{\delta_x, \delta_y} = 1 \) for \( (\delta_x, \delta_y) = (\delta_x, 0) \) and \( a_{n_1}^{\delta_x, \delta_y} = 2 \) for \( (\delta_x, \delta_y) > (0,0) \). Fig. 6 shows some learned deformation parameters \( a_{n_1}^{\delta_x, \delta_y} \). Fig. 7 shows some visualized parts.

Example 3. The def-pooling layer in [36] is a special case of the def-pooling layer by enforcing that \( z_{\delta_x, \delta_y} \) in (1) covers all the locations in \( conv_{1 \ldots 1,c} \) and only one output with a pre-defined location is allowed for the def-pooling layer (i.e. \( R = \infty \)). The proof can be found in Appendix A. To implement quadratic deformation penalty used in [12], we can define \( \{a_{n_1}^{\delta_x, \delta_y}\}_{n=1,2,3,4} = \{\delta_x, \delta_y, (\delta_x)^2, (\delta_y)^2\} \). As shown in Appendix A, the def-pooling layer under this setting can represent deformation constraint in the deformable part based model (DPM) [12] and the DP-DPM [18].

Take Example 2 as an example for BP learning. \( a_{c,n} \) is the parameter in this layer and \( d_{c,n}^{\delta_x, \delta_y} \) is pre-defined constant. Then we have:

\[
\frac{\partial \hat{b}^{(x,y)}_c}{\partial a_{c,n}} = -d_{c,n}^{\delta_x, \delta_y},
\]

where \( (\Delta_x, \Delta_y) \) is the position with the maximum value in (1). The gradients for the parameters in the layers before the def-pooling layer are back-propagated like max-pooling layer. Similar to max-pooling and average pooling, subsampling can be done as follows:

\[
b^{(x,y)}_c = \hat{b}^{(x-x, y-y)}_c
\]
The def-pooling layer has the following advantages.
1) It can replace any pooling layer, and learn deformation of parts with different sizes and semantic meanings. For example, at a higher level, visual patterns can be large parts, e.g., human upper bodies, and the def-pooling layer can capture the deformation constraint of human upper parts. At a middle level, the visual patterns can be smaller parts, e.g., heads. At the lowest level, the visual patterns can be very small, e.g., mouths. A human upper part is composed of a deformable head and other parts. The human head is composed of a deformable mouth and other parts. Object parts at different semantic abstraction levels with different deformation constraints are captured by def-pooling layers at different levels. The composition of object parts is naturally implemented by CNN with hierarchical layers.

2) The def-pooling layer allows for multiple deformable parts with the same visual cue, i.e., multiple response peaks are allowed for one filter. This design is from our observation that an object may have multiple object parts with the same visual pattern. For example, three light bulbs co-exist in a traffic light in Fig. 5.

3) As shown in Fig. 3, the def-pooling layer is a shared representation for multiple classes and therefore the learned visual patterns in the def-pooling layer can be shared among these classes. As examples in Fig. 8, the learned circular visual patterns are shared as different object parts in traffic lights, cars, and ipods.

The layers proposed in [36], [18] do not have these advantages, because they can only be placed after the final convolutional layer, assume one instance per object part, and does not share visual patterns among classes.

3.5 Fine-tuning the deep model with hinge-loss
In RCNN, feature representation is first learned with the softmax loss in the deep model after fine-tuning. Then in a separate step, the learned feature representation is input to a linear binary SVM classifier for detection of each class. In our approach, the softmax loss is replaced by the 200 binary hinge losses when fine-tuning the deep model. Thus the deep model fine-tuning and SVM learning steps in RCNN are merged into one step. The extra training time required for extracting features (∼ 2.4 days with one Titan GPU) is saved.

3.6 Contextual modeling
The deep model learned for the image classification task (Fig. 3 (c)) takes scene information into consideration while the deep model for object detection (Fig. 3 (a) and (b)) focuses on local bounding boxes. The 1000-class image classification scores are used as contextual features, and concatenated with the 200-class object detection scores to form a 1200 dimensional feature vector, based on which a linear SVM is learned to refine the 200-class detection scores. For a specific object class, not all object classes from the image classification model are useful. We learn a two-stage SVM to remove most of the classes. In the first stage, all scores from the 1000 classes are used for learning a linear SVM. At the second stage, the 10 classes with the largest magnitude in the linear SVM
weights learned in the first stage are selected as features and
then a linear SVM is learned for a given object class to be
detected. Therefore, only the classification scores of 10 classes
from the image classification deep model are used for each
class to be detected. The SVM is explicitly trained but not
within the network framework. If 5, 20, 50, 100 or 1000
classes are used, the mAP drops by 0, 0.2%, 0.8%, 0.9% and
4.5% respectively when compared with the result of using 10
classes. This result shows that only a few number of classes
are helpful for detection. The heuristic selection of 10 classes
helps to remove the effect from uncorrelated classes.

Take object detection for class volleyball as an example
in Figure 9. If only considering local regions cropped from
bounding boxes, volleyball are easy to be confused with
bathing caps and golf balls. In this case, the contextual infor-
mation from the whole-image classification scores is helpful,
since bathing caps appear in scenes of beach and swimming
pools, golf balls appear in grass fields, and volleyballs appear
in stadiums. The whole images of the three classes can be
better distinguished because of the global scenery information.
Fig. 9 plots the learned linear SVM weights on the 1000-class
image classification scores. It is observed that image classes
bathing cap and golf ball suppress the existence of volleyball
in the refinement of detection scores with negative weights,
while the image class volleyball enhances the detection score
of volleyball.

3.7 Combining models with high diversity

Model averaging has been widely used in object detection. In
existing works [75], [25], [21], the same deep architecture was
used. Models were different in cropping images at different
locations or using different learned parameters. In our model
averaging scheme, we learn models under multiple settings.
The settings of the models used for model averaging are shown
in Table 4. They are different in net structures, pretraining
schemes, loss functions for the deep model training, adding
def-pooling layer or not. The motivation is that models gen-
erated in this way have higher diversity and are complementary
to each other in improving the detection results after model
averaging. For example, although model no. 4 has low mAP,
it is found by greedy search because its pretraining scheme
is different from other models and provides complementary
scores for the averaged scores.

The 6 models as shown in Table 4 are automatically selected
by greedy search on ImageNet Det val2 from 10 models, and
the mAP of model averaging is 50.3% on the test data of
ILSVRC2014, while the mAP of the best single model is
47.9%.

4 Experimental results

Our experimental results are implemented based on the Caffe
[24]. Only selective search is used for proposing regions if not
specified.

Overall result on PASCAL VOC. For the VOC-2007 detec-
tion dataset [11], we follow the approach in [16] for splitting
the training and testing data. Table 1 shows the experimental
results on VOC-2007 testing data, which include approaches
using hand-crafted features [17], [48], [63], [62], [12], deep
CNN features [16], [21], [15], [47], [46], [68], and CNN
features with deformation learning [18]. Since all the state-
of-the-art works reported single-model results on this dataset,
we also report the single-model result only. Our model was
pretrained on bounding box annotation, with deformation,
without context, and with GoogLeNet as the baseline net.
Ours outperforms RCNN [16] and SPP [21] by about 5%
in mAP. RCNN, SPN and our model are all pre-trained on the
ImageNet Cls-Loc training data and fine-tuned on the VOC-
2007 training data. Table 2 shows the per-class mAPs for our
approach with G-Ntt and RCNN with VGG and GoogleNet
[16]. Fig. 10 shows the analysis on false positives using the
approach in [23].

Overall result on MS-COCO. Without using context, our
single model has mAP 25.6% on the MS-COCO Test-dev
dataset [28].

Experimental Setup on ImageNet. The ImageNet Large
Scale Visual Recognition Challenge (ILSVRC) 2014 [49]
contains two different datasets: 1) the classification and lo-
calization (Cls-Loc) dataset and 2) the detection (Det) dataset.
The training data of Cls-Loc contains 1.2 million images with
labels of 1,000 categories. It is used to pretrain deep models.
The same split of train and validation data from the Cls-Loc
is used for image-level annotation and object-level annotation
pretraining. The Det contains 200 object categories and is
split into three subsets, train, validation (val), and test data.
We follow RCNN [16] in splitting the val data into val1 and
val2. Val1 is used to train models, val2 is used to evaluate
separate components, and test is used to evaluate the overall
performance. The val1 /val2 split is the same as that in [16].

Overall result on ImageNet Det. RCNN [16] is used as the
state-of-the-art for comparison. The source code pro-
vided by the authors was used and we were able to re-
peat their results. Table 3 summarizes the results from
ILSVRC2014 object detection challenge. It includes the best
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2587642, IEEE Transactions on Pattern Analysis and Machine Intelligence.

4.1 Ablation study
The ImageNet Det is used for ablation study. Bounding box regression is not used if not specified.

4.1.1 Baseline deep model and bounding box rejection
As shown in Fig. 3, a baseline deep model is used in our DeepID-Net. Table 5 shows the results for different baseline deep models and bounding box rejection choices. AlexNet in [25] is denoted as A-net, ZF in [75] is denoted as Z-net, and Overfeat in [52] is denoted as O-net. GoogLeNet in [60] is denoted as G-net. Except for the two components investigated in Table 5, other components are the same as RCNN, while the new training schemes and the new components introduced in Section 3.2 are not included. The configuration in the second column of Table 5 is the same as RCNN (mean mAP 29.9%). Based on RCNN, applying bounding box rejection improves mAP by 1%. Therefore, bounding box rejection not only saves the time for training and validating new models, which is critical for future research, but also improves detection accuracy. Bounding box rejection is not constrained to particular detectors such as RCNN or fast RCNN. The time required to process one image is around 3.5 seconds per image using RCNN and around 0.2 seconds using fast RCNN. Both with bounding box rejection, ZF [75] performs better than AlexNet [25], with 0.9% mAP improvement. Overfeat [52] performs better than ZF, with 4.8% mAP improvement. GoogLeNet [60] performs better than Overfeat, with 1.2% mAP improvement.

Experimental results in Table 5 show the further investigation on the influence of bounding box rejection scheme in training and testing stage. Experimental results on two different CNN architectures, i.e. A-net and Z-net, show that the mAP is similar whether the rejection scheme in the testing stage is used or not. And the rejection scheme in the training stage is the main factor in improving the mAP. If there is concern that the rejection scheme results in lower recall of the candidate windows at the testing stage, the rejection scheme at the testing stage can be skipped. If not specified, bounding box rejection is used in both training and testing stages.

4.1.2 Investigation on the number of object classes at the pretraining stage
In order to investigate the influence from the number of object classes at the pretraining stage, we use the AlexNet and train on the ImageNet classification data without using the bounding box labels. Table 7 shows the experimental results. As pointed out in [50], the 200 classes in ImageNet detection corresponds to 494 classes in ImageNet classification.
shown in Table 8 using the new pretraining scheme in Section 3.4.2. Further pretraining, each sample carries much more information: for an apple image, the 3000-class pretraining provides further improvement by 2.4% compared with 1000-class pretraining. For 3000-class pretraining, each sample carries much more information: for an apple image, the 3000-class pretraining provides further improvement by 2.4% compared with 1000-class pretraining. 1000-class pretraining configuration is used for these three pretraining settings. Pretrained without context or def-pooling. Def-pooling, context and bounding box rejection are not used. Therefore, we investigate three pretraining settings: 1) use the corresponding 494-class samples in ImageNet classification training data but train the deep model as a 1000-class classification problem; 2) use the corresponding 494-class samples in ImageNet classification training data but train the deep model as a 200-class classification problem; 3) use the 1000-class samples in ImageNet classification training data and train the deep model as a 494-class classification problem. The same fine-tuning configuration is used for these three pretraining settings. Experimental results show that 494-class pretraining performs better than 200-class pretraining by 3% mAP. 1000-class pretraining performs better than 494-class pretraining by 4.3% mAP. 3000-class pretraining further improves the mAP by 2.4% compared with 1000-class pretraining. For 3000-class pretraining, each sample carries much more information: for an apple image, the 3000-class pretraining provides further information that it is not the other 2999 classes. And the use of more classes makes the training task challenging and not easy to overfit.

4.1.3 Investigation on def-pooling layer

Different deep model structures are investigated and results are shown in Table 8 using the new pretraining scheme in Section 3.3. Our DeepID-Net that uses def-pooling layers as shown in Fig. 3 is denoted as D-Def. Using the Z-net as baseline deep model, the DeepID-Net that uses def-pooling layer in Fig. 3 improves mAP by 2.5%. Def-pooling layer improves mAP by 2.3% for both O-net and G-net. This experiment shows the effectiveness of the def-pooling layer for generic object detection. In our implementation of def-pooling for G-net, we only replace max-pooling by def-pooling but did not add an additional feature maps like that in Fig. 3(b). 2.3% mAP improvement is still observed on G-net by replacing the max-pooling with def-pooling.

4.1.4 Investigation on different pretraining schemes and baseline net structures

There are two different annotation levels, image and object. Table 9 shows the results for investigation on annotation levels and net structures. When producing these results, other new components introduced in Section 3.4.3.6 are not included. For pretraining, we drop the learning rate by 10 when the classification accuracy of validation data reaches plateau, until no improvement is found on the validation data. For fine-tuning, we use the same initial learning rate (0.001) and the same number of iterations (20,000) for dropping the learning rate by 10 for all net structures, which is the same setting in RCNN [16].

Pretraining on object-level-annotation performs better than pretraining on image-level annotation by 4.4% mAP for A-net and 4.2% for Z-net. This experiment shows that object-level annotation is better than image-level annotation in pretraining deep model.

4.1.5 Investigation on the overall pipeline

Table 10 summarizes how performance gets improved by adding each component step-by-step into our pipeline. RCNN has mAP 29.9%. With bounding box rejection, mAP is improved by about 1%, denoted by +1% in Table 10. Based on that, changing A-net to Z-net improves mAP by 0.9%. Changing Z-net to O-net improves mAP by 4.8%. O-net to G-net improves mAP by 1.2%. Replacing image-level annotation by object-level annotation in pretraining, mAP is increased by 2.6%. By combining candidates from selective search and edgeboxes [84], mAP is increased by 2.3%. The def-pooling layer further improves mAP by 2.2%. Pretraining the object-level annotation with multiple scales [4] improves mAP by 2.2%. After adding the contextual information from image classification scores, mAP is increased by 0.5%. Bounding box regression improves mAP by 0.4%. With model averaging, the final result is 50.7%.

4.2 Per-Class Accuracy as a Function of Object Properties on ILSVRC14 Object Detection Data

Inspired by the analysis in [50], we perform analysis on the object properties that influence the variation in object detection accuracy for different classes in this section. Our result with 50.7% mAP on the val2 data is used for analysis.

---

**Table 5**

| Study of bounding box (bbox) rejection and baseline deep model on ILSVRC2014 val, Pretrained without bounding box labels. Def-pooling, context and bounding box regression are not used. |
|bbox rejection? | n | y | y | y | y | n | y | n | y | y | y | y | n | y | y | y |
| mAP (%) | 29.9 | 30.9 | 31.8 | 36.6 | 37.8 | 28.9 | 29.4 | 30.5 | 36.7 | 37 |
| median AP (%) | 28.9 | 30.5 | 30.5 | 30.5 | 30.5 | 28.9 | 29.4 | 30.5 | 30.5 | 30.5 |

**Table 6**

| Study of bounding box (bbox) rejection at the training and testing stage without context or def-pooling. Pretrained without bounding box labels. Def-pooling, context and bounding box regression are not used. |
|bbox rejection train? | n | y | y | y | y | n | y | n | y | y | y | y | n | y | y | y |
|bbox rejection test? | n | n | n | n | n | n | n | n | n | n | n | n | n | n | n | n |
| mAP (%) | 29.9 | 30.9 | 30.8 | 31.8 | 31.5 | 28.9 | 29.4 | 29.3 | 30.5 | 30.4 |
| median AP (%) | 28.9 | 29.4 | 29.3 | 30.5 | 30.4 |

**Table 7**

| Study of number of classes used for pretraining. AlexNet is used. Pretrained without bounding box labels. Def-pooling, context and bounding box regression are not used. |
|number of classes | 200 | 494 | 1000 | 3000 |
| mAP (%) | 22.6 | 25.6 | 29.9 | 32.3 |
| median AP (%) | 19.8 | 23.0 | 28.9 | 31.7 |

**Table 8**

| Investigation of def-pooling for different baseline net structures on ILSVRC2014 val, Use pretraining scheme 1 but no bounding box regression or context. |
| net structure | Z-net | D-Def(Z) | O-net | D-Def(O) | G-net | D-Def(G) |
| mAP (%) | 36.0 | 38.5 | 39.1 | 41.4 | 40.4 | 42.4 |
| median (%) | 34.9 | 37.4 | 37.9 | 41.9 | 39.3 | 42.3 |
Deformable objects have higher accuracy than rigid objects. Similar to [50], we also find that deformable (e.g., snake) and edgeboxes are used for proposing the regions. After

\[ N_c = \frac{1}{P_c} \sum_{p_c=1}^{P_c} (n_{p_c}), \]

where \( n_{p_c} \) is the number of objects within the image of the \( p_c \) th sample for class \( c \), \( p_c = 1 \ldots P_c \). \( N_c \) is obtained from the val1 data. When there are large number objects within an image, they may occlude each other and appear as background for the ground truth bounding box of each other, resulting in the added complexity of object appearance and background clutter. As shown in Fig. 13, some small objects, bee and butterfly, have less than 2 objects per image on average. And they have very high AP, 90.6% for butterfly and 76.9% for bee. We find that the images in val1 with these samples are mostly captured by tight shot, and they have relatively simple background. As shown in Fig. 13, the model performs better when the number of objects per image is smaller.

**Variance in rotation.** In-plane and out-of-plane rotation are factors that influence the within-class appearance variation. An ax with frontal view is very different in appearance from an ax with side view. An upright ax is very different in appearance from a horizontal ax. We labeled the rotation of objects for all the 200 classes on the val1 data and use them for obtaining the variance in rotation. As shown in Fig. 12, objects with lower variance in part existence performs better.

**Number of objects per image.** The number of object per image for the \( c \) th object class, denoted by \( N_c \), is obtained as follows:

\[ N_c = \frac{1}{P_c} \sum_{p_c=1}^{P_c} (n_{p_c}), \]

where \( n_{p_c} \) is the number of objects within the image of the \( p_c \) th sample for class \( c \), \( p_c = 1 \ldots P_c \). \( N_c \) is obtained from the val1 data. When there are large number objects within an image, they may occlude each other and appear as background for the ground truth bounding box of each other, resulting in the added complexity of object appearance and background clutter. As shown in Fig. 13, some small objects, bee and butterfly, have less than 2 objects per image on average. And they have very high AP, 90.6% for butterfly and 76.9% for bee. We find that the images in val1 with these samples are mostly captured by tight shot, and they have relatively simple background. As shown in Fig. 13, the model performs better when the number of objects per image is smaller.
bounding box rejection, 302 boxes per image are obtained on val1. The average recall is 89.19% for overlap greater than 0.5 and 78.98% for overlap greater than 0.7.

Fig. 15 shows the 5 classes with lowest and highest average precision and their corresponding factors. The 5 object classes with the lowest accuracy are mostly non-deformable, having low texture, small bounding box size, large number of objects per image, large variation in aspect ratio, part existence and rotation.

We also tried other properties, like variation in bounding box size, average aspect ratio, number of positive samples, but did not find them to have strong correlation to the detection accuracy.

Fig. 16 shows the object classes with large mAP improvement and mAP drop when the def-pooling is used. 140 of the 200 classes have their mAP improved. Def-pooling brings large mAP gains for mammals like squirrel with deformation and instruments like banjo with rotation. However, man-made objects such as waffle iron, digital clock, cocktail shaker and vacuum have inconsistent existence of object parts, large variation in rotation and part appearance. Therefore, the mAP gains are negative for these man-made objects.

Fig. 17 shows the detection accuracy for object classes grouped at different WordNet hierarchical levels. It can be seen that vertebrates that are neither mammal nor fish, i.e. bird, frog, lizard, snake, and turtle, have the largest mAP. Mammals also have large mAP because mammals share similar appearance, have rich texture and have many object classes that help each other in learning their feature representations. Generally, artifacts have lower mAP because they have low texture and large variation in shape. Texts in dashed boxes of Fig. 17 show the absolute mAP gain obtained by bounding box rejection and def-pooling for each group. It can be seen that def-pooling has 9.5% mAP gain for detecting person. Def-pooling has higher mAP gain (4.6%) for mammals with regular deformation and part appearance than substances (0.4%) with irregular deformation and part appearance.

5 Appendix A: Relationship between the Deformation Layer and the DPM

The quadratic deformation constraint in [12] can be represented as follows:

\[ \tilde{m}_{i,j}^{(l,j)} = m_{i,j}^{(l,j)} - a_1(i - b_1) - a_2(j - b_2) - \frac{a_3}{2a_1} (j - b_2)^2 + \frac{a_4}{2a_2} (i - b_1)^2, \]

where \( m_{i,j}^{(l,j)} \) is the \((i,j)\)th element of the part detection map \( M \). \((b_1, b_2)\) is the predefined anchor location of the \(l\)th part. They are adjusted by \( a_3/2a_1 \) and \( a_4/2a_2 \), which are automatically learned. \( a_1 \) and \( a_2 \) (9) decide the deformation cost. There is no deformation cost if \( a_1 = a_2 = 0 \). Parts are not allowed to move if \( a_1 = a_2 = \infty \). \((b_1, b_2)\) and \( (a_3, a_4) \) jointly decide the center of the part. The quadratic constraint in Eq. (9) can be represented using Eq. (10) as follows:

\[ \tilde{m}_{i,j}^{(l,j)} = m_{i,j}^{(l,j)} - a_1d_1^{(l,j)} - a_2d_2^{(l,j)} - a_3d_3^{(l,j)} - a_4d_4^{(l,j)} - a_5, \]

\[ d_1^{(i,j)} = (i - b_1)^2, \quad d_2^{(i,j)} = (j - b_2)^2, \quad d_3^{(i,j)} = i - b_1, \]

\[ d_4^{(i,j)} = j - b_2, \quad a_5 = a_3^2/(4a_1) + a_4^2/(4a_2). \]

In this case, \( a_1, a_2, a_3 \) and \( a_4 \) are parameters to be learned and \( d_4^{(i,j)} \) for \( n = 1, 2, 3, 4 \) are predefined. \( a_5 \) is the same in all locations and need not be learned. The final output is:

\[ b = \max_{(i,j)} \tilde{m}_{i,j}^{(l,j)}, \]

where \( \tilde{m}_{i,j}^{(l,j)} \) is the \((i,j)\)th element of the matrix \( M \) in (9).

6 Conclusion

This paper proposes a deep learning based object detection pipeline, which integrates the key components of bounding box reject, pretraining, deformation handling, context modeling, bounding box regression and model averaging. It significantly advances the state-of-the-art from mAP 31.0% to mAP 41.3%.
We will design an end-to-end system that jointly learns these detectors with large diversity are obtained, which leads to more effective model averaging. This work shows the important information handling approaches and deep architectures. Motivated by our insights on how to learn feature representations more suitable for the object detection task and with good generalization capability, a pretraining scheme is proposed. By changing the configurations of the proposed detection pipeline, multiple detectors with large diversity are obtained, which leads to more effective model averaging. This work shows the important modules in an object detection pipeline, although each has its own parameter setting set in an ad hoc way. In the future, we will design an end-to-end system that jointly learns these modules.

Acknowledgment: This work is supported by the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project Nos. CUHK14206114, CUHK14205615 and CUHK14207814, CUHK147011,CUHK14203015).

REFERENCES

[1] Deepdraw. Deepdraw on github.com/auduno/deepdraw. 6
Table 9
Ablation study of the two pretraining schemes in Section 3.3 for different net structures on ILSVRC2014 val. Scheme 0 only uses image-level annotation for pretraining. Scheme 1 uses object-level annotation for pretraining. Def-pooling bounding box regression and context are not used.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>class number</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>bbox rejection</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>pretrain scheme</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mAP (%)</td>
<td>29.9</td>
<td>34.3</td>
<td>34.9</td>
<td>31.8</td>
<td>29.9</td>
<td>35.6</td>
<td>36.0</td>
<td>36.6</td>
<td>39.1</td>
<td>37.8</td>
<td>40.4</td>
</tr>
<tr>
<td>meadian AP (%)</td>
<td>28.9</td>
<td>33.4</td>
<td>34.4</td>
<td>30.5</td>
<td>29.7</td>
<td>34.0</td>
<td>34.9</td>
<td>36.7</td>
<td>37.9</td>
<td>37.0</td>
<td>39.3</td>
</tr>
</tbody>
</table>

Table 10
Ablation study of the overall pipeline for single model on ILSVRC2014 val2. It shows the mean AP after adding each key component step-by-step.

<table>
<thead>
<tr>
<th>detection pipeline</th>
<th>RCNN +bbox rejection to Z-net to O-net to G-net</th>
<th>image to bbox</th>
<th>+edgbox</th>
<th>+Def</th>
<th>+multi-scale</th>
<th>+context</th>
<th>+bbox regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP (%)</td>
<td>29.9</td>
<td>30.9</td>
<td>31.8</td>
<td>36.6</td>
<td>37.8</td>
<td>42.7</td>
<td>44.9</td>
</tr>
<tr>
<td>meadian AP (%)</td>
<td>28.9</td>
<td>29.4</td>
<td>30.5</td>
<td>36.7</td>
<td>37.0</td>
<td>42.3</td>
<td>42.5</td>
</tr>
<tr>
<td>mAP improvement (%)</td>
<td>+1</td>
<td>+0.9</td>
<td>+4.8</td>
<td>+1.2</td>
<td>+2.6</td>
<td>+2.3</td>
<td>+2.2</td>
</tr>
</tbody>
</table>
IEEE TRANSACTIONS PATTERN ANALYSIS AND MACHINE INTELLIGENCE

2014. 1


Wanli Ouyang received the PhD degree in the Department of Electronic Engineering, The Chinese University of Hong Kong, where he is now a Research Assistant Professor. His research interests include image processing, computer vision and pattern recognition.

Xingyu Zeng received the B.S. degree in Electronic Engineering and Information Science from University of Science and Technology of China in 2011. He is currently a PhD student in the Department of Electronic Engineering at the Chinese University of Hong Kong. His research interests include computer vision and deep learning.

Xiaogang Wang received the PhD degree from the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology in 2009. He is currently an Assistant Professor in the Department of Electronic Engineering at The Chinese University of Hong Kong. His research interests include computer vision and machine learning.

Shi Qiu received the BS and PhD degree from Tsinghua University (2009) and the Chinese University of Hong Kong (2014), respectively. He is currently a research scientist at SenseTime Group Limited. His research interests include object recognition, object detection, and image search. He worked as a postdoctoral fellow at the Chinese University of Hong Kong from 2014 to 2015.

Ping Luo is a postdoctoral researcher at the Department of Information Engineering, The Chinese University of Hong Kong, where he received his Ph.D. in 2014. His research interests include deep learning, computer vision, and computer graphics, focusing on face analysis, pedestrian analysis, and large-scale object recognition and detection.

Yonglong Tian is currently an M.Phil student at Department of Information Engineering in The Chinese University of Hong Kong. His research interests include computer vision and machine learning, and object detection.

Hongsheng Li received the masters and doctorate degrees in computer science from Lehigh University, Pennsylvania, in 2010 and 2012, respectively. He is an associate professor in the School of Electronic Engineering at University of Electronic Science and Technology of China. His research interests include computer vision, medical image analysis, and machine learning.

Shuo Yang received the B.E. degree in Software Engineering from Wuhan University, China, in 2013. He is currently working toward the Ph.D. degree in the Department of Information Engineering at the Chinese University of Hong Kong. His research interests include computer vision and machine learning, in particular, face detection and object recognition.

Zhe Wang received the BEng in Optical Engineering from Zhejiang University, China, in 2012. He is currently working toward a Ph.D. degree in the Department of Electronic Engineering, the Chinese University of Hong Kong. His research interest is focused on compressed sensing, magnetic resonance imaging, image segmentation and object detection.

Hongyang Li received his B.E. degree in major Electronic and Information Engineering at Dalian University of Technology (DUT), 2014. From 2013 to 2014, he was a research assistant in the computer vision group at DUT. He is currently a Ph.D. candidate under supervision of Prof. Xiaogang Wang at The Chinese University of Hong Kong. His research interests focus on deep learning and object recognition.

Junjie Yan received his PhD degree in 2015 from National Laboratory of Pattern Recognition, Chinese Academy of Sciences. Since 2016, he is the principle scientist in SenseTime Group Limited and leads the research in object detection and tracking, and applications in video surveillance.

Kun Wang received the B.S. degree in Software Engineering from Beijing Jiaotong University in 2014. He is currently a research assistant in the Department of Electronic Engineering at the Chinese University of Hong Kong. His research interests include crowd analysis and object detection with deep learning.

Chen Change Loy received the PhD degree in Computer Science from the Queen Mary University of London in 2010. He is currently a Research Assistant Professor in the Department of Information Engineering, Chinese University of Hong Kong. Previously he was a postdoctoral researcher at Vision Semantics Ltd. His research interests include computer vision and pattern recognition, with focus on face analysis, deep learning, and visual surveillance.

Xiaou Tang received the B.S. degree from the University of Science and Technology of China, Hefei, in 1990, and the M.S. degree from the University of Rochester, Rochester, NY, in 1991. He received the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a Professor in the Department of Information Engineering and Associate Dean (Research) of the Faculty of Engineering at the Chinese University of Hong Kong. He worked as the group manager of the Visual Computing Group at the Microsoft Research Asia from 2010 to 2013. His research interests include computer vision, pattern recognition, and video processing. Dr. Tang received the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a program chair of the IEEE International Conference on Computer Vision (ICCV) 2009 and has served as an Associate Editor of IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) and International Journal of Computer Vision (IJCV). He is a Fellow of IEEE.