Chained Cascade Network for Object Detection

Wanli Ouyang[1,2], Kun Wang[1], Xin Zhu[1], Xiaogang Wang[1]
1. The Chinese University of Hong Kong 2. The University of Sydney
{wlouyang, kwang, xzhu, xgwang}@ee.cuhk.edu.hk

Abstract

Cascade is a widely used approach that rejects obvious negative samples at early stages for learning better classifier and faster inference. This paper presents chained cascade network (CC-Net). In this CC-Net, there are many cascade stages. Preceding cascade stages are placed at shallow layers. Easy hard examples are rejected at shallow layers so that the computation for deeper or wider layers is not required. In this way, features and classifiers at latter stages handle more difficult samples with the help of features and classifiers in previous stages. It yields consistent boost in detection performance on PASCAL VOC 2007 and ImageNet for both fast RCNN and Faster RCNN. CC-Net saves computation for both training and testing. Code is available on https://github.com/wk910930/ccnn.

1. Introduction

Object detection is a fundamental computer vision task. It differs from image classification in that the number of background samples (image regions not belonging to any object class of interest) is much larger than the number of object samples. This leads to the undesirable imbalance in the number of samples for different classes during training.

In order to handle the imbalance problem from the background samples, bootstrapping, cascade, and hard negative mining have been developed [32, 4, 34]. In these approaches, classifier learning is divided into multiple stages. In each stage, only a subset of background samples are used for training. The classifiers at earlier stages handle easier samples while the classifiers at latter stages handle more difficult samples. Bootstrapping and hard negative mining aim at learning more accurate classifier. In comparison, cascade improves both accuracy and speed of the detection process by rejecting easy background samples at both training and testing time. The essence of cascade is to learn more discriminative classifiers by using multi-stage classifiers. Classifiers at early stages discard large number of easy negative samples so that classifiers at latter stage can focus on handling more difficult examples. Motivated by these approaches, we propose a CC-Net to learn features and classifiers with multiple stage. The more discriminative features and classifiers are used for handling more difficult examples as the network goes deeper or wider.

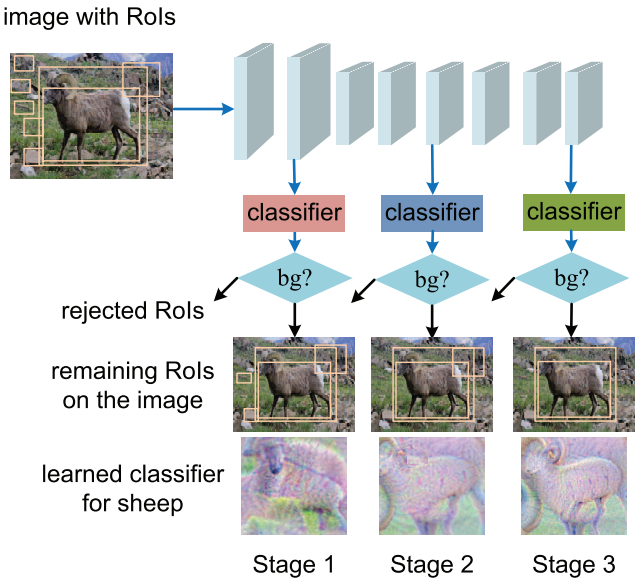

Figure 1. Motivation of the chained cascade ConvNet in rejecting large number of easy background samples for faster speed and more discriminative features.

Deep ConvNets have led to significant improvement in accuracy for object detection [11, 10]. Based on a baseline network, adding deeper layers [30] or branches for making the network wider [9, 41, 1] were found to improve accuracy for object detection and classification. But this also increases computational time for both training and testing. Empirical results in [29] show that very large batch size for ConvNet provides improvement in detection accuracy, but leads to increase in training time. In order to reduce the computational complexity from more training samples and more complex networks, we design a cascaded network structure. This network has many stages for rejecting easy samples within a single CNN. When an easy sample is rejected at a shallow layer, it need not go through deeper layers, for which the computation and memory are not required. With this design, the huge number of training samples can be used but the memory and time for training and testing are still acceptable by rejecting large number of easy examples at earlier cascade stages.

Based on the observations above, we design a chained cascade network (CC-Net) for object detection. The contribution of this design is as follows:

- Cascade is used in both training and testing the CC-Net to save computation. In the CC-Net, when a sample is rejected at shallow layers, the computation and memory at deeper layers are not require for it.
- In the network, the early cascade stage and contextual cascade stage are used for learning more and more discriminative features. By rejecting easy samples at shallow layers in the network, the features and classifiers learned at deeper layers or extra branches focus on harder samples. In this way, the learned features and classifiers are better in handling the harder examples. As shown by the example in Fig. 1, when the first classifier finds that the object should only be a mammal, then the features and classifier at the second stage can focus more on distinguishing specific class of mammal like horse, sheep and cattle. And the classifier at the third stage then learns features that help to distinguish mammals with horns, e.g. sheep and cattle.
- All the cascaded classifiers and their corresponding features are jointly learned through a single ConvNet.

With BN-Net [14] as the baseline model, our approach provides 6.3% mAP gain compared with BN-Net. Testing time of the CC-Net is 14% of the GBD-Net and 30% of the CC-Net without cascade. Experimental results on ImageNet and Pascal VOC 2007 show improvement in detection accuracy for different region proposal methods.

2. Related work

Cascade and bootstrapping for hand crafted features. Cascade has appeared in various forms dating back to the 1970s, as was pointed out by Schneiderman [27]. It has been widely used in object detection [7, 2, 5, 17]. Cascade can be applied for SVM [7], boosted classifiers [5, 17, 35]. Chaining of classifiers among cascade stages was called soft cascade [2] and boosting chain in [35]. In these approaches, the detection scores in the previous stages were added to the detection scores in the current stage. Classifier chaining was effective in face detection [2, 35] and generic object detection [7] for hand crafted features. Bootstrapping was introduced in the work of Sung and Poggio [32] in the mid-1990s for training face detection models. Bootstrapping was frequently used when training SVMs for object detection [4, 8]. These works for hand-crafted features did not learn features and cascaded classifiers jointly. And they learned cascaded classifiers by multiple separate steps. Different from these works, multiple cascaded classifiers and features in our CC-Net are learned by the ConvNet with a one-step training.

Design of better CNN model. Deeper ConvNets were effective for image classification and object detection [15, 28, 30, 33, 13]. On the other hand, wide residual network [38], inception modules [33, 3], and multi-region features [9, 1, 39, 41, 40] improved the image classification and object detection accuracy by increasing the width of ConvNets. Our work is complementary to the works above that learn better features. CC-Net can use these deeper and wider networks to obtain diverse features for cascade.

Recent CNN based methods for object detection. Deep models were found to be effective in object detection [11, 10, 12, 24, 25, 18, 21, 22, 41, 19, 19, 20]. Cascade of ConvNets was found to be effective in region proposal and region classification [20, 37, 16]. These works learn separate CNNs through multiple steps, while our CC-Net jointly learns all cascaded features and classifiers through one-step training. Shrivastava *et al*. introduced online mining of hard positive and negative examples for ConvNet-based detector [29]. Joint learning of cascaded ConvNets were proposed in [23]. The approaches in [16, 23] cascade on multiple networks, where each network has its own input image and is used as one cascade stage. In our work, a single network with one input image has many cascade stages, which is complementary to the cascade of multiple networks in [16, 23]. Early cascade, contextual cascade, and chaining of scores in multiple stages are not built up in [23, 29] for face detection but are used for learning better features in our CC-Net for generic object detection. Our approach saves memory and computational time for both training and testing. In comparison, the approach in [23] only saves time for testing and the approach in [29] only saves time for backward-propagation during training.

3. The CC-Net for object detection

3.1. Overview of the CC-Net

Brief introduction of the fast RCNN. This paper adopts the fast RCNN framework for illustrating the CC-Net for object detection. In fast RCNN, 1) a set of regions of interest (RoIs) are generated by a region proposal approach; 2) CNN feature maps for the input image are generated by several convolutional layers; 3) the roi-pooling layer projects the RoIs onto the CNN feature maps and extracts feature maps of the same size for RoIs of different sizes; 4) the layers after roi-pooling are conducted to obtain the final features, from which the classification scores and the regressed coordinates for bounding-box relocalization are predicted.

Fig. 2 shows an overview of the CC-Net. The existing approaches in [31, 37] are used for generating RoIs in our implementation. Based on the fast RCNN framework, it uses several convolutional layers for extracting convolutional features from the image and then use roi-pooling for projecting features of RoIs into the fixed size.

At the *early cascade stage*, features from shallow layers are used for rejecting RoIs belonging to background by multiple cascade stages. In each stage, features are roi-pooled from the image feature map. The roi-pooled features are then used for classification. At a stage, the RoIs rejected by previous stages do not have their features roi-pooled from

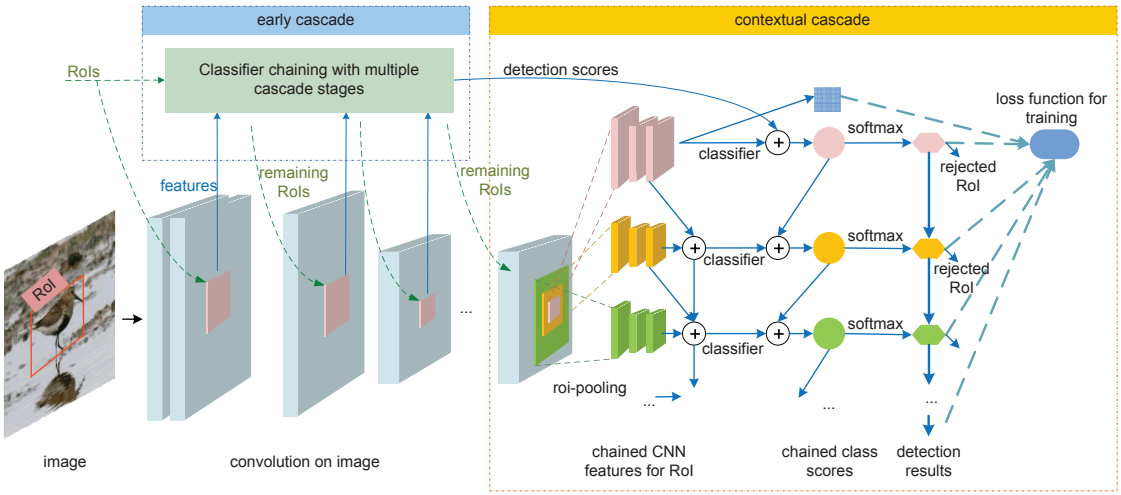

Figure 2. Overview of the CC-Net. Several convolutional layers are used on the input image. At the early cascade stage, roi-pooled features are used for rejecting easy background samples. Then roi-pooling is used for obtaining features of different layers, resolutions and contextual regions at the contextual cascade stage for further rejecting easy background samples. Classifiers chaining is used in both early cascade and contextual cascade stages for both training and testing. Feature chaining is used for passing the information from one contextual region to another at the contextual cascade stage. Bounding box regression and all cascaded classifiers are jointly learned. Best viewed in color.

image feature map for classification.

At the *contextual cascade stage*, the RoIs not rejected in the early cascade stage are used for obtaining features of different contextual regions. These features are then used by the chained features and classifiers with multiple cascade stages for further rejecting easy negative samples. If a RoI is not rejected by the cascade, its final classification score is used as the detection score.

The major modifications to fast RCNN are as follows:

- Classifiers with several cascade stages are used for object detection. At each cascade stage, many background RoIs are rejected. RoIs not rejected go to the next stage. By classifier chaining, classification scores in previous stages are used for classification in the current stage.
- Classifiers at different stages use different features. These CNN features can be different in depth, learned parameters, resolution and contextual regions.
- The features in previous stages can be chained with the features at the current stage. With this chaining, the features at previous stages serve as the prior knowledge for the features at the current stage.
- The bounding box regressor, feature chaining, classifier chaining in both early cascade and contextual cascade stages are learned end-to-end through back-propagation from a loss function.

In our implementation, the BN-Net in [14] is used as the baseline ConvNet if not specified. Fig. 3 shows the CC-Net based on the BN-Net. If only single stage of features and classifiers is used and the early cascade stage is removed, then Fig. 3 becomes a fast RCNN implementation of the BN-Net. There are ten inception modules in the BN-Net. The roi-pooling layer is placed after the second, third, fourth and seventh module, which are denoted by icp(3b), (3c), (4a) and (4d) in Fig. 3. In the early cascade stage, the roi-pooling for icp(3b), (3c), and (4a) use the same contextual region as the tight RoI. At stage 1, the features from icp(3b) is used for obtaining classification score. At stage 2, if the RoI is not rejected at stage 1, then the roi-pooled features from icp(3c) for this RoI is used for classification. Similarly for icp(4a) at stage 3. The remaining RoIs after stage 3 are then used for contextual cascade. In the contextual cascade stage, the roi-pooling from icp(4d) is used for obtaining features of different resolutions and contextual regions. The features after roi-pooling from icp(4d) for stage t is denoted by $\mathbf{h}_t$, $t = 1, 2, 3, 4$. At stage t, the features in $\mathbf{h}_t$ go through the modules icp(4e)$_t$, (5a)$_t$, (5b)$_t$ and global average pooling for obtaining features $\mathbf{f}_t$. Then these features are combined by feature chaining, with details in Section 3.2. The chained features are then used by chained classifiers with multiple cascade stages for detecting objects, with details in Section 3.3.

3.2. Feature chaining

3.2.1 Preparation of features with diversity

Classifiers in different cascade stages can use different features. Multi-region, multi-context features were found to be effective in [1, 9, 41]. In order to obtain features with diversity, we apply roi-pooling from image features using different contextual regions and resolutions. In our CC-Net, the roi-pooled features have the same number of channels but have different sizes at different stages. The sizes of roi-pooled features are respectively 14×14, 22×22, 16×16 and 14×14 for features at stages 1, 2, 3 and 4. These sizes are heuristically selected to have features with different contexts. The contextual regions for these features are

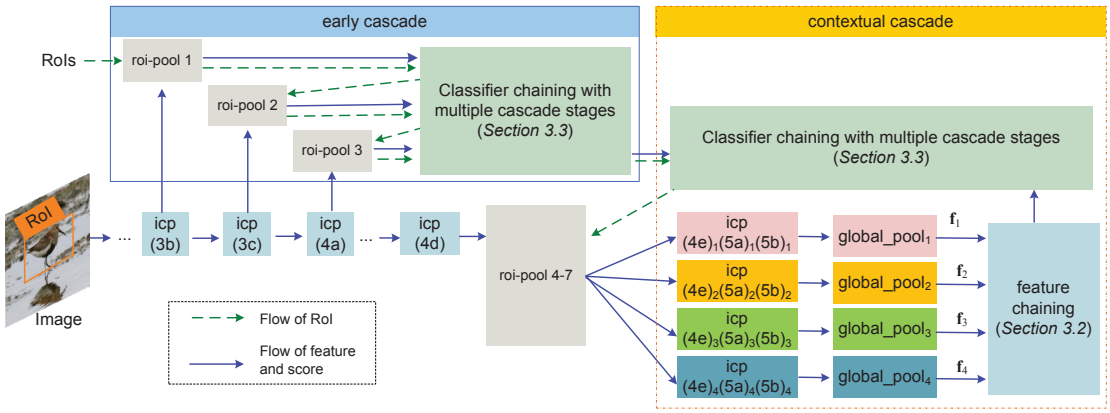

Figure 3. An example of the CC-Net for the BN-Net [14]. There are ten inception modules in the BN-Net. Each inception module consists of several convolutional layers. icp(3b), (3c), (4a), (4d), (4e), (5a), (5b) in the figure are respectively the second, third, fourth, seventh, eighth, ninth and tenth inception modules of the BN-Net. Best viewed in color.

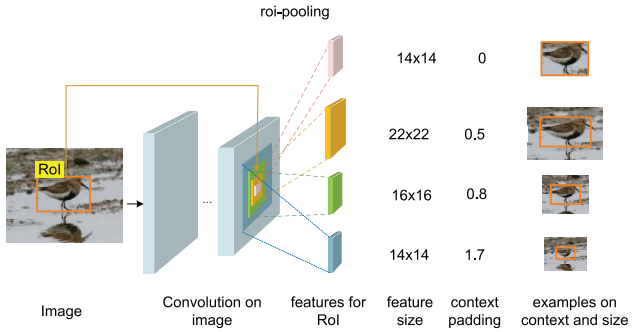

Figure 4. The use of roi-pooling to obtain features with different resolutions and contextual regions. After roi-pooling, features in different stages have different sizes and contextual padding value c. The original box size is used when $c = 0$. And 1.5 times the original box size is used when $c = 0.5$. Best viewed in color.

also different. Suppose the RoI has width W and height H. Denote c as the context padding value for the RoI. The padded region has the same center as the RoI and has width $(1+c)\cdot W$ and height $(1+c)\cdot H$. $c = 0, 0.5, 0.8$, and 1.7 for stages 1, 2, 3, and 4 respectively. Fig. 4 shows the contextual regions for features at different stages. These features are arranged with increasing contextual regions.

After features of different resolutions and contextual regions are obtained, they go through the remaining three inception modules (4e), (5a) and (5b). In order to increase the variation of features, the inception modules at different cascade stages have different parameters. Denote the inception module (4e) at stage t by $(4e)_t$. The modules $(4e)_1$, $(4e)_2$, $(4e)_3$, and $(4e)_4$ are initialized from the same pretrained inception module (4e) but have different parameters during the finetuning stage because they receive different gradients in backpropagation. The treatment for the module $(4e)_t$ are also applied for the inception modules $(5a)_t$ and $(5b)_t$. The CNN features obtained from the inception modules $(5b)_t$ have different sizes. We use global average pooling for these features so that they have the same size before feature chaining.

3.2.2 The feature chaining structure

Denote the features at depth l and stage t as $\mathbf{h}_{l,t}$. In order to use the features in previous stages as the prior knowledge when learning features for stage t, we design the feature chaining which has the following formulation:

$$\mathbf{h}_{l,t} = \psi(\mathbf{h}_{l,t-1}, \mathbf{o}_{l,t}), \quad (1)$$

$$\mathbf{o}_{l,t} = \phi(\mathbf{h}_{l-1,t}, \Theta_{l-1,t}), \quad (2)$$

where $\Theta_{l-1,t}$ contains the parameters learned from the network. In this design, the $\mathbf{h}_{l,t}$ is obtained from the features in previous stages $\mathbf{h}_{l,t-1}$ and features from the shallower layer $\mathbf{h}_{l-1,t}$. $\mathbf{o}_{l,t}$ denotes the features after nonlinear mapping of $\mathbf{h}_{l-1,t}$. The nonlinear mapping in $\phi(\mathbf{h}_{l-1,t}, \Theta_{l-1,t})$ in (2) can be implemented by convolutional layer or fully connected layer with nonlinear activation function. $\psi(\mathbf{h}_{l,t-1}, \mathbf{o}_{l,t})$ can be implemented by concatenation, *i.e.* $\psi(\mathbf{h}_{l,t-1}, \mathbf{o}_{l,t}) = Concat(\mathbf{h}_{l,t-1}, \mathbf{o}_{l,t})$, where $Concat$ is the feature concatenation operation. As another choice, $\psi(\mathbf{h}_{l,t-1}, \mathbf{o}_{l,t})$ in (1) can also be implemented by weighted averaging, *i.e.* $\psi(\mathbf{h}_{l,t-1}, \mathbf{o}_{l,t}) = \mathbf{h}_{l,t-1} + \mathbf{a}_{l,t} \odot \mathbf{o}_{l,t}$. The operation $\odot$ denotes the Hadamard product, where $[\alpha_1\ \alpha_2] \odot [\beta_1\ \beta_2] = [\alpha_1\beta_1\ \alpha_2\beta_2]$. $\mathbf{a}_{l,t-1}$ denotes a vector of scalers for scaling the $\mathbf{o}_{l,t}$.

In our implementation for the BN-Net as shown in Fig. 5, feature chaining is placed after the global average pooling, where all features are spatially pooled to have spatial size 1×1 and 1024 channels. Denote the features after global pooling for stage t as $\mathbf{o}_t$. The following procedure can be used for obtaining the chained features when weighted averaging is used:

$$\begin{aligned} \mathbf{f}_1 &= \mathbf{o}_1, \\ \mathbf{f}_2 &= \mathbf{o}_2 \odot \mathbf{a}_2 + \mathbf{f}_1, \\ \mathbf{f}_3 &= \mathbf{o}_3 \odot \mathbf{a}_3 + \mathbf{f}_2, \\ \mathbf{f}_4 &= \mathbf{o}_4 \odot \mathbf{a}_4 + \mathbf{f}_3. \end{aligned} \quad (3)$$

In this implementation, the feature $\mathbf{f}_t$ at stage t is obtained by summing up features $\mathbf{f}_{t-1}$ in the previous stage and the

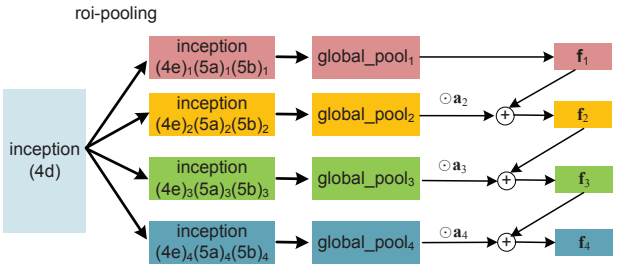

Figure 5. Chaining features for BN-Net.

$\mathbf{o}_t \odot \mathbf{a}_t$, which which is the output from the previous layer global_pool$_t$ weighted by $\mathbf{a}_t$. The summed features $\mathbf{f}_t, t = 1, 2, 3, 4$ are then used for classification.

3.2.3 Discussion

Feature chaining includes the concept of stage. Features $\mathbf{h}_{l,t}$ and $\mathbf{h}_{l,t+1}$ have the same depth but are different in stages. Features in different stages have specific objectives – they are used by classifiers for rejecting easy background samples. The features of the same depth but different stages communicate through feature chaining.

If the feature at the current stage is helpful for detection, its learned weight should be high. Otherwise, the weight is low. Thus the weight controls the amount of information to be transmitted for better detection accuracy.

With feature chaining, features at the current stage take the features in previous stages into consideration. Therefore, the CNN layers at the stage t no longer need to represent the information existing in previous stages. Instead, they will focus on representations that are complementary to those in previous stages.

3.3. Classifier chaining in CC-Net

3.3.1 Cascade with classifier chaining

This section briefly introduces cascade and chaining of binary classifiers, which is called soft cascade in [2] and boosting chain in [35].

Classifier chaining. Denote $\mathbf{f}_t$ as the features for the classifier at stage t, $t = 1, 2, \ldots, T$. Denote $c_t(\mathbf{f}_t)$ as the classification function for the feature $\mathbf{f}_t$ at the stage t. The partial sum of classification scores up to and including the tth stage is defined as follows:

$$ps_t = \sum_{i=1,\ldots,t} c_t(\mathbf{f}_t). \tag{4}$$

In classifier chaining, the partial sum ps_t of classification scores are obtained.

Cascade after classifier chaining. In the cascade, the partial sum ps_t is compared with the threshold r_t. If $ps_t < r$, then the sample is not considered as an object. Otherwise, the next stage of comparison is performed. If the sample is not rejected after T stages of such rejection scheme, the score ps_T will be used as the detection score. The main difference between cascade with classifier chaining and conventional cascade is that conventional cascade only uses $c_t(\mathbf{f}_t)$ as the score at the stage t but cascade with classifier chaining includes the previous scores.

3.3.2 Classifier chaining at the testing stage in CC-Net

In the CC-Net, the partial sum of classification scores up to and including the tth stage is obtained from the set of features $\{\mathbf{f}_t\}$ as follows:

$$\tilde{\mathbf{p}}_t = [p_{t,1} \ \ldots \ p_{t,K+1}]^{\mathrm{T}} = \sum_{i=1,\ldots,t} (\mathbf{b}_t \odot \mathbf{c}_t(\mathbf{f}_t)). \tag{5}$$

The $\mathbf{c}_t(\mathbf{f}_t)$ in (5) denotes the $K+1$-class classifier which takes the feature $\mathbf{f}_t$ as input and outputs $K+1$ classification scores on the input sample being one of the K classes or background. $\mathbf{c}_t(\mathbf{f}_t)$ is implemented using the fully connected (fc) layer in the CC-Net. The $\sum$ in (5) denotes the summation over vectors. The operation $\odot$ in (5) denotes the Hadamard product. $\mathbf{b}_t$ for this dot product is the vector of scaling parameters for controlling the scale of the classification scores. The scores $\tilde{\mathbf{p}}_t$ in (5) are normalized to probabilities $\mathbf{p}_t$ using the softmax function as follows:

$$\mathbf{p}_t = [p_{t,1} \ \ldots \ p_{t,K+1}] = \mathrm{softmax}(\tilde{\mathbf{p}}_t), \tag{6}$$

$$\text{where } p_{t,k} = \tilde{p}_{t,k} / \sum_{k=1}^{K+1} \tilde{p}_{t,k}. \tag{7}$$

The probabilities $\mathbf{p}_t$ are used for deciding whether to reject the given sample or not as follows:

$$u(\mathbf{p}_t, r_t) = \begin{cases} 1, & \text{if } \max\{p_{t,1} \ \ldots \ p_{t,K}\} > r_t, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

If $u(\mathbf{p}_t, r_t) = 0$, then the sample is considered as a background and the convolutional layers at latter stages are not used for saving testing time. For example, if a RoI is considered as the negative sample after using its roi-pooled features from the second inception module, which is the icp(3b) in Fig. 3, then this RoI is not used by the latter classifiers and the contextual cascade for this RoI is not done for saving computation. Conservative threshold r_t is chosen so that many background RoIs are rejected and most of the foreground RoIs are retained in the cascade stages. If a RoI is not rejected after T cascade stages, then $\mathbf{p}_T$ is used as its detection result. Fig. 6 shows the diagram for cascade chaining at the testing stage. In the CC-Net, cascade chaining is used for early cascade and contextual cascade. It is also used between these two stages, i.e. the score from early cascade is transmitted to the contextual cascade stage.

3.3.3 Training CC-Net

A multi-task loss of classification and bounding-box regression is used to jointly optimize the CC-Net. Suppose there

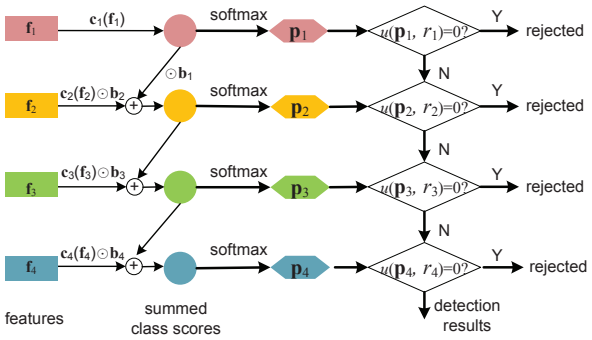

Figure 6. The chaining of features and classifiers at the testing stage. Given features $\mathbf{f}_t$, an fc layer is used for obtaining classification score at stage t. The classification sores from previous stages are combined with the scores at the current stage to obtained the summed scores. The summed scores undergo softmax to obtain the normalized scores $\mathbf{p}_t$ at stage t. Then the thresholding function $u(\mathbf{p}_t, r_t)$ decides whether to reject the sample or not. The sample not rejected after T stages uses the $\mathbf{p}_T$ as the detection result. $T = 4$ in the figure. Best viewed in color.

are K object classes to be detected. Denote the set of estimated class probabilities for a sample by $\mathbf{p} = \{\mathbf{p}_t | t = 1, \ldots, T\}$, where $\mathbf{p}_t = [p_{t,0} \ \ldots p_{t,K}]$ is the estimated probability vector at stage t and $p_{t,k}$ is the estimated probability for the kth class. $k = 0$ denotes the background. $\mathbf{p}_t$ is obtained by a softmax over the $K + 1$ outputs of a fc layer. Another layer outputs bounding-box regression offsets $\mathbf{l} = \{\mathbf{l}^k | k = 1, \ldots \mathrm{K}\}$, $\mathbf{l}^k = (l^k_{\mathrm{x}}, l^k_{\mathrm{y}}, l^k_{\mathrm{w}}, l^k_{\mathrm{h}})$ for each of the K object classes, indexed by k. Parameterization for $\mathbf{l}^k$ is the same as that in [11]. The loss function is defined as follows:

$$L(\mathbf{p}, k^*, \mathbf{l}, \mathbf{l}^*) = L_{cls}(\mathbf{p}, k^*) + L_{loc}(\mathbf{l}, \mathbf{l}^*, k^*), \quad (9)$$

$$L_{cls}(\mathbf{p}, k^*) = -\sum_{t=1}^{T} \lambda_t u_t \log p_{t,k^*}, \quad (10)$$

$$u_t = \prod_{i=1}^{t-1} [p_{i,k^*} < r_i] \text{ when } t > 1, \quad u_1 = 1. \quad (11)$$

$L_{cls}(*)$ is the loss for classification and L_{loc} is the loss for bounding-box regression. If $\lambda_t = u_t = 1$ and $T = 1$, then $L_{cls}(*)$ is a normal cross entropy loss. u_t evaluates whether the sample is rejected in previous stages. If a sample is rejected in previous stages, it is no longer used for learning the classifier in the current stage. And this sample will not be used for extracting its features in the latter CNN layers. Since we did not constrain the sample to be background for rejection, easy positive samples are also rejected at early stages during training. λ_t is a hyper parameter that controls the weight of loss for each stage of cascaded classifier. We set $\lambda_T = 1$ and $\lambda_t = 0.01/T$ for $t = 1, \ldots T - 1$. Loss is used for $t = 1, \ldots T - 1$ so that the learned classifiers in these stages can learn reasonable classification scores for rejecting background samples. Since the score in the last classifier is used as the final detection score, the classification loss in the last stage has much higher weight than the loss in other stages. For L_{loc}, we use the smoothed L_1 loss in [10]. With this loss function, bounding box regression, chained features and all cascaded classifiers are learned jointly through backpropagation.

4. Experimental results

4.1. Experimental setup

The CC-Net is implemented based on the fast RCNN pipeline. The BN-Net is used as the baseline network if not specified. The CC-Net for BN-Net [14] is shown in Fig. 5. In the CC-Net, the layers belonging to the baseline networks are initialized by these baseline networks pre-trained on the ImageNet dataset. The parameters $\mathbf{a}_t$ in feature chaining and $\mathbf{b}_t$ cascade chaining are initialized as 1. For region proposal, we use the Craft in [37] for ImageNet and the selective search in [31] for VOC 2007 if not specified. The Craft in [37] is an improved approach based on the faster RCNN [25].

We evaluate our method on two public object detection datasets, ImageNet [26] and PASCAL VOC 2007 [6]. Since the ImageNet object detection task contains a sufficiently large number of images and object categories to reach a conclusion, evaluations on component analysis of our training method are conducted on this dataset. This dataset has 200 object categories and consists of three subsets. i.e., train, validation and test data. We follow the same setting in [11] and split the whole validation subset into two subsets, val1 and val2. The network finetuning step uses training samples from train and val1 subsets. The val2 subset is used for evaluating components. All our results are for single model with single-scale training and testing if not specified. Single-stage bounding box regression is used.

4.2. ImageNet results

On this dataset, we compare with the methods tested on the val2 dataset. We compare our framework with several other state-of-art approaches [11, 33, 14, 20, 36, 13, 41]. The mean average precision for these approaches are shown in Table 1. Our work is trained using the provided data of ImageNet. Compared with these approaches, our single model result ranks No. 2. [1]

4.3. PASCAL VOC 2007 results

On this dataset, the VOC07 and VOC12 trainval dataset are optionally used for training and the VOC07 test set is used for evaluation.

When only VOC07 is used for training, as shown in Table 2, the baseline BN-Net+FRCN has mAP 70.8%. Adding our design in chaining features and cascaded classifiers,

[1] The ResNet result with mAP 60.5% in [13] used multi-scale testing, bounding box voting and contextual scores. Without them but with the same region proposal from Craft, the ResNet-152 has mAP 54% and our CC-Net based on BN-Net has mAP 54.5% .

appraoch	RCNN [11]	Berkeley [11]	GoogleNet [33]	DeepID-Net [20]	Superpixel [36]	ResNet [13]	FRCN [10]	GBD-Net [41]	CC-Net
val2(sgl)	31.0	33.4	38. 5	48.2	42.8	60.5	49.4	51.4	54.5

Table 1. Object detection mAP (%) on ImageNet val2 for state-of-the-art approaches with single model. FRCN and CC-Net use the same region proposal [37]. BN-Net is used for FRCN and CC-Net is based on BN-Net.

method	M	N	R	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [10]		V		07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
OHEM [29]		V		07	69.9	71.2	78.3	69.2	57.9	46.5	81.8	79.1	83.2	47.9	76.2	68.9	83.2	80.8	75.8	72.7	39.9	67.5	66.2	75.6	75.9
Ours		V		07	**72.4**	78.3	79.4	69.1	63.5	53.2	82.1	79.7	86.3	56.0	75.6	72.3	83.4	79.0	76.3	76.4	43.1	67.6	71.8	77.3	76.6
FRCN [10]		V		07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
MRCN [9]	✓	V	✓	07+12	74.9	78.7	81.8	76.7	66.6	61.8	81.7	85.3	82.7	57.0	81.9	73.2	84.6	86.0	80.5	74.9	44.9	71.7	69.7	78.7	79.9
OHEM [29]	✓	V		07+12	78.9	80.6	85.7	79.8	69.9	60.8	88.3	87.9	89.6	59.7	85.1	76.5	87.1	87.3	82.4	78.8	53.7	80.5	78.7	84.5	80.7
ION [1]	✓	V		07+12	79.2	80.2	85.2	78.8	70.9	62.6	86.6	86.9	89.8	61.7	86.9	76.5	88.4	87.5	83.4	80.5	52.4	78.1	77.2	86.9	83.5
Ours	✓	V		07+12	**80.4**	83.0	85.8	80.0	73.4	64.6	88.3	88.3	89.2	63.2	86.0	76.8	87.6	88.2	83.4	84.1	54.9	83.7	77.7	86.0	83.6
FRCN [10]		B		07	70.8	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Ours		B	✓	07	**77.1**	77.0	84.2	78.0	66.8	62.6	82.6	85.7	88.4	63.1	85.6	73.1	87.4	86.9	79.9	76.9	50.1	76.1	78.4	79.6	79.5

Table 2. **VOC 2007 test** detection average precision (%). All methods use selective search for region proposal. Training set key: **07**: VOC07 trainval, **07+12**: union of **07** and VOC12 trainval. Multi-region features are not used for our CC-Net for VGG16. Legend: **N**: using VGG16 (V) or BN-Net (B) as the baseline network. **R**: whether the multi-region features are used. **M**: whether multi-step bounding box regression and the multi-scale is used for training and testing.

the mAP is 77.1%. The baseline VGG16+FRCN has mAP 66.9%, our CC-Net based on VGG16 has mAP 72.4%. For VGG16, our CC-Net did not use multi-region features but only use the features at conv3-3, conv4-3 and conv5-3 for chaining. The ION net in [1] also used conv3-3, conv4-3 and conv5-3 features. The IRNN structure for using context and segmentation labels are used in ION but not in our model for VGG. Our CC-Net based on VGG provides 1.2% absolute mAP gain compared with ION based on VGG. We also list the hard example example mining approach in [29] and the multi-region approach in [9] for comparison. The results show that our model performs better.

CC-Net is independent of detection pipeline like fast RCNN or Faster RCNN. When CC-Net is applied for the Faster RCNN, the baseline VGG16+Faster-RCNN has mAP 73.2% when trained on VOC07+12 and tested on VOC07, the VGG16+Faster-RCNN+CC-Net has mAP 80.8% when multi-scale training and multi-step bounding box regression are used. Multi-region features are not used in CC-Net for the results above.

4.4. Component analysis

4.4.1 Results on cascade chaining

In order to evaluate the performance gain from chaining cascaded classifiers, we use the BN-Net as the baseline. Multi-context multi-resolution features are not included. For the results in Table 3, all cascaded classifiers take the output the global_pool layer in BN-Net as the feature. Features and classifiers are jointly learned. Compared with the baseline, adding two extra stages of cascaded classifiers improves the mAP by 1.1%, and adding four extra cascaded classifiers improves mAP by 1.5%. The use of more cascaded classifiers provides better detection accuracy. If the 4 extra stages of cascade do not use chaining, *i.e.* not using previous classification scores for the current classification score, there will be 0.4% mAP drop.

+ 2 cascade stages?		✓		
+ 4 cascade stages?			✓	✓
cascade?		✓	✓	✓
chaining classifier?		✓	✓	
mAP	49.4	50.5	50.9	50.5

Table 3. ImageNet val2 detection mean average precision (%) for baseline BN-Net with different setup on cascade. 'chaining classifier' denotes the result using the chaining for classifier, in which scores in previous stages are used for the current stage. 'cascade' denotes the use of cascade.

4.4.2 Chaining features and classifiers

Table 4 shows the performance for different settings in chaining features and classifiers for VOC07 and ImageNet. The results for VOC07 are trained on VOC07 train+val. The results for ImageNet val2 are trained on ImageNet train+val1. The baseline BN-Net has mAP 49.4%(70.8%) for ImageNet(VOC07). Multi-region features are found to be effective in [9]. When we concatenate features of different contextual regions and resolutions but without the feature chaining or the classification cascade, the mAP is 50.5%(73.8%) for ImageNet(VOC07). This setting has the same depth/width as the CC-Net but does not include classifier or feature chaining. When multi-region features and different classifiers are used in different cascade stages, the network has mAP 73.6% on VOC 2007 if there is no chaining in the cascade.

Based on the multi-region features, cascade chaining provides 0.8%(1.4%) absolute mAP gain for ImageNet(VOC07). Based on the multi-region features features, mAP is 54.5%(77.1%) if both feature chaining and cascade chaining are used in the CC-Net.

In the experimental results, the early cascade stage is used for the results in VOC2007 but not used for the results

multi-region features?		✓	✓	✓
cascade?			✓	✓
classifier chaining?			✓	✓
features chaining?				✓
mAP (VOC 2007)	70.8	73.8	75.2	77.1
mAP (ImageNet)	49.4	50.5	51.3	54.5

Table 4. **VOC 2007 test** and **ImageNet val2** detection mean average precision (%) for baseline BN-Net with different settings on feature chaining and classifier chaining. Early cascade is used for the results on VOC 2007 but not for the results on ImageNet.

in ImageNet val2. This might be the reason on the different improvement in these two datasets for chaining cascaded classifier.

When learning the chaining of features and classifiers, scaling vectors $\mathbf{a}$ and $\mathbf{b}$ are used for controling the scales of features and classification scores, if these scalers are fixed as 1 but not learned, the mAP will drop by 1.7%. No mAP gain is observed when the scaling vector $\mathbf{a}$ for feature chaining with C_1 parameters is replaced by fully connected layer with C_1C_2 parameters.

4.5. Experimental comparison with GBD-Net

Tested on the ImageNet val2, CC-Net has mAP 54.5%. In comparison, the GBD-Net in [41] has mAP 51.4%. CC-Net performs better than GBD-Net by learning more complementary features. GBD-Net aims for context. Our feature chaining aims for learning complementary features and is not constrained to contextual information. For the CNN model in Fig. 3, if all features in the inception modules icp(4e)$_t$, (5a)$_t$ and (5b)$_t$ for $t = 1, 2, 3, 4$ have the same contextual region, GBD-Net has 50.2% mAP while the CC-Net has 52% mAP. This experiment shows that CC-Net learns complementary features even if the contextual regions are the same. But GBD-Net, which aims at passing message among contextual features, does not provide extra gain when the contextual regions is the same.

In GBD-Net [41], when a message is passed from a layer with C_1 channels to another layer with C_2 channels, the computational cost and the number of parameters is proportional to C_1C_2. The computational cost and the number of parameters of which are proportional to C_1 when passing message from a layer to another. Therefore, feature chaining saves the computation and parameter size in message passing by C_2 times. $C_1 = C_2 = 1024$ in the BN-Net. On the other hand, the feature chaining increases the number of parameters when the inception module icp(4e)$_1$ and icp(4e)$_2$ do not share parameters in our implementation but share parameters in the GBD-Net. But the computation and parameter sizes required in icp(4e)- icp(5b) are smaller than that for message passing. Using the same BN-Net as baseline, GBD-Net and CC-Net require 107M and 31M parameters respectively.

4.6. Computational complexity and memory cost

We evaluate the computational complexity using the selective search for the region proposal on Titan X GPU. The training time required for batch size 256 is 0.85 and 1.4 seconds per iteration respectively for CC-Net without cascade and GBD-Net. The training time required for batch size 2048 is 1.15 seconds per iteration for CC-Net with cascade. When the batch size increases by 7 times, the computational time increases only by 0.35 times. For single image with 128 RoIs, the GBD-Net run out of GPU memory on the 12GB GPU. For 256 RoIs, CC-Net without cascade runs out of memory. At the testing stage, the GBD-Net and CC-Net without cascade require 11 and 5 seconds per image respectively. With simpler design in passing messages among features, CC-Net without cascade is faster. The CC-Net with cascade requires 1.6 seconds per image, around 15% of the time required by GBD-Net and 32% the time required by CC-Net without cascade.

Stage	Initial	Early cascade	Contextual cascade
Recall	93.3%	92.0 %	89.8%
BgRoI Rej.	0	915	752

Table 5. The recall rate (%) and number of background RoIs rejected (BgRoI Rej) on VOC07.

The recall rate (%) and number of background RoIs (BgRoI) rejected in different stages are shown in Table 5. Initially there are 1940 background RoIs per image, the early cascade stage rejects 915 RoIs and the contextual cascade further rejects 752 RoIs.

5. Conclusion

In this paper, we present a chained cascade network (CC-Net) for object detection. In this network, the cascaded classifiers in multiple stages are jointly learned through a single end-to-end neural network. This network includes classifier chaining and feature chaining, in which feature and classifier at a stage take the feature and classification scores in previous stages as the prior knowledge. By rejecting easy examples at earlier stages, the features and classifiers learned at latter stages focus more on hard examples for better detection accuracy. After an easy sample is rejected at earlier stage in shallow layer, its computation for deeper or wider layers is not required for faster speed. Experimental results on Pascal VOC and ImageNet for different region proposals show the effectiveness of the CC-Net in improving the detection accuracy.

Acknowledgement This work is supported by SenseTime Group Limited, the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project Nos. CUHK14213616, CUHK14206114, CUHK14205615, CUHK419412, CUHK14203015, CUHK14207814, and CUHK14239816), the Hong Kong Innovation and Technology Support Programme (No.ITS/121/15FX), National Natural Science Foundation of China (No. 61371192), and ONR N00014-15-1-2356.

References

[1] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 1, 2, 3, 7

[2] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, 2005. 2, 5

[3] F. Chollet. Xception: Deep learning with separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016. 2

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2

[5] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Trans. PAMI*, 36(8):1532–1545, 2014. 2

[6] M. Everingham, L. V. Gool, C. K. I.Williams, J.Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 6

[7] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010. 2

[8] P. Felzenszwalb, R. B. Grishick, D.McAllister, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 32:1627–1645, 2010. 2

[9] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *ICCV*, 2015. 1, 2, 3, 7

[10] R. Girshick. Fast r-cnn. In *CVPR*, 2015. 1, 2, 6, 7

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 6, 7

[12] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015. 2

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6, 7

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015. 2, 3, 4, 6

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[16] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015. 2

[17] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Trans. PAMI*, 26(9):1112–1123, 2004. 2

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[19] W. Ouyang, H. Li, X. Zeng, and X. Wang. Learning deep representation with large-scale attributes. In *ICCV*, 2015. 2

[20] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, 2015. 2, 6, 7

[21] W. Ouyang, X. Zeng, and X. Wang. Learning mutual visibility relationship for pedestrian detection with a deep model. *IJCV*, 120(1):14–27, 2016. 2

[22] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, K. Wang, J. Yan, C.-C. Loy, and X. Tang. Deepid-net: Deformable deep convolutional neural networks for object detection. *IEEE Trans. PAMI*, page accepted, 2016. 2

[23] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded cnn for face detection. In *CVPR*, 2016. 2

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 2

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015. 2, 6

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 6

[27] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *CVPR*, 2004. 2

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[29] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 1, 2, 7

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2

[31] A. Smeulders, T. Gevers, N. Sebe, and C. Snoek. Segmentation as selective search for object recognition. In *ICCV*, 2011. 2, 6

[32] K.-K. Sung and T. Poggio. Learning and example selection for object and pattern detection. *MIT A.I. Memo No. 1521*, (1521), 1995. 1, 2

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2, 6, 7

[34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 1

[35] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *ICCV*, 2003. 2, 5

[36] J. Yan, Y. Yu, X. Zhu, Z. Lei, and S. Z. Li. Object detection by labeling superpixels. In *CVPR*, 2015. 6, 7

[37] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Craft objects from images. In *CVPR*, 2016. 2, 6, 7

[38] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 2

[39] X. Zeng, W. Ouyang, and X. Wang. Window-object relationship guided representation learning for generic object detections. *arXiv preprint arXiv:1512.02736*, 2015. 2

[40] X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang, et al. Crafting gbd-net for object detection. *TPAMI*, 2017 (accepted). 2

[41] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang. Gated bi-directional cnn for object detection. In *ECCV*, 2016. 1, 2, 3, 6, 7, 8