A. Object Detection Baselines

In this section we introduce our detection method based on the baseline Faster R-CNN [6] system. The models are initialized by the ImageNet classification models, and then fine-tuned on the object detection data. We have experimented with ResNet-50/101 at the time of the ILSVRC & COCO 2015 detection competitions.

Unlike VGG-16 used in [6], our ResNet has no hidden fc layers. We adopt the idea of “Networks on Conv feature maps” (NoC) [7] to address this issue. We compute the full-image shared conv feature maps using those layers whose strides on the image are no greater than 16 pixels (i.e., conv1, conv2_x, conv3_x, and conv4_x, totally 91 conv layers in ResNet-101). We consider these layers as analogous to the 13 conv layers in VGG-16, and by doing so, both ResNet and VGG-16 have conv feature maps of the same total stride (16 pixels). These layers are shared by a region proposal network (RPN, generating 300 proposals) [6] and a Fast R-CNN detection network [2]. RoI pooling [2] is performed before conv5_1. On this RoI-pooled feature, all layers of conv5_x and up are adopted for each region, playing the roles of VGG-16’s fc layers. The final classification layer is replaced by two sibling layers (classification and box regression [2]).

For the usage of BN layers, after pre-training, we compute the BN statistics (means and variances) for each layer on the ImageNet training set. Then the BN layers are fixed during fine-tuning for object detection. As such, the BN layers become linear activations with constant offsets and scales, and BN statistics are not updated by fine-tuning. We fix the BN layers mainly for reducing memory consumption in Faster R-CNN training.

**PASCAL VOC**

Following [2, 6], for the PASCAL VOC 2007 test set, we use the 5k trainval images in VOC 2007 and 16k trainval images in VOC 2012 for training (“07+12”). For the PASCAL VOC 2012 test set, we use the 10k trainval+test images in VOC 2007 and 16k trainval images in VOC 2012 training data. Table 1 shows the results. ResNet-101 improves the mAP by >3% over VGG-16. This gain is solely because of the improved features learned by ResNet.

**MS COCO**

The MS COCO dataset [5] involves 80 object categories. We evaluate the PASCAL VOC metric (mAP @ IoU = 0.5) and the standard COCO metric (mAP @ IoU = .5:0.95). We use the 80k images on the train set for training and the 40k images on the val set for evaluation. Our detection system for COCO is similar to that for PASCAL VOC. We train the COCO models with an 8-GPU implementation, and thus the RPN step has a mini-batch size of 8 images (i.e., 1 per GPU) and the Fast R-CNN step has a mini-batch size of 16 images. The RPN step and Fast R-CNN step are both trained for 240k iterations with a learning rate of 0.001 and then for 80k iterations with 0.0001.

Table 2 shows the results on the MS COCO validation set. ResNet-101 has a 6% increase of mAP@[.5,.95] over VGG-16, which is a 28% relative improvement, solely contributed by the features learned by the better network. Remarkably, the mAP@[.5,.95]’s absolute increase (6.0%) is nearly as big as mAP@[.5]’s (6.9%). This suggests that a deeper network can improve both recognition and localiza-
### B. Object Detection Improvements

For completeness, we report the improvements made for the competitions. These improvements are based on deep features and thus should benefit from residual learning.

#### MS COCO

**Box refinement.** Our box refinement partially follows the iterative localization in [1]. In Faster R-CNN, the final output is a regressed box that is different from its proposal box. So for inference, we pool a new feature from the regressed box and obtain a new classification score and a new regressed box. We combine these 300 new predictions with the original 300 predictions. Non-maximum suppression (NMS) is applied on the union set of predicted boxes using an IoU threshold of 0.3 [3], followed by box voting [1]. Box refinement improves mAP by about 2 points (Table 3).

**Global context.** We combine global context in the Fast R-CNN step. Given the full-image conv feature map, we pool a feature by global Spatial Pyramid Pooling [4] which can be implemented as “RoI” pooling using the entire image’s bounding box as the RoI. This pooled feature is fed into the post-RoI layers to obtain a global context feature. This global feature is concatenated with the original per-region feature, followed by the sibling classification and box regression layers. This new structure is trained end-to-end. Global context improves mAP@.5 by about 1 point (Table 3).

**Multi-scale testing.** In the above, all results are obtained by single-scale training/testing as in [6], where the image’s shorter side is \( s = 600 \) pixels. Multi-scale training/testing has been developed in [4, 2] by selecting a scale from a feature pyramid, and in [7] by using maxout layers. In our current implementation, we have performed multi-scale testing following [7]; we have not performed multi-scale training because of limited time. In addition, we have performed multi-scale testing only for the Fast R-CNN step (but not yet for the RPN step). With a trained model, we compute conv feature maps on an image pyramid, where the image’s shorter sides are \( s \in \{ 200, 400, 600, 800, 1000 \} \). We select two adjacent scales from the pyramid following [7]. RoI pooling and subsequent layers are performed on the feature maps of these two scales [7], which are merged by maxout as in [7]. Multi-scale testing improves the mAP by over 2 points (Table 3).

**Using validation data.** Next we use the 80k+40k trainval set for training and the 20k test-dev set for evaluation. The test-dev set has no publicly available ground truth and the result is reported by the evaluation server. Under this setting, the results are an mAP@.5 of 55.7% and an mAP@[.5, .95] of 34.9% (Table 3). This is our single-model result.
<table>
<thead>
<tr>
<th>method</th>
<th>LOC network</th>
<th>testing</th>
<th>LOC error on GT CLS</th>
<th>classification network</th>
<th>top-5 LOC error on predicted CLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>our single model</td>
<td>ResNet-101</td>
<td>1-crop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ILSVRC'15)</td>
<td>ResNet-101</td>
<td>dense</td>
<td>11.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ours (ILSVRC'15)</td>
<td>ResNet-101</td>
<td>dense</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Localization error (%) on the ImageNet validation. In the column of “LOC error on GT class” ([10]), the ground truth class is used. In the “testing” column, “1-crop” denotes testing on a center crop of 224×224 pixels, “dense” denotes dense (fully convolutional) and multi-scale testing.

<table>
<thead>
<tr>
<th>method</th>
<th>top-5 localization err</th>
</tr>
</thead>
<tbody>
<tr>
<td>OverFeat [9]</td>
<td>30.0</td>
</tr>
<tr>
<td>VGG [10]</td>
<td>25.3</td>
</tr>
<tr>
<td>ours (ILSVRC'15)</td>
<td>8.9</td>
</tr>
</tbody>
</table>

Table 8. Comparisons of localization error (%) on the ImageNet dataset with state-of-the-art methods.

based on the predicted classes. We adopt the “per-class regression” (PCR) strategy [9, 10], learning a bounding box regressor for each class. We pre-train the networks for ImageNet classification and then fine-tune them for localization. We train networks on the provided 1000-class ImageNet training set.

Our localization algorithm is based on the RPN framework of [6] with a few modifications. Unlike the way in [6] that is category-agnostic, our RPN for localization is designed in a per-class form. This RPN ends with two sibling 1 × 1 convolutional layers for binary classification (cls) and box regression (reg), as in [6]. The cls and reg layers are both in a per-class form, in contrast to [6]. Specifically, the cls layer has a 1000-d output, and each dimension is binary logistic regression for predicting being or not being an object class; the reg layer has a 1000×4-d output consisting of box regressors for 1000 classes. As in [6], our bounding box regression is with reference to multiple translation-invariant “anchor” boxes at each position.

As in our ImageNet classification training, we randomly sample 224×224 crops for data augmentation. We use a mini-batch size of 256 images for fine-tuning. To avoid negative samples being dominate, 8 anchors are randomly sampled for each image, where the sampled positive and negative anchors have a ratio of 1:1 [6]. For testing, the network is applied on the image fully-convolutionally.

Table 7 compares the localization results. Following [10], we first perform “oracle” testing using the ground truth class as the classification prediction. VGG’s paper [10] reports a center-crop error of 33.1% (Table 7) using ground truth classes. Under the same setting, our RPN method using ResNet-101 net significantly reduces the center-crop er-

PASCAL VOC
We revisit the PASCAL VOC dataset based on the above model. With the single model on the COCO dataset (55.7% mAP@.5 in Table 3), we fine-tune this model on the PASCAL VOC sets. The improvements of box refinement, context, and multi-scale testing are also adopted. By doing so we achieve 85.6% mAP on PASCAL VOC 2007 (Table 4) and 83.8% on PASCAL VOC 2012 (Table 5)1. The result on PASCAL VOC 2012 is 10 points higher than the previous state-of-the-art result [1].

ImageNet Detection
The ImageNet Detection (DET) task involves 200 object categories. The accuracy is evaluated by mAP@.5. Our object detection algorithm for ImageNet DET is the same as that for MS COCO in Table 3. The networks are pre-trained on the 1000-class ImageNet classification set, and are fine-tuned on the DET data. We split the validation set into two parts (val1/val2) following [3]. We fine-tune the detection models using the DET training set and the val1 set. The val2 set is used for validation. We do not use other ILSVRC 2015 data. Our single model with ResNet-101 has 58.8% mAP and our ensemble of 3 models has 62.1% mAP on the DET test set (Table 6). This result won the 1st place in the ImageNet detection task in COCO 2015.

C. ImageNet Localization
The ImageNet Localization (LOC) task [8] requires to classify and localize the objects. Following [9, 10], we assume that the image-level classifiers are first adopted for predicting the class labels of an image, and the localization algorithm only accounts for predicting bounding boxes.
ror to 13.3%. This comparison demonstrates the excellent performance of our framework. With dense (fully convolutional) and multi-scale testing, our ResNet-101 has an error of 11.7% using ground truth classes. Using ResNet-101 for predicting classes (4.6% top-5 classification error), the top-5 localization error is 14.4%.

The above results are only based on the proposal network (RPN) in Faster R-CNN [6]. One may use the detection network (Fast R-CNN [2]) in Faster R-CNN to improve the results. But we notice that on this dataset, one image usually contains a single dominate object, and the proposal regions highly overlap with each other and thus have very similar RoI-pooled features. As a result, the image-centric training of Fast R-CNN [2] generates samples of small variations, which may not be desired for stochastic training. Motivated by this, in our current experiment we use the original R-CNN [3] that is RoI-centric, in place of Fast R-CNN.

Our R-CNN implementation is as follows. We apply the per-class RPN trained as above on the training images to predict bounding boxes for the ground truth class. These predicted boxes play a role of class-dependent proposals. For each training image, the highest scored 200 proposals are extracted as training samples to train an R-CNN classifier. The image region is cropped from a proposal, warped to 224×224 pixels, and fed into the classification network as in R-CNN [3]. The outputs of this network consist of two sibling fc layers for cls and reg, also in a per-class form. This R-CNN network is fine-tuned on the training set using a mini-batch size of 256 in the RoI-centric fashion. For testing, the RPN generates the highest scored 200 proposals for each predicted class, and the R-CNN network is used to update these proposals’ scores and box positions.

This method reduces the top-5 localization error to 10.6% (Table 7). This is our single-model result on the validation set. Using an ensemble of networks for both classification and localization, we achieve a top-5 localization error of 9.0% on the test set. This number significantly outperforms the ILSVRC 14 results (Table 8), showing a 64% relative reduction of error. This result won the 1st place in the ImageNet localization task in ILSVRC 2015.

References