Summer 9-28-2017

Refining Bounding-Box Regression for Object Localization

Naomi Lynn Dickerson
Portland State University

Refining Bounding-Box Regression for Object Localization

by

Naomi Lynn Dickerson

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Thesis Committee:
Melanie Mitchell, Chair
Feng Liu
Bart Massey

Portland State University
2017

Abstract

For the last several years, convolutional neural network (CNN) based object detection systems have used a regression technique to predict improved object bounding boxes based on an initial proposal using low-level image features extracted from the CNN. In spite of its prevalence, there is little critical analysis of bounding-box regression or in-depth performance evaluation. This thesis surveys an array of techniques and parameter settings in order to further optimize bounding-box regression and provide guidance for its implementation. I refute a claim regarding training procedure, and demonstrate the effectiveness of using principal component analysis to handle unwieldy numbers of features produced by very deep CNNs.

Acknowledgements

Special thanks to my advisor, Melanie Mitchell, for her support and making this work possible.

Many thanks to Portland State's New Beginnings program and its faculty, without which I never would have pursued this degree. I have built everything on the foundations I learned from Bart Massey, Bryant York and Simon Niklaus and am especially grateful for the mentorship of Jonathan Walpole and Mark Morrissey.

Thank you so much to my partner and family for always supporting me and going above and beyond. Everything I do is better because of your love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

State-of-the-art object detection employs a suite of techniques and strategies in pursuit of two fundamental questions about an image: what salient objects are in it, and where are they within the image? The task is seemingly effortless for most humans, but poses a perennial challenge for computer vision. Results from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1] show that machines have finally surpassed humans at the classification task—choosing an appropriate label for an image from a set of of 1000 diverse categories—but that only describes what is in the image, not where it is. Potential applications such as autonomous driving and robotics further demand precise object localization, which is a more challenging and complex task to solve [2].

The localization task introduces two separate but closely related challenges. First, it requires finding a number of candidate objects within the image, generally referred to as proposals. The classifier can only select one label for any given input image, but an image might contain any number of

significant objects, so it is necessary to generate enough proposals to capture all the potential objects in an image. Second, these proposals provide coarse localization and must be refined. Consider that the most basic mechanism for generating proposals is a sliding-window approach that creates a large number of candidates by methodically sampling regions at a variety of scales and aspect ratios. There is no guarantee or likelihood that any of these will precisely match a target object's dimensions, even when thousands of proposals are created in this manner.

This is the motivation for bounding-box regression (BBR), a technique for fine-grained localization developed by Girshick et al. for R-CNN, a detector based on "regions with CNN (convolutional neural net) features" [3]. Figure 1.1 outlines the basic architecture of their system: a region proposal method such as selective search [4] or the sliding-window approach mentioned above, a CNN trained for the classification task, and a set of class-specific bounding-box regressors trained on image features extracted by the CNN (not pictured). Furthermore, Hoiem et al. show that poor localization is the dominant error-mode for CNN-based object detectors [5]. Bounding-box regression is a simple, computationally inexpensive technique applied after a proposal has been classified that can compensate for a CNN's potential weakness at localization.

An object is considered correctly localized if its proposal bounding-box sufficiently overlaps with ground truth—a human-labeled bounding-box thought best to encapsulate the target object. Accuracy is measured by intersection

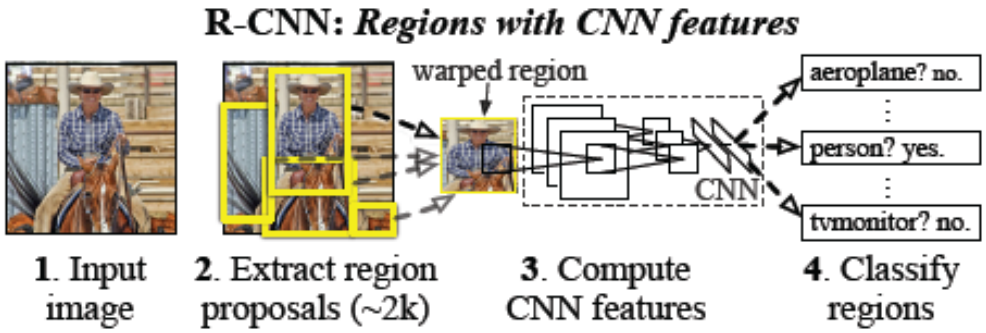

Figure 1.1: R-CNN architecture from [3]. Figure best viewed in color.

over union (IOU) of the two bounding-boxes, and a successful localization must be above a certain threshold, generally set to 0.5 [6] [7]. Anything less is a mislocalization error. Figure 1.2 shows overlaps at a variety of IOUs on test data, demonstrating what counts as a successful localization. The leftmost image shows that it is possible to capture a significant and recognizable part of the object, in this case the dog's head, and still fail to successfully localize it.

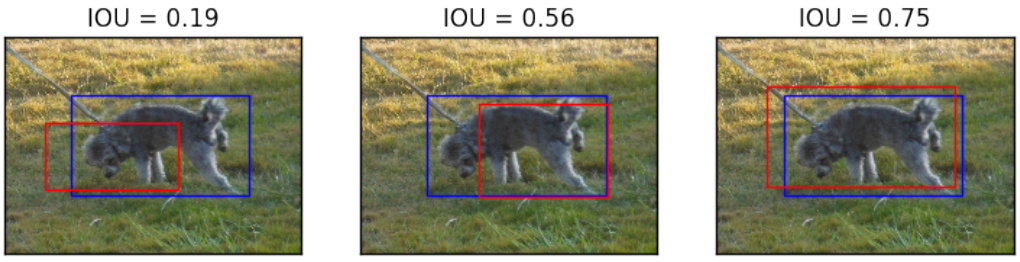

Figure 1.2: An example of IOU differences using actual results from my tests on the dataset from [8]. The blue rectangle represents the ground truth for the dog, and the red rectangle shows a sample detection with the given IOU. The leftmost image shows a mislocalization, the other two show acceptable localizations. Figure best viewed in color.

BBR learns a transformation that attempts to map a proposal bounding-box's coordinates to the ground truth values using image features extracted by the CNN. That is to say, it uses complex image features to adjust the bounding-box and better capture the target. Along with pushing proposals above the localization threshold, BBR can reduce the frequency of multiple detections for the same object, each of which counts as a false positive during performance evaluation. Figure 1.3 illustrates some of the problems endemic to object detection. The top-left image shows a single armadillo that has been classified correctly but incorrectly detected twice. When two proposals with the same class overlap one another with IOU above a certain threshold, non-maximal suppression is used to keep only the highest scoring proposal. In this case, a smaller threshold could prevent the redundant detection, but the top-right image illustrates the ambiguity in selecting such a threshold—several cars share significant overlap within the frame, so suppressing those overlapping bounding-boxes would prevent a correct detection.

On the other hand, rather than try to modify the threshold, BBR can assure that boxes move closer to the ground truth, causing multiple detections to overlap at a higher IOU and be more easily discarded. In the case of the dogs in the bottom-right image, one detection captures only the left dog's head, while the other is too large. Successfully applying BBR would result in the head bounding-box expanding and the other tightening, allowing both to be automatically recognized as detections of the same target.

Bounding-box regression provides a cheap, adaptable method to solve

Figure 1.3: Detection results from R-CNN [3] on ILSVRC. Labels show predicted class and detection score. Figure best viewed in color.

multiple problems arising from poor localization. It has been used in several different object detectors; however the literature suggests that BBR has not been evaluated in detail, and recent implementations use many of the same settings as those initially proposed by Girshick et al. [3]. Given the simplicity of the method and the continuing popularity of CNN-based object detectors,

I assert that BBR warrants a detailed study in its own right. This thesis attempts to validate the original findings from [3] regarding optimal image preprocessing, training, and regularization. In addition, recent implementations of BBR use features from different CNN layers—I examine whether the feature layer has any effect on performance, and investigate principal component analysis (PCA) as an alternative to ridge regression for handling the large feature dimensions found in some of the CNN layers.

Chapter 2

Background and Previous Work

2.1 R-CNN and Its Successors

My work is based on the bounding-box regression method developed by Girshick et al. for R-CNN—an architecture combining bottom-up region proposals with a convolutional neural network to achieve state-of-the-art performance in object detection [3]. Although CNNs have been around for decades, prior to R-CNN most visual recognition tasks were performed using SIFT [9] and HOG [10] to generate features for images. These methods can be computationally very fast and are still used for many tasks, but rely on hand-tuned filter banks and produce only a top layer of features for the image. CNNs, on the other hand, are able to learn their own filters and further convolve and pool stacked layers to generate deep, non-linear features. Krishevsky et al. demonstrated significant performance gains on the ImageNet classification task [7] using a CNN in 2012 [11], and afterward R-CNN showed how to translate the application to the more complicated object detection task.

The results from [3] also indicated that the rich features generated by a

CNN could be used for the localization task, via BBR, as well as for classification without retraining the CNN for a separate task. Overall, they found that most of the features learned in the convolutional layers were general purpose, whereas the final fully-connected layers responded most highly to domain-specific fine-tuning. Thus, they were able to take a CNN pre-trained on the large ImageNet dataset and tune it for the much smaller, 20-class PASCAL VOC challenge [6] with minimal modification. The other implication of this is that the convolutional layers are sufficiently general purpose to produce quality features for BBR, although they were not trained specifically to detect object bounds.

Over the last four years, R-CNN has been fine-tuned, sped-up, and tweaked into numerous evolutions, but it remains the base for many top-performing object detection models. Its two direct successors, Fast R-CNN and Faster R-CNN, both increase the efficiency of the system and train the CNN using multi-task loss [12] [13]. In both cases, the network is trained jointly for classification and bounding-box regression. This increases the CNNs ability to score better-localized proposals more highly, and thus prioritize quality localization. Nonetheless, results from Fast R-CNN show that adding BBR increases performance further, even when the CNN has been trained for localization.

Compare Fast R-CNN's architecture in Figure 2.1 to the original R-CNN in Figure 1.1. BBR has been explicitly built-in, and shared features speed up computation time. Yet, it makes use of the same basic structure and process.

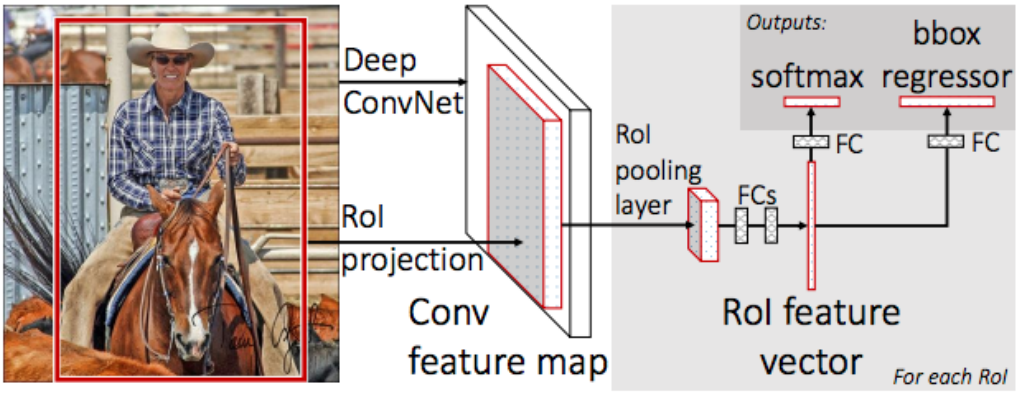

Figure 2.1: Fast R-CNN Architecture [12]. An input image and multiple proposals, labeled here as regions of interest (RoIs), are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per proposal: softmax probabilities over object classes and per-class bounding-box regression offsets.

In order to simplify my experiments and make the results as widely applicable as possible, I use a stripped down architecture based on R-CNN. I simulate the region proposal module by creating randomly jittered bounding-boxes from an object's ground truth, use a deep CNN pre-trained on the ImageNet dataset that is not fine-tuned for localization, and attach an independent BBR module.

2.2 VGG-16 Architecture

I use VGG-16, a deep CNN used in both Fast and Faster R-CNN, for all of my experiments. It was first proposed by Simonyan et al. and was runner-up in the 2014 ImageNet Challenge [14], showing that small convolutional

filters and increased network depth can significantly improve performance. Although there are more advanced CNN designs at the point of writing, I use VGG-16 because there are several easily accessible pre-trained models available and it has been used prevalently enough that there are robust benchmarks for comparison. In addition, VGG-16 is directly based on the AlexNet CNN architecture used in R-CNN [11]. It seems appropriate to use the descendants of the initial models in order to generate results that are applicable and relevant in my experiments.

The network has been pre-trained on ImageNet's 1000-class dataset. In order to compute features from an object proposal's image region, it must first be converted to the 224×224 input size required by the CNN. There are a number of possibilities for transforming the image region, including scaling then center-cropping and padding out to a square aspect ratio then scaling; I use a simple affine warp, as was used in [3], scaling the proposal to the correct dimensions regardless of aspect ratio. In addition, R-CNN adds enough context padding to the proposal such that the warped result has exactly 16 pixels of padding on all sides, which I replicate in my setup and refer to as proportional padding. I also test BBR performance with constant and no context padding for reference.

VGG-16 has 16 weighted layers and 5 max pooling layers, visible in Figure 2.2. Convolutional filters are learned via mini-batch gradient descent, and several layers of 3×3 filters followed by ReLU non-linearity precede each max pooling operation (see [11] for a description of ReLu). Network layers

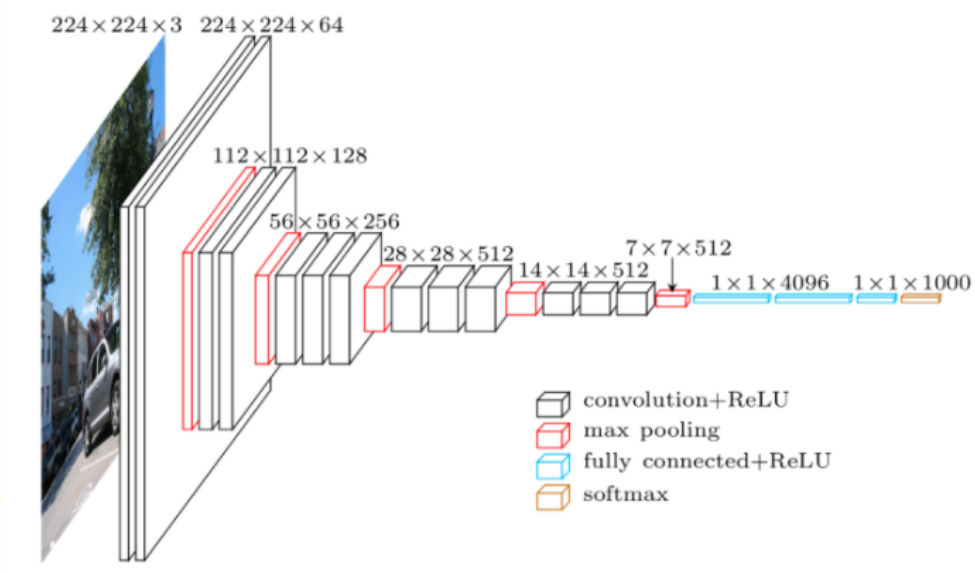

Figure 2.2: VGG-16 Architecture [14]. Figure best viewed in color.

are named by type and stacked position, such that pool_5 refers to the fifth max pooling layer, fc_6 is the fully connected layer directly following pool_5, and conv_{5_2} is the second convolutional layer in the stack directly preceding pool_5.

2.3 Bounding-Box Regression

As opposed to classification, which outputs a discrete class label, linear regression predicts a continuous real number value from a given feature vector. It uses ordinary least squares as a loss function to predict weight coefficients for each feature. It has the advantage of being mathematically simple and computationally cheap, but is prone to over-fitting data sets with large num-

bers of features. BBR uses ridge regression to counteract this tendency to overfit, which shrinks the weight coefficients by a regularization penalty, λ. The greater λ, the more high-value coefficients are penalized and pushed toward zero.

The first instance of BBR was used in the deformable part models [15] and was later adapted for R-CNN [3]. Each proposal is first scored and classified by the CNN, then a class-specific regressor is applied to predict a refined bounding-box. The regression relies on image features computed by the CNN for the proposal. The model of VGG-16 I use was not trained with regards to localization accuracy, so it is likely that the features it produces are not optimal for object bound detection; however, in practice the computed features are sufficient to produce significant localization improvement.

Training a model for BBR attempts to learn a transformation that maps a proposal box to the ground truth box, where P is the proposal, G is the ground truth, and $\hat{G}$ is the predicted bounding-box, which I refer to as the prediction. All equations come from [3] unless otherwise stated. Each box is specified as a set of top-left coordinates and a width and height in pixels, where $P = (P_x, P_y, P_w, P_h)$. Four learnable functions, $d_*(P)$, transform P

into $\hat{G}$:

$$\hat{G}_x = P_w d_x(P) + P_x \tag{2.1}$$

$$\hat{G}_y = P_h d_y(P) + P_y \tag{2.2}$$

$$\hat{G}_w = P_w \exp(d_w(P)) \tag{2.3}$$

$$\hat{G}_h = P_h \exp(d_h(P)) \tag{2.4}$$

Here, $d_x(P)$ and $d_y(P)$ are scale-invariant translations of the coordinates (x, y). As shown in equations 2.1 and 2.2, $d_x(P)$ and $d_y(P)$ learn an adjustment factor which is then multiplied by the proposal's width and height, respectively. $d_w(P)$ and $d_h(P)$ are log-space translations of width and height, which produce better results on the wide range of bounding-box dimensions.

The set of functions $d_*(P) = w_*^T \phi(P)$, where $\phi(P)$ is the feature vector for proposal P, and w_* are the weight coefficients learned by ridge regression for each function $d_*(P)$. In R-CNN, ϕ_{pool_5} is used, whereas later iterations use ϕ_{FC_6} or ϕ_{FC_7}. There is no discussion of this decision in the literature, but it is likely because, in the deeper networks such as VGG-16, the size of the pool_5 feature vector is 25k, an order of magnitude larger than the fully-connected layers.

In my experiments, I pre-compute features for all proposals before applying BBR; thus, the regression can be solved efficiently in the standard regularized least squares closed form [16]:

$$w_* = (\Phi^T \Phi + \lambda(I))^{-1} \Phi^T Y_* \tag{2.5}$$

In Equation 2.5, Φ is the $n \times m$ matrix of features, where n is the number of training examples and m is the number of features (ϕ). Recall that λ is the regularization constant. Y_* is a $n \times 1$ matrix of regression targets, t_* for each training pair (P, G):

$$t_x = (G_x - P_x)/P_w \tag{2.6}$$

$$t_y = (G_y - P_y)/P_h \tag{2.7}$$

$$t_w = \log(G_w/P_w) \tag{2.8}$$

$$t_h = \log(G_h/P_h) \tag{2.9}$$

Girshick et al. found that high regularization was necessary for good results and set $\lambda = 1000$, counteracting a strong tendency for this model to overfit. They also found that it was important to carefully select the training set such that P had significant overlap with G, and only used training pairs (P, G) in which the IOU of the two was greater than a threshold of 0.6. BBR is only performed on classified proposals; it makes no sense to attempt to localize a non-object for which the ground-truth is undefined.

2.4 Performance of Bounding-Box Regression on PASCAL VOC

The PASCAL Visual Object Challenge [6] was the canonical object detection competition for several years, and its datasets and benchmarks are still used to compare detector performance. R-CNN and its successors, as well as SPPnet [17] among others, report results on the PASCAL VOC detection challenge for networks both with and without BBR. The standard evaluation metric for the object detection task is mean average precision (mAP), which takes the mean of average precision (AP) across 20 object classes. AP for PASCAL VOC classes is calculated by averaging the precision at 11 discrete points in a single class precision-recall curve.

The previous results on PASCAL VOC show that in all cases, BBR is able to improve mAP by 2-4 percentage points [3] [12] [13]. In many cases, this is as much a margin as the state-of-the-art model has over its competitors, so it is a significant difference. Girshick et al. further divide their results into performance on all 20 PASCAL VOC classes, so it is possible to see that BBR improves the average precision of some classes far more than others. For example, BBR increases the AP of detecting trains by 8.4 points, from 52.8% to 61.2%, but only increases detection of chairs by 0.9 points. It should be noted that regardless of magnitude, BBR has a positive effect on all classes.

Chapter 3

Methodology

In this chapter, I outline my experimental setup and describe in detail the dataset used in all my experiments, the method I use for generating object proposals, and the performance metrics I use to evaluate the results. One overarching design decision is to separate BBR as much as possible from any specific system or architecture; this allows me to examine BBR on its own and keeps the computational costs of running a wide variety experiments low.

The most complete and extensive discussion of BBR occurs in Girschick et al. [3], thus their analysis and results serve as the baseline for my experiments. Where possible, I compare my findings to theirs, as well as note any cases where our methodology differs.

3.1 Experimental Setup

I examine the effects of training set IOU distribution, context padding, CNN feature layer, PCA, and object class on BBR performance, in that order. I assume that the results of these experiments are independent, and in each

subsequent experiment use the best parameters from the previous. That is to say, I assume that if proportional context padding has the best performance of all the padding methods using fc_6 features, it will also have the best performance on $pool_5$ features.

I divide the data into a training set (3000 proposals), validation set (1000 proposals), and test set (1000 proposals). The validation set is used in each experiment to determine the best hyper-parameters—λ for ridge regression and number of components for PCA. Each dataset is generated using the method described in Section 3.3, and new training, validation, and test sets are created for each experiment to avoid overfitting to a single test set.

In all experiments I use a pre-trained VGG-16 model written in Tensorflow [18] [19] to compute image features. This model was trained on the ImageNet dataset and I perform no additional fine-tuning. Its only role in my system is to produce image features; I discard all of its classification scores. I assume each proposal is correctly classified and apply the appropriate class-specific regressor. In real use cases, BBR is applied to misclassified proposals. However, there is no way to localize an incorrectly classified object—the problem inherently does not make sense. For all but the last experiment, I use the same object class: in particular, I train the regressor to localize dogs within images.

3.2 Portland Dog-Walking Dataset

I use The Portland Dog-Walking Images dataset described in [8] for training and testing in all of my experiments. This dataset is by and for the Mitchell Research Group, and I use it here as part of that research corpus. Each of the 500 images contains a labeled dog, dog-walker, and leash with human-drawn bounding box; additional pedestrians, cars, buildings, and objects may be present, but are not labeled. For my initial experiments, I only use the dog class, but later obtain results for both people and leashes. Dogs are present in both the ImageNet and PASCAL VOC datasets, and have almost exactly the same average improvement as a class (3.4 percentage points) from BBR in [3], thus are a natural choice for baseline testing. These images are split into a training set of 300 and validation and test sets of 100 images each.

3.3 Generating Proposals for Training and Test Sets

Rather than rely on selective search or another specific algorithm to generate region proposals, I simplify the process by randomly jittering the ground truth bounding box to create a proposal box. This method allows me to easily specify the quality-level of proposals, i.e., how close to ground truth proposals tend to be, and the spread of proposal IOUs.

My algorithm generates proposals using the same set of parameters for all experiments, unless otherwise specified, to ensure that results are comparable among experiments. For each dimension (x, y, w, h), I generate a random

amount of skew drawn from a normal distribution with a standard deviation of one-quarter width or height, depending on the dimension. Any proposals below a minimum IOU threshold are discarded; for the test sets, the minimum IOU is 0.05 and the median IOU is 0.43. Proposals tend toward a similar aspect ratio, but Figure 3.1 shows the wide variety of proposals generated from a single ground truth box.

Figure 3.1: Automatically generated proposals from one of the test sets, showing a variety of scales and aspect ratios. Figure best viewed in color.

Hypothetically, it is possible to generate an arbitrarily large quantity of training data this way. There are no studies to indicate at what point oversaturation would occur and the benefits of increased sample size would be outweighed or negated by overfitting to the specific image batch, so I opt

for a very conservative estimate and generate 10 samples per ground truth 'dog' box.

3.4 Measuring Performance

Recall that successfully localizing an object requires finding a bounding box that overlaps at least 0.5 IOU with ground truth. The percentage of initially low IOU proposals that BBR is able to localize is in some ways the most relevant performance measurement, since being able to increase an IOU from 0.2 to 0.4 ultimately has no effect on the localization performance of a system, nor does increasing the IOU of an already-localized proposal. However, localization improvement is sensitive to the exact distribution of proposal IOUs, e.g., if there are more proposals in the 0.4-0.5 IOU range for a given test set, a model with low mean improvement (prediction IOU - proposal IOU) may still achieve a high localization percentage. Thus, I also track mean improvement and mean relative improvement (MRI).

Given that P_{low} is the set of proposals where $\text{IOU}(P_i, G_i) < 0.5$, localization percentage is the percentage of P_{low} where for a proposal in the set, P_i, its BBR prediction $\hat{G}_i$ has $\text{IOU}(\hat{G}_i, G_i) \geq 0.5$. Similarly, I measure mislocalization as the percentage of proposals in the set P_{high} where $\text{IOU}(\hat{G}_i, G_i) < 0.5$. Mislocalization can be characterized as the error measurement for BBR. MRI is calculated as the mean of $(IOU(\hat{G}_i, G_i) - IOU(P_i, G_i))/IOU(P_i, G_i)$, which rewards improving low IOU proposals more highly. It can also serve as an indicator of whether two models improve proposals similarly, or have different

performance depending on proposal IOU.

Chapter 4

Experiments and Results

In this chapter I describe the details and results of my five experiments: the effect of training set IOU distribution, context padding, CNN feature layer, PCA, and object class on BBR performance. The first two experiments validate claims made in [3] about BBR performance. In the CNN feature layer experiment, I notice that different systems extract features from different layers and test whether this has any effect on BBR performance. Then, I examine PCA as an alternative for regularization. Finally, I look at the other two object classes present in the Portland Dog-Walking dataset: people (dog-walkers) and leashes.

4.1 Effect of Training Set IOU Distribution

Girshick et al. claim that it is important to only use training examples with a sufficiently high overlap with ground truth, noting that the problem of transforming a far away box doesn't make sense. R-CNN implemented this by only using proposals with ground truth overlap of IOU ≥ 0.6 in the

training set. However, I posit that attempting to improve a low IOU proposal makes little sense when the regressor has only seen high IOU examples, and improving low IOU bounding boxes should be higher priority than improving those which have already been successfully localized.

In order to test the claim made in [3], I create six different training sets—all are derived from the same 300 training images, but each has a different minimum ground truth IOU cutoff threshold and independently generated proposals. The goal of this experiment is threefold: (1) Validate or refute the claims made in [3] regarding the optimal range of training proposal IOUs. (2) Investigate what the optimal training data looks like, and if different use cases call for different training methods. (3) Determine whether or not the optimal training distribution is dependent on the test set distribution. Is there really some magic cutoff point that produces the best results, or will each system need to optimize BBR depending on the average quality of proposals it tends to produce? And, if the latter is true, is it possible to predict the best-performing parameters if we know something about the object-proposal module?

I use features from fc_6 and 16px proportional padding (recommended in [3]). The test set always contains a full range of proposal IOUs distributed between 0.05 and 1.0. All results from this section use individually tuned λ. Early experiments suggested that performance continued to improve with λ well above the value of 1000 given in [3], so I ran BBR on the validation set with a range of $0.1 \leq \lambda \leq 18000$.

The literature's recommended 0.6 IOU threshold for training data not only under-performs other training set distributions in most cases, but the optimum threshold varies by usage—namely, the quality of proposals produced by the object detection algorithm. Surprisingly, optimum λ was between 8000-18000 for each IOU threshold. Although heavy regularization makes sense given the fact that training set is smaller than the number of features (3000 vs 4096), such a high value is rarely seen in practice for ridge regression.

IOU Thresh.	Mean Improvement	MRI	% Localized	% Mislocalized
0.05	0.113	0.469	56.6	30.7
0.2	0.144	0.513	59.6	14.7
0.3	0.147	0.477	59.0	9.4
0.4	0.145	0.450	55.4	6.6
0.5	0.138	0.402	52.4	3.6
0.6	0.126	0.350	46.3	1.9

Table 4.1: Effect of training set IOU threshold on performance on test set.

A summary of results on the effect of IOU thresholding is given in Table 4.1. This table demonstrates that the idea of training the regressor only on samples that are sufficiently close to the ground truth is helpful, but 0.6 is not necessarily a magic number. All thresholded groups outperform the 0.05 group, which from here out I will refer to as the full-range training set. However, the regressor trained on samples above 0.6 IOU performs the second worst in all but one category: percentage of test samples mislocalized.

The relationship between training IOU, performance (percent localized), and error (percent mislocalized) can be seen in Figure 4.1. Localization peaks at the 0.2 threshold, and both localization and mislocalization decrease as threshold increases. Thus, higher training thresholds have fewer errors but have less ability to improve low IOU proposals. These results make sense; a BBR model trained to improve high IOU proposals will learn lower magnitude transformations (since the greatest possible IOU improvement it must learn during training is 0.4), but will be trained specifically on high IOU proposals and thus be less likely to mislocalize them.

If it were possible to apply BBR only to low IOU proposals, which would require an omniscience that never exists in practice, the models trained on a larger range of IOUs would have an even greater performance advantage. Such idle speculation is rarely useful, except that in this case it reveals an important point: if we have some prior knowledge about the quality of proposals the object detector outputs, we can choose an appropriate training regime. For example, if the region proposal algorithm uses a sliding-window approach and most proposal IOUs are low, it makes sense to use a BBR model trained at IOU threshold 0.3. On the other hand, if the region proposal algorithm regularly produces high-quality, well-localized proposals, it would make sense to use the 0.6 IOU threshold.

The mean improvement based on proposal IOU is given in Figure 4.2. The plots shows a curious relationship: the shape is the same for all groups, but as the training IOU threshold increases, the magnitude of improvement

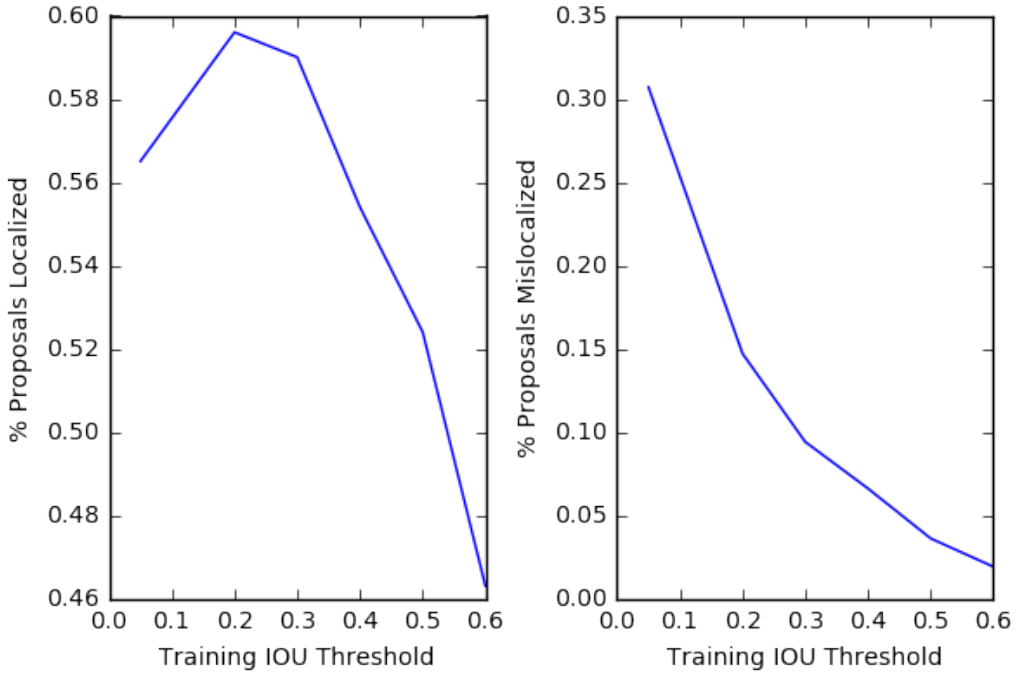

Figure 4.1: Trade-off between performance and mislocalization. Percentage localized is calculated as the number of samples from the test set where proposal IOU < 0.5 and prediction IOU ≥ 0.5 divided by the total number of proposals with IOU < 0.5. Mislocalization is the opposite calculation.

is squashed while the point at which the magnitude flips from positive to negative moves to ever higher proposal IOUs. Of course, as the proposal IOU increases, the amount that it is possible to improve decreases, as the maximum is 1.0—ground truth. In practice, mean IOU improvement does not come close to the ground truth limit, but it highlights an important facet of BBR: it is easy to improve a box with low IOU, and hard to improve one that is already good.

Figure 4.3 illustrates this is another way. Looking at proposal IOU vs

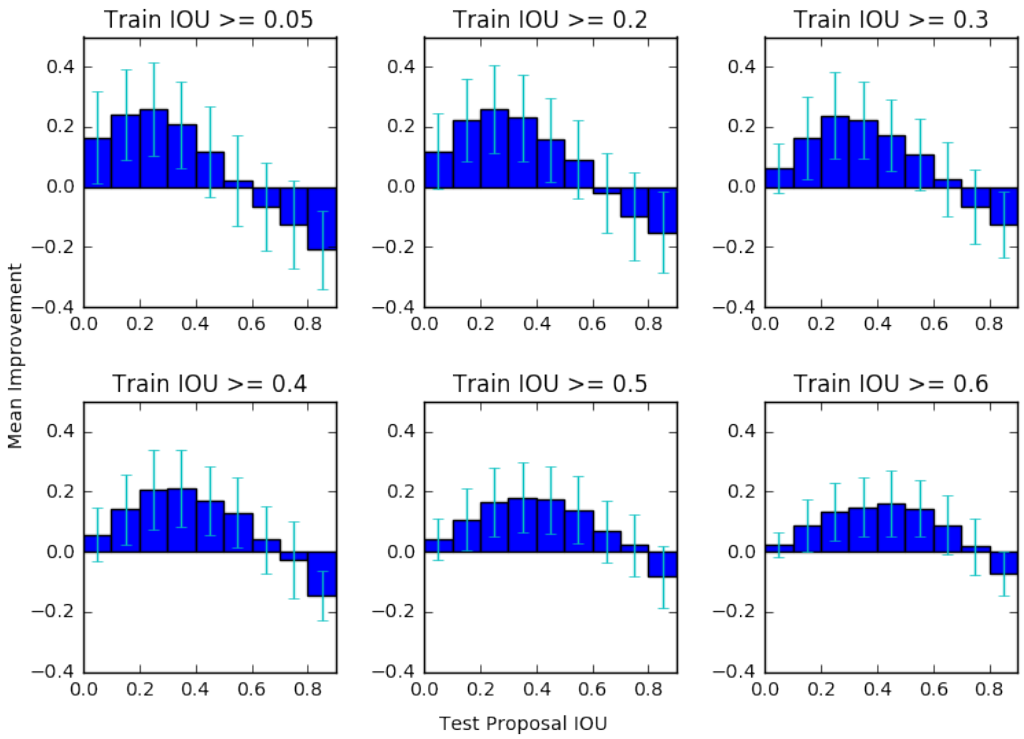

Figure 4.2: Test proposal IOU plotted against mean improvement from BBR. Each bar shows the expected prediction improvement based on initial proposal IOU. Error bars show standard deviation to demonstrate expected variation.

final IOU again, the two form a conical shape that is tighter as training IOU increases. The training set with 0.6 IOU threshold produces consistent, even results, but may not provide enough improvement to get low IOU proposals past the localization threshold of 0.5 IOU. Lower IOU training sets produce sloppier, less predictable results, especially on higher IOU proposals, but are more often capable of localizing proposals with low IOUs.

I combined all six training sets to briefly examine the effect of a larger

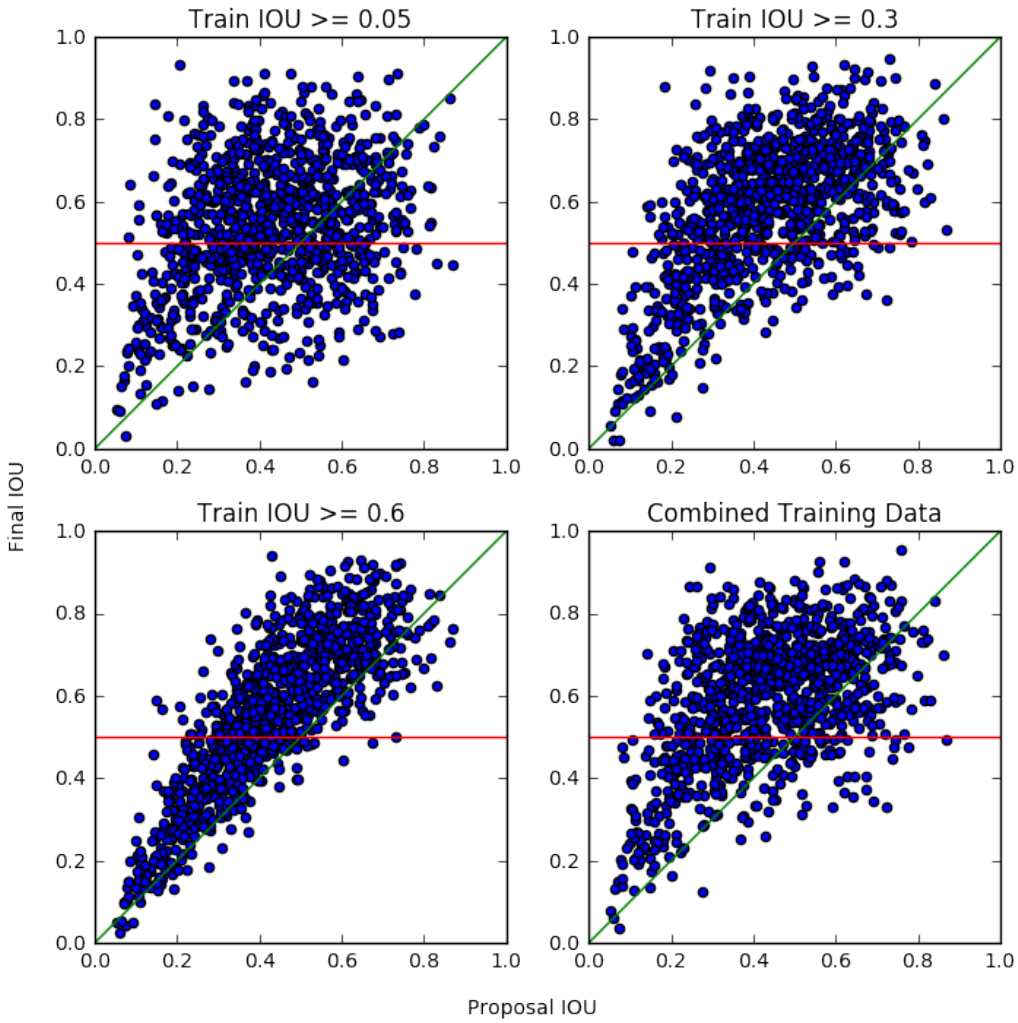

Figure 4.3: Proposal IOU vs Predicted IOU on Test Set. The green line represents the baseline of no improvement, and the red line shows the localization threshold. Figure best viewed in color.

training set on performance; note that this set has a very different IOU distribution from any of the individual training sets. Thus, the results in Table 4.2 aren't directly comparable to those in Table 4.1. The combined training set, which contains 60 proposals from each image, out-performs every other

category. This result suggests that 10 is a conservative estimate on the number of jitters than can be sampled from a single image before oversaturating the training set. It demonstrates that some of the largest performance gains can be obtained simply by generating more training examples.

Training Set	Mean Improvement	MRI	% Localized	% Mislocalized
All Data	0.157	0.536	62.8	10.8

Table 4.2: Effect of training set size on performance on test set.

It seemed possible that a larger training set might also not require such a large amount of regularization, in the form of the λ value, since the training set size is several times larger than the feature vector; however, that proved false. Even with a training set several times larger than the number of features, the optimum λ was still 18000 as determined by the validation set.

4.2 Effect of Context Padding

In this experiment, I test the findings in Girshick et al., who claim that 16px of proportional context padding around the proposal produces the best results. This means that enough context padding is taken from the image around a proposal such that once scaled to 224×224, there are exactly 16px of padding on all sides. If the proposal is taken from the edge of an image, I use zero padding (black pixels) to fill in that edge. I test proportional padding against constant context padding and no context padding. For constant padding, I take 16px of context padding around a proposal before scaling,

regardless of size or aspect ratio, and for no context padding I crop the image directly at the proposal bounding box coordinates.

Results in Table 4.3 show that context padding improves performance, although proportional padding only has a 4% higher mean improvement relative to constant padding. It is a small edge, enough to localize 2.5% more of the test set. Context padding has little significant effect on performance, but the proportional padding recommended in [3] outperforms other variants slightly and is used in the remaining experiments.

Context Padding	Mean Improvement	MRI	Loc. Improvement
No Padding	0.129	0.412	0.31
Constant Padding	0.136	0.453	0.317
Proportional Padding	0.142	0.459	0.342

Table 4.3: Effect of context padding on test set. Model trained on proposals ≥ 0.3 IOU, features from fc_6, and $\lambda = 18000$.

4.3 Effect of CNN Feature Layer

It is possible to use features from any layer of a CNN for BBR—R-CNN uses $pool_5$ features whereas Faster R-CNN uses fc_7 features. SPPnet uses $conv_{5\_1}$ features [17]. Girshick et al. suggest using $pool_5$ because it captures more general features than the fully-connected layers, but none of the literature explicitly discusses how the CNN feature layer was chosen. Thus, I examine the effect of features from different layers of VGG-16 on BBR performance.

Networks continue to deepen, so it is difficult to say if pool_5 in AlexNet (used by R-CNN) is equivalent to pool_5 in VGG-16 (used here and in Faster R-CNN) in any meaningful way. Instead, I focus on relative network depth in the same network and perform BBR with features from pool_4, pool_5, fc_6, and fc_7.

For general classification problems, the last several years of improvements show that deeper networks produce better results. However, many networks are not trained to localize, so deeper layers will not necessarily be tuned in any meaningful way for BBR. They may even focus on complex, fine-tuned features that are irrelevant to the task of determining if an object has been adequately localized. As mentioned above, [3] suggests that earlier layers are more generalizable, but in deep networks these layers may have hundreds of thousands of features, which may only exacerbate the overfitting problem.

Feature layer tests all use the mid-range (IOU threshold ≥ 0.3) training data found to be highest performing in the previous experiment. Initial runs using the validation data to determine λ for each layer revealed that pool_5 and pool_4 features performed almost identically across the entire range of lambda, potentially due to the large number of features, 25,088 and 100,352, respectively. fc_7, on the other hand, was the first set of features to reach a λ cap and start degrading for values over 2000.

The results of the final tests in Table 4.4 show that fc_6 outperforms all other feature layers by a small margin. Notably, pool_5 and pool_4 have comparable localization performance, but pool_5 especially suffers from a much

Feature Layer	Mean Improvement	MRI	% Localized	% Mislocalized
fc_7	0.139	0.436	54.8	8.3
fc_6	0.147	0.477	59.0	9.4
$pool_5$	0.133	0.453	57.9	18.0
$pool_4$	0.140	0.437	57.4	10.2

Table 4.4: Effect of extracting features at different CNN layers on regression performance. Layers displayed here from deepest to shallowest.

higher mislocalization rate. Using features from deeper in the network and closer to the output decreases performance, although it marginally decreases errors as well.

4.4 Effect of PCA

The huge number of features in $pool_5$ and $pool_4$ motivated me to test whether ridge regression is sufficient for regularization, especially given that the training set is many times smaller than the feature vectors. I look to principal component analysis (PCA) to perform dimensionality reduction. PCA finds linear combinations of features, known as components, with the greatest variance then discards all but the n most important features—essentially creating the most descriptive set of features in n-dimensions from the given data.

I use the validation set to determine the best number of components for each CNN feature layer. Validation runs showed reduction to 1000 principal components optimal for $pool_5$, 2000 components for $pool_4$, and 1500 components for fc_6 and fc_7. The value of λ has no effect on the results; although

PCA and ridge regression work quite differently, both perform regularization and feature selection. All values of λ produced the same results in validation tests with PCA. In this experiment, ridge regression could be replaced with unregularized linear regression, but I use ridge regression with $\lambda = 0.1$ both for consistency, and because it ensures the matrix Φ is invertible.

Feature Layer	Mean Improvement	MRI	% Localized	% Mislocalized
fc_7 + PCA	0.145	0.436	55.9	10.0
fc_6 + PCA	0.123	0.426	54.0	16.1
$pool_5$ + PCA	0.161	0.495	60.7	3.9
$pool_4$ + PCA	0.145	0.439	57.6	5.3

Table 4.5: Effect of extracting features at different CNN layers on regression performance using PCA. Layers displayed here from deepest to shallowest.

A summary of the effects of applying PCA to each feature layer is presented in Table 4.5. Compared to the results in Table 4.4, both pooling layers demonstrate significant performance increase and a significantly lower percentage of mislocalizations. After PCA, $pool_5$ shows the greatest improvement, and outperforms all layers (both with and without PCA) at all metrics. The two fully connected layers perform worse after PCA than before. This supports my hypothesis that ridge regression is not sufficient to handle the unwieldy number of features in the pooling layers, but both contain effective features for BBR once reduced to a meaningful and manageable number.

4.5 Effect of Object Class

Finally, I compare the leash, person, and dog classes from the Portland Dog-Walking dataset. As one might expect, person and dog, which are both present in the ImageNet dataset VGG-16 used for training, perform much better than leash, which has no analogue in the training classes. Table 4.6 summarizes the findings: dog and person have similar results, while leash has less than half the percentage of localizations and more than three times the percentage of errors. This sounds more dismal than it really is; the leash regressor still has a net gain in localizations. It is not impressive performance, but neither is it a loss.

Class	Mean Improvement	MRI	% Localized	% Mis-Localized
Dog	0.161	0.495	60.7	3.9
Person	0.151	0.439	63.2	4.1
Leash	0.056	0.198	28.6	15.2

Table 4.6: Performance comparison across object class. Pool_5 features + PCA are used for all tests.

It is possible, and even likely, that having a CNN trained to recognize leashes would improve the features produced for BBR; however it is still an intrinsically hard problem to find what is, essentially, a very small edge connecting person and dog. Appendix A contains image sets of the best and worst bounding box predictions, and it would seem that one of the features the regressor prioritizes is finding the hand in the image. Such predictions

are hard to eyeball, and would require a carefully crafted batch of inputs to validate.

4.6 Good Proposals and Bad Proposals

One question lingers: What makes a proposal easy or hard to regress from? The previous experiments show that the initial proposal IOU and object class are significant factors, but is there anything else that might help predict how a given proposal will fare? It seems likely that the initial image itself would affect the difficulty of the problem, but examination of the best and worst predictions shows that not only does the same image rarely crop up more often than is probable in each set, the same image sometimes shows up in the top 10 best and worst. Figure 4.4 shows one example of this from the dog set. Further illustration of this can be seen in Appendix A.

No other obvious characteristics link the proposals that produce high quality predictions or those that do not. Examining the proposal bounding boxes reveals nothing: aspect ratio or direction(s) of skew from ground truth do not appear significant. Further answers must lay in the low-level image features themselves, analysis of which is beyond the scope of this work. The proposal IOU remains the most telling factor, which makes sense. It is easier to improve a bad box than a good one. Figure 4.5 demonstrates this; some of the worst predictions were on proposal boxes that were already bad to begin with, but overall most of the bad predictions come from already localized proposals. The median proposal IOU for the worst predictions is 0.67,

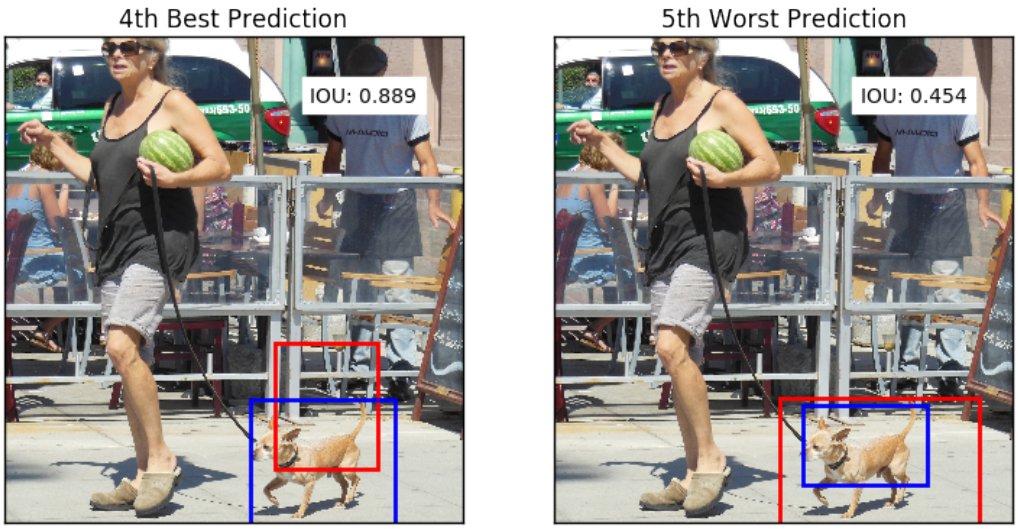

Figure 4.4: Localizations on the same image. One of the best and one of the worst predictions on the dog dataset come from the same original image. The red rectangle is the proposal bounding box and the blue is the predicted bounding box. IOU is given for the predicted box against ground truth.

compared to a median 0.30 for the best. (Notably, exactly at the training IOU threshold for this dataset.)

4.7 A Note on Performance Speed

Although speed is a critical factor in all but the most trivial applications, I choose not to discuss it here for a number of reasons. First and foremost, the most time-consuming aspect of this process is extracting image features with the CNN. In any serious implementation, the features should be shared between the CNN during classification and the bounding box regressor, so that the overhead of adding a regression stage is only the time to perform the

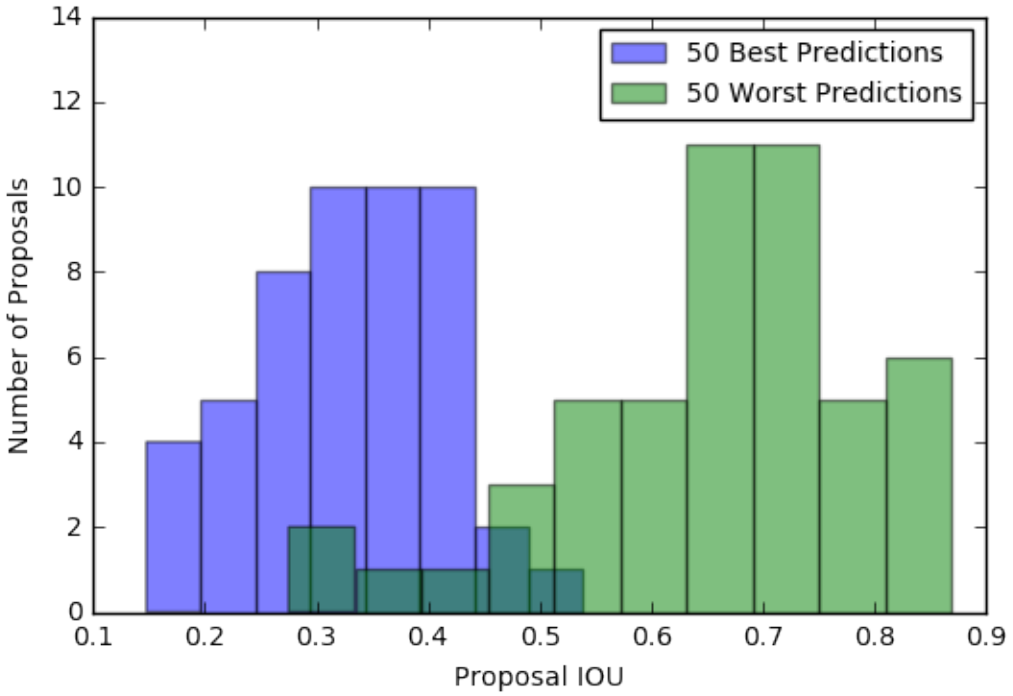

Figure 4.5: Best and worst predictions on the dog class by initial proposal IOU. Best and worst predictions as measured by IOU improvement of the prediction from proposal IOU, demonstrating that the best predictions generally come from unlocalized proposals, and the worst predictions from already localized proposals.

regression itself and some extra memory to store the features for as long as they are needed. The time needed for a prediction is trivial, the predictions can be performed efficiently in batches, and any quality machine learning library will have an optimized ridge regression function.

Chapter 5

Discussion

5.1 Conclusion

Bounding-box regression has been incorporated into many different object recognition systems as a cheap way to compensate for localization errors. In spite of its prevalence, there is little discussion or analysis of its performance in the literature. This thesis presents a survey of techniques for optimizing BBR, which used in conjunction with one another produce a 14.7 percentage point increase in localizations over my default method[1], from 46.3% to 60.7%.

I show that pool_5 features produce the best results, as first suggested by Girschick et al. [3], even on a much deeper CNN architecture than AlexNet used in R-CNN. More recent implementations of BBR have switched to using features from the fully-connected layers, since pool_5 is an order of magnitude larger in deep CNNs; however, I show that PCA can be used to reduce feature dimensionality from 25k to 1k and get superior results. This has the additional benefit of eliminating the need for regularization.

[1] FC_6 features, proportional padding, training on samples greater than 0.6 IOU.

Further results from my experiments reject the prevailing wisdom that it is necessary to choose training samples with high ground-truth overlap; rather, a training IOU threshold is useful but should be chosen carefully by use case. Training on a greater range of samples increases performance on low IOU proposals, at the expense of greater likelihood of mislocalizing high IOU proposals. Thus, some knowledge of the general quality of proposals produced by the object detection algorithm can help choose an appropriate training regime. It is possible that BBR could be paired with an extremely efficient but not particularly accurate detection method to great effect.

5.2 Future Work

The next step is to integrate this work with an actual object detection system, such as R-CNN, in order to be able to compare these results in a meaningful way. It is unclear how an increase in localization percentage translates into mAP for a system as a whole, and until then the significance of my findings remains unknown.

Object classes also merit further study; in-depth analysis of bounding box regression for some of the PASCAL VOC classes which benefited from it least, such as *chair* and *tv*, might reveal important class-specific parameters that can be tuned to further increase performance. My experiments only briefly examine the results on different object classes, and do not delve into whether the optimum settings I have selected vary based on object class.

Bibliography

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[2] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, (Washington, DC, USA), pp. 3354–3361, IEEE Computer Society, 2012.

[3] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[4] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vision*, vol. 104, pp. 154–171, Sept. 2013.

[5] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *Proceedings of the 12th European Conference on Computer*

Vision - Volume Part III, ECCV'12, (Berlin, Heidelberg), pp. 340–353, Springer-Verlag, 2012.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[7] 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," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[8] M. H. Quinn, A. D. Rhodes, and M. Mitchell, "Active object localization in visual situations," *CoRR*, vol. abs/1607.00548, 2016.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[10] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *International Conference on Computer Vision & Pattern Recognition (CVPR '05)* (C. Schmid, S. Soatto, and C. Tomasi, eds.), vol. 1, (San Diego, United States), pp. 886–893, IEEE Computer Society, June 2005.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Infor-*

mation Processing Systems 25 (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[12] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[13] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 1627–1645, Sept. 2010.

[16] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014.

[18] machrisaa, "tensorflow-vgg," 2016.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

Appendix A

Visualizing Best and Worst Predictions

Best and worst predictions are chosen by prediction IOU improvement from proposal IOU out of a test set of 1000 proposals extracted from 100 images. The best are defined as the those with the greatest IOU improvement (predicted IOU - proposal IOU), similarly the worst are those with the least IOU improvement.

The twelve best dog predictions, based on IOU improvement. Red box is proposal, blue box is prediction.

The twelve worst dog predictions, based on IOU improvement.

The twelve best leash predictions, based on IOU improvement.

The twelve worst leash predictions, based on IOU improvement.