6-1-2018

Improving region based CNN object detector using bayesian optimization

Amgad Muhammad

Improving Region based CNN object detector using Bayesian Optimization

Amgad Muhammad

Supervisor: Professor Mohamed Moustafa

A thesis presented for the Master's degree

Department of Computer Science and Engineering

The American University in Cairo

Egypt

December 07, 2017

Abstract

Using Deep Neural Networks for object detection tasks has had ground breaking results on several object detection benchmarks. Although the trained models have high capacity and strong discrimination power, yet inaccurate localization is a major source of error for those detection systems. In my work, I'm developing a sequential searching algorithm based on Bayesian Optimization to propose better candidate bounding boxes for the objects of interest.

The work is focusing on formulating effective region proposal as an optimization problem and using Bayesian Optimization algorithm as a black-box optimizer to sequentially solve this problem. The proposed algorithm demonstrated the state-of-the-art performance on PASCAL VOC 2007 benchmark under the standard localization requirements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning has improved many computer vision tasks and motivated by its success in the object classification tasks [1, 2, 10, 14, 15, 20], a significant effort has been put in place to use deep learning to improve the object detection systems. The most notable effort is Region based CNN architectures [3, 7, 8, 19] that demonstrate state-of-the-art performance on standard detection bench marks [5, 16] and near real time execution.

There are two crucial components that makes Region based CNN architectures excel at the object detection task. The first component is replacing the low-level hand engineered features like HOG [4] or SIFT [17], with CNN feature maps which arguably have larger representation capacity. The only downside of the CNN features is that they are expensive to compute and that's where the second component is used, a region proposal algorithm used to propose regions of interest (ROI), that will likely contain the object of interest hence reducing the features computation time by focusing the network attention on a smaller set of ROIs.

Despite the success of region based CNN detectors, the region proposal algorithm is still considered a week point. Consider the example that none of proposed regions is near the ground truth bounding box, so no matter how discriminant the features extracted from the CNN are, there is no way to detect the correct bounding box of the object of interest, and indeed in many application, like autonomous driving, an accurate bounding box is important.

In my work, I will use a search algorithm that uses the initial detections to propose, sequentially, new bounding boxes that will likely have higher detection scores and are closer to the ground truth. The search algorithm is implemented using the Bayesian Optimization framework [24] to replace the complex detection function with surrogate model that captures the probability distribution of the detection scores given the extracted CNN features.

The rest of the thesis is structured such that in chapter two I will discuss the related work and modern object detection frameworks. Chapter three serves as an introduction to Bayesian Optimization as a black box optimizer in its general form. In chapter 4, I'll discuss how we can formulate the region proposal problem and make it fit into Bayesian Optimization mould by introducing the local maxima search algorithm to refine the proposed regions. Before I conclude in chapter 6, I'll discuss the experiments and results in chapter 5.

Chapter 2

Related Work

R-CNN: Region based CNN (RCNN)[8] was the approach that pioneered using region proposal on top of CNN features as an object detector.

In RCNN, there are three modules. The first module is an external region proposal algorithm called selective search, where class-independent regions are proposed randomly [25] and each ROI is cropped and wrapped and sent to the second module for feature extraction. In feature extraction module, a 4096-dimensional feature vector is extracted for each region. The features are calculated by forward propagating a 227x227 RGB image through five convolution layers and two fully connected layers. Regardless of the size or the aspect ratio of the candidate region, a warping operation is done to resize the region to the network expected size. The last module is a one v.s. all SVM trained for every class using the extracted features from the CNN. At test time the selective search algorithm is used to propose 2000 regions per image which are then propagated through the CNN for feature extraction and then scored across the different trained SVM classifier. Given all the scored regions a greedy non-maximum suppression is applied to reject regions that has an intersection-over-union (IoU)[6] overlap with a higher scoring selected region larger than a learned threshold. Figure 2.1 is an overview of the RCNN architecture. The main issue with RCNN approach was the repeated, expensive computation that was needed to be done to extract the CNN features for each of the 2000 proposed ROI at test time.

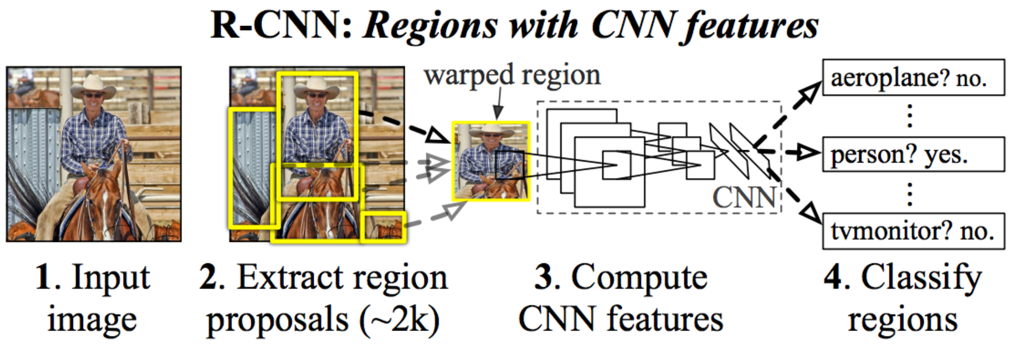


Figure 2.1: RCNN overview. The system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolution neural network (CNN), and then (4) classifies each region using class-specific linear SVMs[8]

In Fast-RCNN [7], the author took a different approach, instead of cropping the ROI from the image directly, the cropping is delayed until an intermediate feature map, with this trick many feature computation is done on the whole image and done only once. What enabled the Fast-RCNN network to extract a fixed size feature vector from each proposed ROI, regardless of its shape or aspect ratio, is a newly introduced layer called ROI pooling layer. The ROI pooling layer uses max pooling to convert features inside any valid proposed region into a small, fixed size feature map. Another differentiator of the Fast-RCNN approach is the elimination of the multi-stage training used in RCNN, where we had to train a CNN for feature extraction and then train an SVM for detection. In Fast-RCNN, the authors introduced a multi-task loss in a form of two sibling output layers. The first output layer is a softmax loss function to model the discrete probability of the K object classes we want to detect, and the second sibling layer is a bounding-box regression offsets for each of the K object classes. With every ROI labeled with a ground truth class u and a ground truth bounding box target v, the multi-task loss function L can be formulated as follows

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \tag{2.0.1}$$

, where p is the predicted class probability and t^u is predicted bounding box coordinates. Figure 2.2 is an overview of the Fast R-CNN architecture.

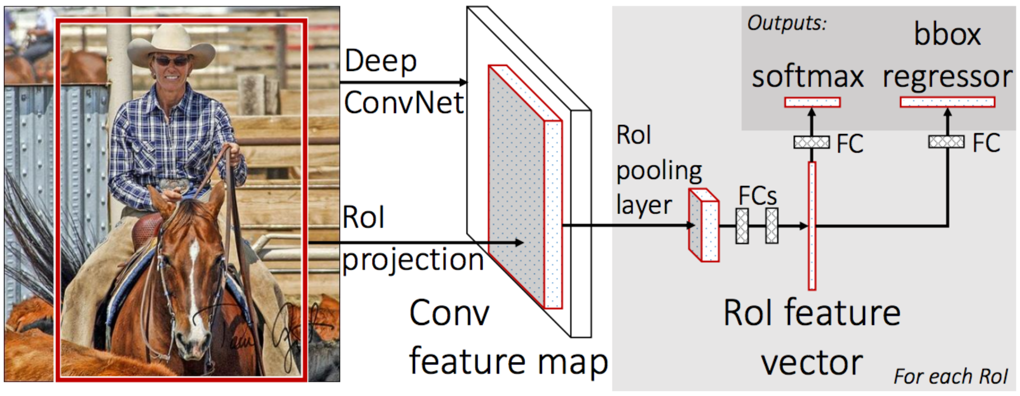


Figure 2.2: Fast R-CNN overview. An input image and multiple regions of interest (RoIs) are input into a fully convolution network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: soft-max probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.[7]

Both RCNN and Fast-RCNN relied on an external region proposal algorithm, however in Faster-RCNN [19], the architecture included two modules, a Fast-RCNN network and a dedicated region proposal network, which is trained on anchors to generate good candidate regions while at the same time sharing the convolution layers computations with the Fast-RCNN network making the region proposal cost-free. To generate region proposals a small, sub network with two sibling heads, a classification head and a bounding box regression head, is trained on the features extracted from a shared set of convolution layers. This mini-network is illustrated at figure 2.3 extracting features from the last shared convolution layer at a single location.

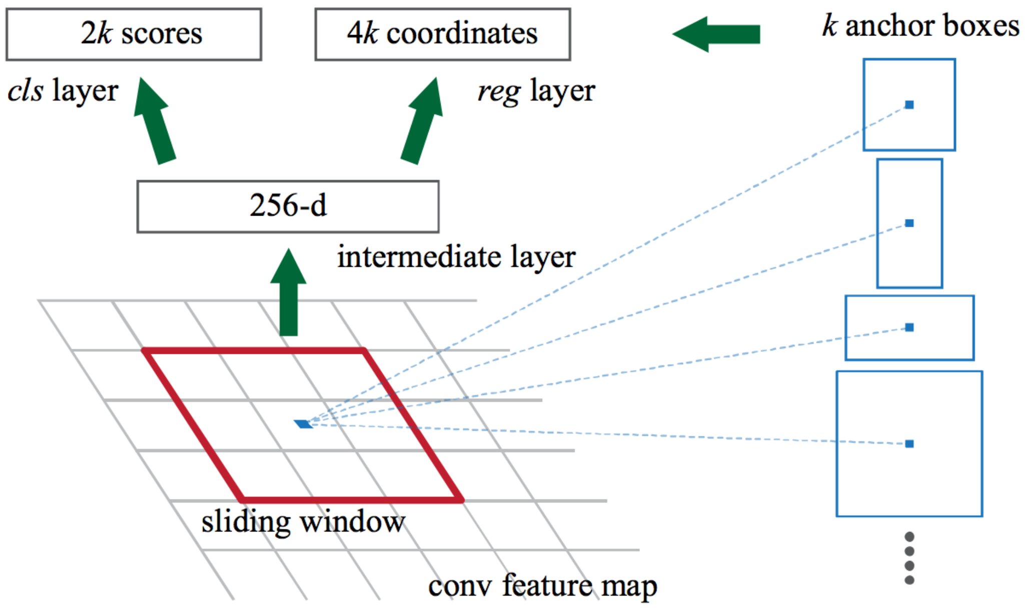


Figure 2.3: RPN with a sliding window of size 3x3 extracting features to train the network two sibling heads.[19]

Two train the two modules of the Faster-RCNN the authors opted to a pragmatic 4-step alternating training. In the first step the first module of the architecture, the RPN, is trained on a set of anchors to optimize a loss function similar the one used in the Fast-RCNN framework. In the second step, the second module, the Fast-RCNN detection network, is trained on the regions proposed by RPN in step one. Up until this moment no convolution layers are shared between the two modules. In the third step, the convolution layers of the Fast-RCNN is used to initialize the convolution layers in the RPN, and the training continues to fine tune only the layers specific to the

RPN. The last step, while keeping the convolution layers fixed, the training for the Fast RCNN is continued but this time using the RPN output to fine tune the unique layers of the Fast-RCNN.

Although Faster R-CNN is an order of magnitude faster than Fast R-CNN, but still the region-specific component must be applied several hundred times per image, however in R-FCN[3] (Region-based Fully Convolutional Networks) method the author pushed cropping the features from the same layer where region proposals are projected, down to the last layer of the features prior to prediction, thus minimizing the amount of per-region computation that must be done.

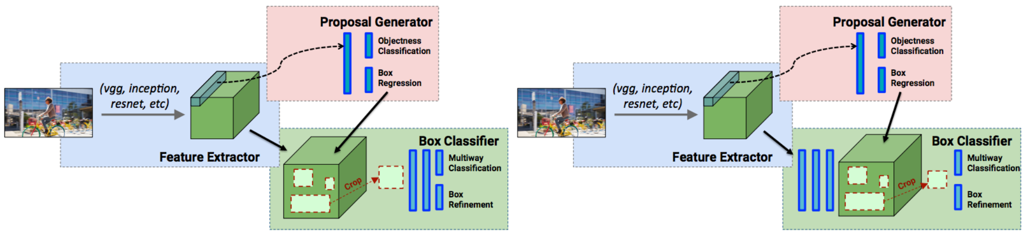


Figure 2.4: Faster RCNN *(left)*, and R-FCN [11] *(right)*.

Localization refinement can also be considered a CNN problem but instead of classification it is regression. Girshick et al. [8] extracted the middle layer features and linearly regressed the initially proposed regions to better locations. The overfeat framework [21] refined bounding boxes from a grid layout to flexible locations and sizes using the higher layers of the deep CNN architecture. Fast and Faster RCNN[7, 19] jointly conducted classification and regression in a single architecture.

My work will be based on [27] which uses the information from multiple existing regions instead of a single bounding box for predicting a new candidate region, and it focuses only on maximizing the localization ability of the CNN classifier instead of doing any regression from one bounding box to another. In addition, instead of applying [27] method on regions proposed using the selective search algorithm [25], the algorithm will be applied on the proposals from the RPN network in the Faster RCNN framework.

Chapter 3

Bayesian Optimization as black-box optimizer

Bayesian optimization is a black-box optimization algorithm that is best used to maximize/minimize expensive objective functions. As a black-box optimization algorithm, Bayesian optimization searches for the extremum of an unknown objective function from which a pair of [input, output] can be obtained. Bayesian optimization is model-based optimization technique which means it creates a model of the objective function with a regression method, uses this model to select the next point to acquire, then updates the model according to the new point and its true objective function evaluation. It is called Bayesian because, in its general formulation, this algorithm chooses the next point by computing a posterior distribution of the objective function using the likelihood of the data already acquired and a prior on the type of function.

$$p(f|D_n) \propto p(D_n|f)p(f), \tag{3.0.1}$$

Bayesian optimization uses the prior and evidence to define a posterior distribution over the space of functions. Following the Bayesian model allows us to describe some of the objective function attributes, such as smoothness, that we think important using the informative prior. Optimizing follows the principle of maximum expected utility, or, equivalently, minimum expected risk. The process of deciding where to sample next requires the choice of a acquisition function and a way of optimizing this function with respect to the posterior distribution of the objective function.

Algorithm 1 General Bayesian Optimization

```
1: for $i \leftarrow 1, t$ do
2:     find $x_i$ by optimizing the acquisition function over the posterior distribution of the objective function.
3:     Evaluate the object function at $x_i$, $y_i = f(x_i)$
4:     Augment the data $D_{1:i} \leftarrow D_{1:i-1} \cup \{(x_i, y_i)\}$ and update objective function posterior distribution.
```

Following the Bayesian optimization loop in Algorithm 1, we can extract two components: the posterior distribution over the objective function and the acquisition function. To build the posterior distribution, we accumulate observations $D_{1:i} \leftarrow D_{1:i-1} \cup \{(x_i, y_i)\}$, a prior distribution $P(f)$ is then combined with the likelihood function $P(D_{1:i}|f)$ to produce the posterior distribution: $p(f|D_n) \propto p(D_n|f)p(f)$. The posterior captures the updated beliefs about the unknown objective function and act as surrogate of the objective function. In the next section we discuss how we fit a GP and use it as the prior, $P(f)$,for the objective function.

3.1 Gaussian Process as objective function prior

A Gaussian process is a generalization of the Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic process governs the properties of functions we can loosely think of a function as a very long vector, each entry in the vector specifying the function value $f(x)$ at a particular input $\mathbf{x}$. Gaussian processes are particularly interesting for regression because they not only model the cost function, but also the uncertainty associated with each prediction. For a cost function $f(x)$, a Gaussian process defines the probability distribution of the possible values $f(x)$ for each point $\mathbf{x}$. This distribution over functions is completely specified by a mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \hat{\mathbf{x}})$.

$$f(x) \sim GP(m(x), k(x, \hat{x})), \tag{3.1.1}$$

To put it differently, when using **GP** as $f(x)$ prior, instead of returning a scalar value the **GP** returns the mean and the variance of a normal distribution over the possible values of f at $\mathbf{x}$.

Now the interesting question would be what the covariance function k be. A very popular choice would be to use the squared exponential function:

$$k(x_i, x_j) = exp(-\frac{1}{2} \parallel x_i - x_j \parallel^2) \tag{3.1.2}$$

This function approaches the value of 1 as the points get closer and approaches 0 as they get further apart, this means that closer points are highly correlated and far away points have less influence on each other. Now let's

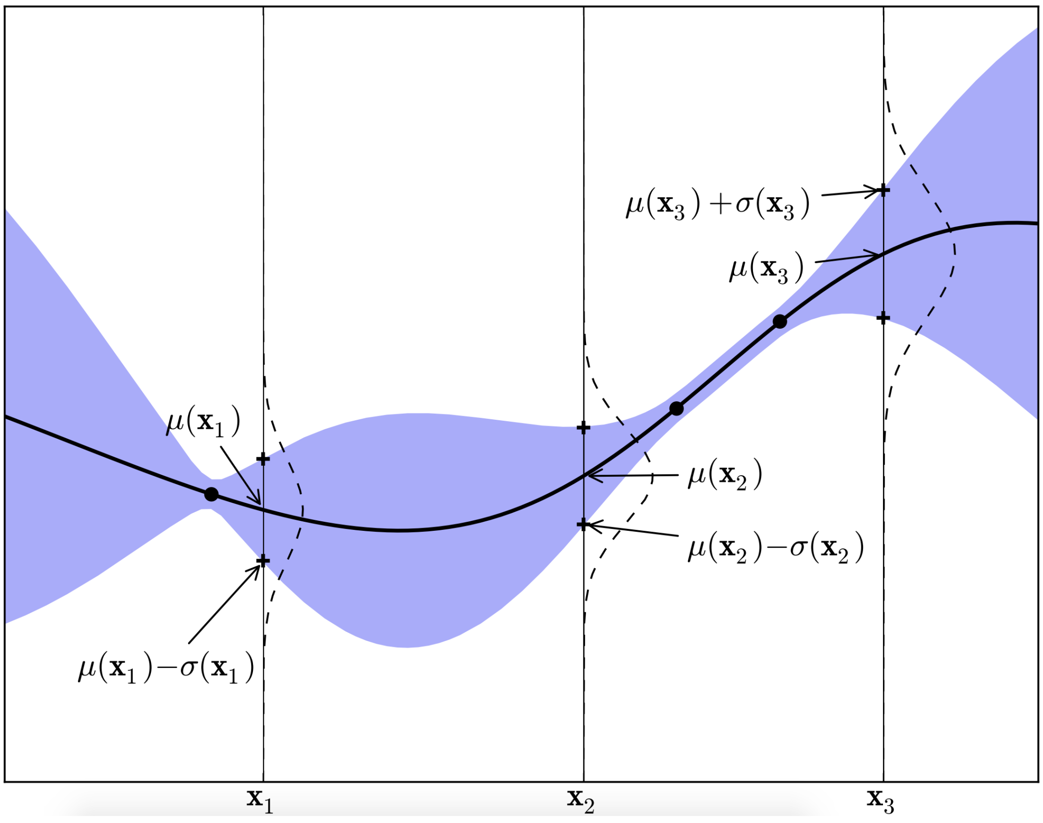


Figure 3.1: Simple 1D Gaussian process with three observations. The solid black line is the GP surrogate mean prediction of the objective function given the data, and the shaded area shows the mean plus and minus the variance. The superimposed Gaussians correspond to the GP mean, $\mu(.)$, and standard deviation, $\sigma(.)$, of the prediction at the points, $x_{1:3}$.[22]

say we have a set of points $x_{1:t}$, and we evaluate the objective function at each point to produce the pairs $x_{1:t,f_{1:t}}$, we can say that the function values are drawn according to to a zero mean multivariate normal distribution $N(0, \mathbf{K})$, where the kernel matrix is given by:

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \dots & k(x_t, x_t) \end{bmatrix}$$

Now we want to decide which point,x_{t+1}, should be considered next. Let us denote the value of the objective function at this arbitrary point as $f_{t+1} = f(x_{t+1})$, then by the properties of Gaussian processes, $f_{1:t} and f_{t+1}$, are jointly Gaussian:

$$\begin{bmatrix} f_{1:t} \\ f_{t+1} \end{bmatrix} \sim N\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k^T} & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right)$$

where

$$\mathbf{k} = [k(x_{t+1}, x_1)\ k(x_{t+1}, x_2)\ \ldots\ k(x_{t+1}, x_t)]$$

Using the Sherman-Morrison-Woodbury formula[23], we can arrive at an expression for the predictive distribution:

$$P(f_{t+1}|D_{1:t}, x_{t+1}) = N(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1}))$$

where

$$\mu_t(x_{t+1}) = \mathbf{k^T}\mathbf{K}^{-1} f_{1:t} \tag{3.1.3}$$

$$\sigma_t^2(x_{t+1}) = k(x_{t+1}, x_{t+1}) - \mathbf{k^T}\mathbf{K}^{-1}\mathbf{k} \tag{3.1.4}$$

That is, $\mu_t(x_{t+1})$ and $\sigma_t^2(x_{t+1})$ are the sufficient statistics of the predictive posterior distribution $P(f_{t+1}|D_{1:t}, x_{t+1})$.

3.2 Acquisition Functions for Bayesian Optimization

In this section we will discuss the second component of Bayesian optimization, acquisition functions. The acquisition function is used to guide the search for the optimum. Typically, acquisition functions are defined such that high acquisition corresponds to potentially high values of the objective function, whether because the prediction is high, the uncertainty is great, or both. In other words, we wish to evaluate the object function at $\operatorname{argmax}_x u(x|D)$, where $u(.)$ is the acquisition function.

3.2.1 Improvement-based acquisition functions

One of the widely used acquisition functions is *probability of improvement*[13] where the next point, $\mathbf{x}$ is chosen to maximize:

$$PI(x) = P(f(x) \geq f(x^+)) = \Phi\left(\frac{\mu(x) - f(x^+)}{\sigma(x)}\right)$$

where x^+ is the best point seen so far, and Φ is the normal cumulative distribution function. The drawback of this formulation is that it is pure exploitation, meaning that points which have high probability of being infinitesimally greater than (x^+) will be chosen over other points that offer larger gains but high uncertainty because the formula divides by $\sigma(x)$.

Another alternative acquisition function would be not to only consider the probability of improvement but also take into account the magnitude of that improvement. This alternative acquisition function is called the *expected improvement*[18] function with respect to $f(x^+)$:

$$EI(x) = \begin{cases} (\mu(x) - f(x^+))\Phi(z) + \sigma(x)\phi(z) & if : \sigma(x) > 0 \\ 0 & if : \sigma(x) = 0 \end{cases} \tag{3.2.1}$$

$$z = \frac{\mu(x) - f(x^+)}{\sigma(x)}$$

where $\phi()$ and $\Phi()$ denote the PDF and CDF of the standard normal distribution respectively.

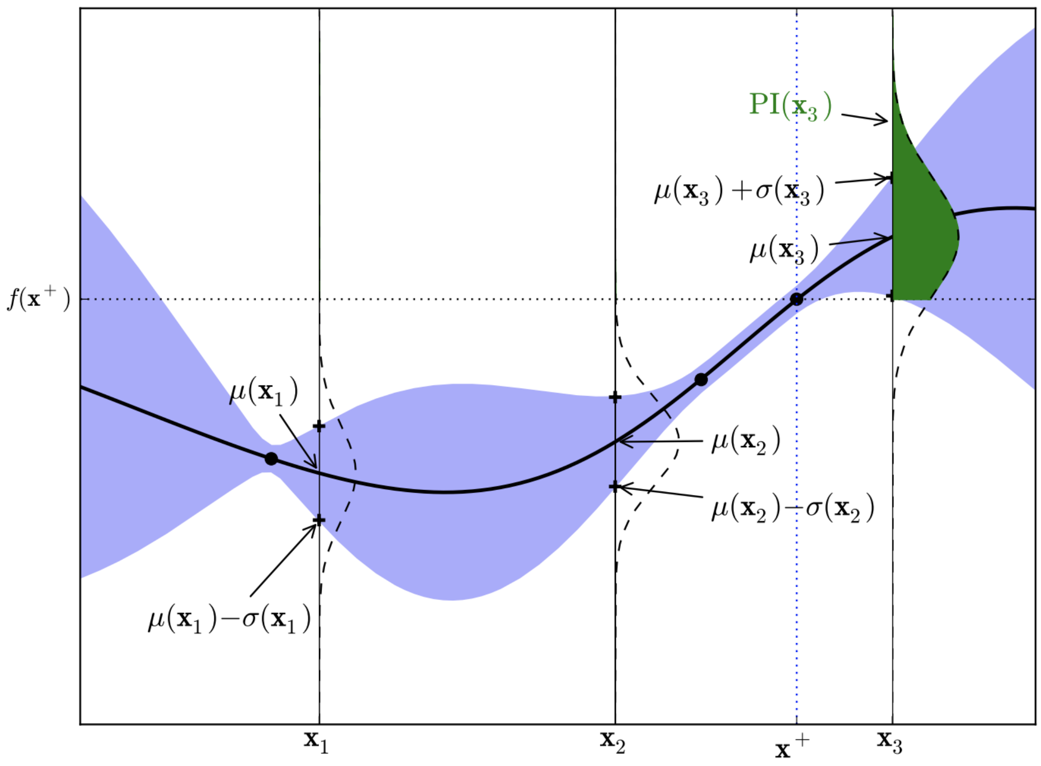


Figure 3.2: 1D Gaussian process additionally showing the region of probable improvement. The best point seen so far is at x^+. The darkly-shaded area in the superimposed Gaussian above the dashed line can be used as a measure of improvement. By sampling at x_1 or x_2, the model predicts almost no possibility of improvement, while sampling at x_3 is more likely to improve on $f(x^+)$.[22]

Chapter 4

Region Proposal using Bayesian Optimization

Let $f(x, y)$ denotes a detection score function that receives two inputs, x is the image and y as a region coordinates. The job of an object detection framework is to find the local maximum of f with respect to y on a new, unseen image x.

With the resolution of images we have right now it is crucial to find an efficient searching algorithm for the candidate regions as we have to evaluate the scoring function at many locations. The sliding window approach was the dominant method where a fixed size window is applied at different locations and scales to find the region that maximizes the detection score, however when f relies on features extracted from a CNN, it becomes more expensive to evaluate and such approach becomes unpractical.

In region-based CNN object detectors, the detection framework requires evaluating the detection scoring function f on a much fewer number of regions (e.g., few hundreds or thousands). However, there exist one potential issue of object detection pipelines based on region proposal which is that the correct detection will not happen if there is no region proposed in good enough proximity to the ground truth bounding box. To mitigate this risk usually, region based CNN detection pipelines propose more bounding boxes to cover the entire image more densely, but again this will increase the detection time significantly.

Within the Bayesian Optimization context which is used as a global maximizer of an unknown function $\boldsymbol{f}$[22], we can develop a sequential searching algorithm that uses the previously proposed regions to find new bounding boxes that are expected to have a higher detection scores without significantly increasing the number of proposed regions.

The main idea is to fit a probabilistic surrogate model, representing the prior distribution that captures our beliefs about the behaviour of $f(x, y)$ and then use an acquisition function that describes how optimal a sequence of queries, in our case regions, are. With the acquisition function guide, we can easily follow a sequence of queries to find the optimal region that maximizes f.

4.1 General Bayesian optimization framework

Let's assume that our detection score function $f(x, y)$ has a set of solutions $\{y_1, y_2, ..., y_n\}$. In the Bayesian Optimization framework, $f(x, y)$ is assumed to be drawn from a probabilistic model:

$$p(f|D_n) \propto p(D_n|f)p(f), \tag{4.1.1}$$

where $D_n = \{(y_j, f_j)\}_{j=1}^{n}$, and $f_j = f(x, y_j)$. Now once we fit a probabilistic model on f, the goal becomes sampling a new solution y_{n+1} with a high chance that it will maximizes the value of f_{n+1}. Here the chance is represented by an acquisition function $\alpha(y_{n+1}|D_n)$ that is used to trade-off between exploration (variance) and exploitation (mean) of the fitted probabilistic model on f. Once the model and the acquisition function is fitted, the Bayesian Optimization algorithm is used recursively to sample new position y_{n+1} and create the set $D_{n+1} = D_n \cup \{y_{n+1}\}$

Algorithm 2 Bayesian Optimization based Region Proposal

```
Require: Image $x$, classifier $f_{cnn}$, a set of regions with classifications scores $D_n$.
Ensure: A set of newly proposed regions with detection scores.
 1: function Propose Regions
 2:     $D \leftarrow D_n$
 3:     for $i \leftarrow 1, t$ do
 4:         $D_{proposal} = \varphi$
 5:         for all $(y_{best}, f_{best}) \in D$ do
 6:             $D_{local} = \{(y, f) \in D : IoU(y, y_{best}) > threshold\}$
 7:             $\hat{y} = \text{argmax}_y\, \alpha(y|D_{local})$
 8:             $\hat{f} = f_{cnn}(x, \hat{y})$
 9:             $D_{proposal} \leftarrow D_{proposal} \cup \{(\hat{y}, \hat{f})\}$
10:     $D \leftarrow D \cup D_{proposal}$
```

4.2 Bayesian optimization and Faster-RCNN

As mentioned in previous chapter Faster-RCNN uses an embedded region proposal network to propose class agnostic bounding boxes. Those boxes are then fed to a classifier which will assign class membership probability for every class we are testing against. In this section I will discuss how Bayesian optimization can improve the objectiveness of the proposed regions and get higher classification probability.

The Faster-RCNN architecture has a layer called region proposal layer, this layer is implemented in python and used to order all the regions that are proposed so far and return the regions with the highest objectiveness score to be tested with the classifier.

In order to apply Bayesian optimization on top of those regions, a network surgery had to take place by splitting the detection pipeline into two parts, the first part starts from the input layer up to the region proposal layer, and the second part, to be refereed to as N_{pro}, starts with the bounding boxes from the region proposal up till the classification layer.

With this split, I was able to formulate the objective function for the Bayesian optimization framework as:

$$P(c|x, y) = N_{pro}(x, y)[c] \tag{4.2.1}$$

where x is the image, y is the region coordinates, c is the class we are optimizing for.

4.3 Local Maxima Search using Bayesian optimization

With the defined object function $P(c|x, y)$, now the next step would be to decide on the set of the initial evaluated regions which will be used for fitting the GP model, and the choice came as follows. *First* we select the top R regions form the proposal layer according to their objectiveness scores, in my experiments R was 30.

For every region y in R we compute $P(c|x, y)$, and all the regions that had the background class as the most probable was discarded. What was left is a set of regions that are not background, and each one of those regions are used as an initial point *(local maxima)* for one Bayesian optimization

optimization run.

Once a local maxima is picked up, we select all the original regions that have pascal distance, IoU, greater than or equal ρ, in my experiments ρ was 0.65. We can think of this step as defining the domain of the regions that the Bayesian optimization can choose from to maximise our object function. The higher the ρ value the smaller the domain, the lower the ρ value the bigger the domain. I would choose hight ρ values to constraint the size of the regions that the Bayesian optimization will propose.

Once we had our set of regions (One local maxima and its IoU set) we map the coordinates of all those regions to a different space, instead of having the regions represented as $(x_1, y_1, x_2, y_2) \Rightarrow (left, top, right, bottom)$, I trained BO on regions with representation $(x_c, y_c, log(w), log(h))$, where x_c, y_c is the region center, and w, h is the region width and height, respectively.

With this setup, we can iterate N times and each time we use Bayesian optimization to maximize $P(c|x, y)$, by feeding in the initial regions, their class membership probability and regions domain, and getting out a new region that should have better classification score, then each newly proposed region is evaluated with the true objective function, and the true score is used to update the Gaussian process model as the objective function prior.

After exhausting the N iterations we end up with a new set of regions that should have higher detection scores. we keep repeating this process for every local maxima we have to support multiple objects in the same picture. Once the process is done a non-maxima suppression algorithm is used to produce the final detections.

Chapter 5

Experiments and Results

I trained and tested both Faster-RCNN and Faster-RCNN with Bayesian optimization on images with a single scale [9]. Each image is rescaled such that its shorter side is $s = 600$ pixels. The total stride for the last convolutional layer, for the ZF architecture [26], is 16 on the re-scaled images. For anchors, I used 3 scales with box areas $128^2, 256^2, 512^2$, and 3 aspect ratios of 1:1,1:2, and 2:1 as shown in figure 5.1.

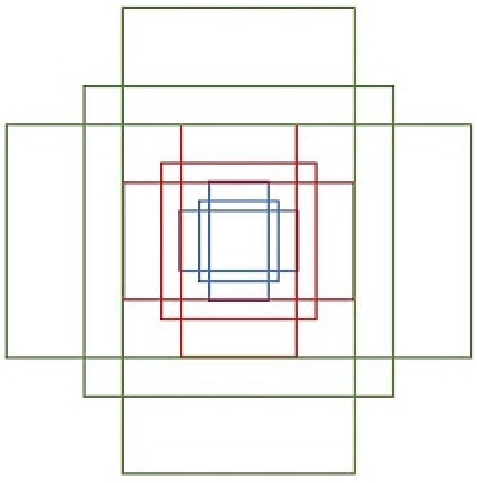


Figure 5.1: Anchors with 3 scales with box areas $128^2, 256^2, 512^2$, and 3 aspect ratios of 1:1,1:2, and 2:1

As anchors are generated for every image pixel, there will be anchors that cross image boundaries, those anchors needs to be handled with care.

For training, all cross-boundary anchors are ignored so that they don't contribute to the network loss. Roughly, there exists 20000 ($\approx 60 \times 40 \times 9$) anchors for a typical 1000×600 image, and with cross-boundary anchors ignored we are left with 6000 anchors for training.

As for testing, the same process is applied as the fully convolutional RPN is applied to the entire image and cross-boundary boxes maybe generated.

Additionally, some RPN proposals are highly overlapped so to reduce redundancy non-maximum suppression (NMS) is applied on proposed regions based on their *class* scores. I fixed the IoU threshold for NMS at 0.7, resulting in roughly 2000 proposed regions per image for testing.

After NMS, I used the top-N ranked proposed regions for both the baseline (Faster-RCNN) and the starting set for the Bayesian optimization local maxima search and top-M ranked regions to build the Bayesian optimization domain, by picking all the proposed regions that have an IoU ≥ 0.65 with the current local maxima. I conducted the local maxima search with top-M set having a maximum of 900 regions.

5.1 Experiments on PASCAL VOC

I evaluate my method on the PASCAL VOC 2007 detection benchmark [6]. This dataset is composed of 5000 trainval images and 5000 test images for 20 object categories. For the ImageNet pre-trained network, I use the ZF net [26] that has five convolutional layers and three fully-connected layers. I primarily evaluate detection mean Average Precision (mAP) as the metric for object detection.

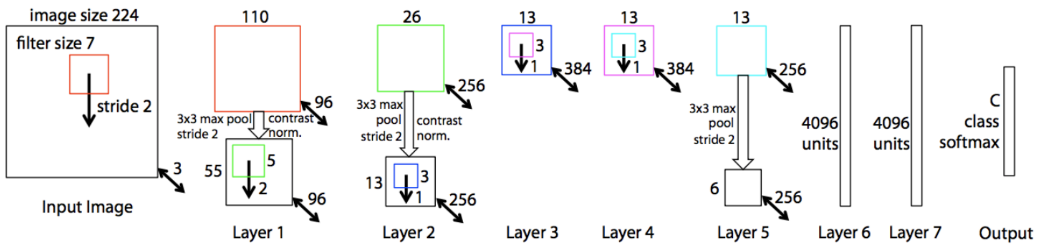


Figure 5.2: ZF Network Architecture.

The results are in Table 5.1 summarizes the performance of Faster R-CNN with and without the Bayesian optimization search algorithm.

Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	
Faster-RCNN (ZF)	0.609	0.609	0.485	0.375	0.233	0.677	0.626	0.70	0.276	0.695	
Faster-RCNN (ZF) + BO(Matern52)	**0.62**	**0.621**	0.483	**0.4**	**0.24**	**0.69**	**0.628**	0.70	0.274	0.68	

	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Faster-RCNN (ZF)	0.684	0.69	0.80	0.611	0.604	0.309	0.622	0.622	0.69	0.426	0.567
Faster-RCNN (ZF) + BO(Matern52)	**0.692**	**0.694**	**0.81**	0.61	0.604	0.3	0.62	**0.65**	**0.7**	**0.431**	**0.572**

Table 5.1: Test set mAP of PASCAL VOC 2007. The first row is the baseline using Faster RCNN with number of regions to be tested equals 30. The second row is the Faster RCNN augmented with Bayesian Optimization search algorithm, a GP is used as the objective function prior and Matern52, $k(r^2) = \left(1 + \sqrt{5\,r^2} + \frac{5\,r^2}{3}\right) \exp\left(-\sqrt{5\,r^2}\right)$, is used as the GP kernel and the GP domain threshold is set to 0.65.

5.1.1 Kernel effect on GP fitting

The results shown in Table 5.1 are for successful integration between Faster-RCNN and ROBO - a Robust Bayesian Optimization framework [12] and in this section I tune ROBO to use different kernel to better capture the spatial relationship between two proposed regions. One popular choice would be to use the ExpSquared kernel to give high correlation coefficient for close-by regions and exponentially decays as the regions move apart.

As shown in Table 5.2, choosing the right kernel is important to get even

Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	
Faster-RCNN (ZF)	0.609	0.609	0.485	0.375	0.233	0.677	0.626	0.70	0.276	0.695	
Faster-RCNN (ZF) + BO(ExpSquared)	**0.6223**	**0.62**	**0.486**	**0.471**	**0.24**	**0.68**	0.626	0.70	0.275	**0.696**	
	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Faster-RCNN (ZF)	0.684	0.69	0.80	0.611	0.604	0.309	0.622	0.622	0.69	0.426	0.567
Faster-RCNN (ZF) + BO(ExpSquared)	**0.6844**	0.683	**0.81**	**0.62**	**0.61**	0.294	0.6144	**0.641**	**0.7**	**0.44**	**0.576**

Table 5.2: Test set mAP of PASCAL VOC 2007. The first row is the baseline using Faster RCNN with number of regions to be tested equals 30. The second row is the Faster RCNN augmented with Bayesian Optimization search algorithm, a GP is used as the function prior and ExpSquared, $k(r^2) = \exp\left(-\frac{r^2}{2}\right)$, is used as the GP kernel.

better results, for our case the points are spatially correlated so choosing a kernel that easily capture and reflect on this property immediately gives better results.

5.1.2 Excluding RPN Regions

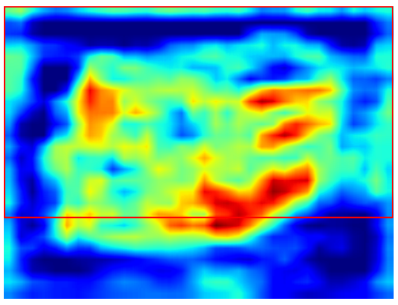
Figure 5.3 is depicting how ill-localized the best regions proposed by the RPN network in the Faster-RCNN framework, which serves as a good motivation behind augmenting the original proposed regions with the, likely to give higher detection score, regions suggested by the Bayesian Optimization search algorithm.

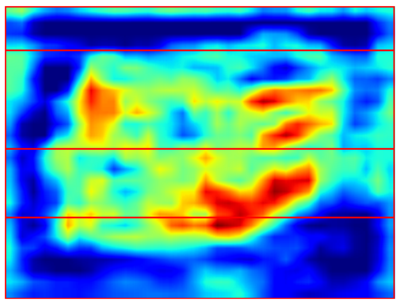
It can be observed in Figure 5.4, how the local maxima search algorithm using Bayesian Optimization can improve on the best regions originally proposed by the RPN.

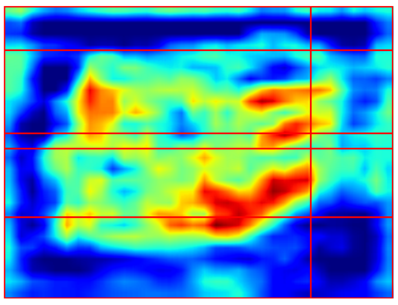
However in the section I present the results of totally excluding the RPN regions and only relying on the regions coming out of the Bayesian Optimization. The intuition would be that these regions are highly optimized to give a higher classification scores so dumping the RPN regions shouldn't affect the detection precision, yet as shown in Table 5.3 the mAP is significantly decreased, especially for classes that usually have multiple overlapping objects in the same scene, leaving us to believe that Bayesian Optimization may converge to a single box if the objects are highly overlapping, and that's why the GP domain construction parameter is very important to control this behaviour by limiting the space where the GP can sample new regions.

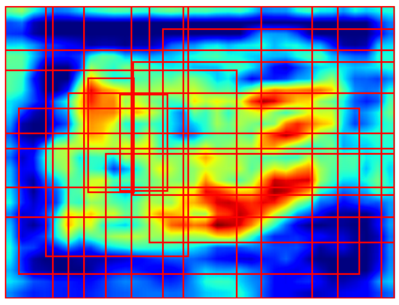
Model	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Faster-RCNN (ZF)	0.609	0.609	0.485	0.375	0.233	0.677	0.626	0.70	0.276	0.695	0.684	0.69	0.80	0.611	0.604	0.309	0.622	0.622	0.69	0.426	0.567
Faster-RCNN (ZF) + BO(Matern52)	**0.62**	0.60	0.42	**0.4**	**0.24**	0.61	0.54	0.69	0.27	0.62	**0.693**	0.68	0.72	0.61	0.53	0.3	0.54	0.62	**0.7**	**0.431**	0.542

Table 5.3: Test set mAP of PASCAL VOC 2007 after excluding RPN regions.


(a) Depicting the top scoring region


(b) Depicting the 3-top scoring regions


(c) Depicting the 5-top scoring regions


(d) Depicting the 200-top scoring regions

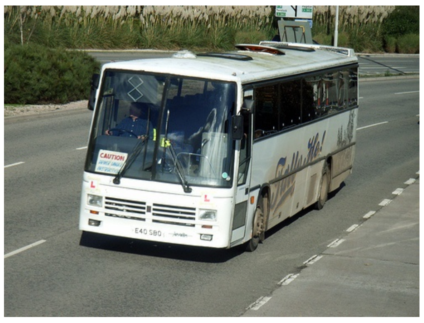

(e) Original picture

Figure 5.3: Projecting the best scoring region(s) to the last feature map in the convolution layers

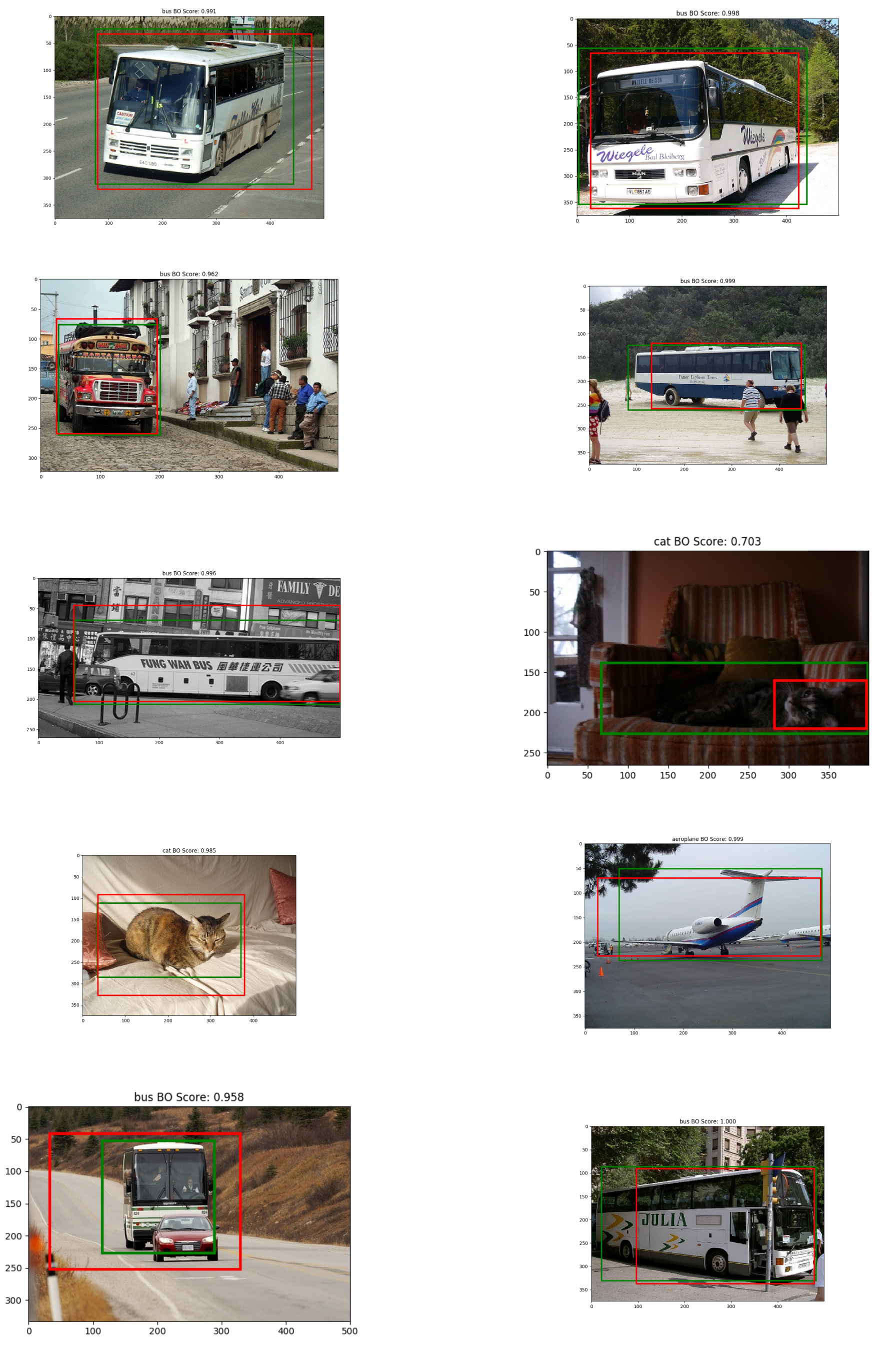


Figure 5.4: Red box depicting the final detection proposed by the RPN. Green box depicting the result after local maxima search using Bayesian Optimization.

Chapter 6

Conclusion and Future work

In this work I proposed a complementary method to improve the performance of the Faster-RCNN object detection framework, a fine-grained search algorithm using Bayesian Optimization to refine the proposed regions. I demonstrated the state-of-the-art performance on PASCAL VOC 2007 benchmark under the standard localization requirements.

With many knobs to tune, the method could yield better results by exploring different models as the object function prior, for example, bayesian neural network and random forests. In addition to choosing different models, still we could try different kernels with the Gaussian process as the function prior and/or hyper-parameter optimization including the IoU threshold used for the GP domain construction, number of iteration for a single optimization run, and number of optimization runs.

Bibliography

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[6] M Everingham, L Van Gool, CKI Williams, J Winn, and A Zisserman. The pascal visual object classes challenge 2007 (voc2007) results http://www. pascal-network. org/challenges. In *VOC/voc2007/workshop/index. html*, 2007.

[7] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[8] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[12] A. Klein, S. Falkner, N. Mansur, and F. Hutter. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian Optimization Workshop*, dec 2017.

[13] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[15] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103, 2011.

[16] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[18] J Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[20] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[21] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[22] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[23] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Statist.*, 21(1):124–127, 03 1950.

[24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[25] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[26] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[27] Yuting Zhang, Kihyuk Sohn, Ruben Villegas, Gang Pan, and Honglak Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–258, 2015.