# LOCALIZATION OF CERTAIN ANIMAL SPECIES IN IMAGES VIA TRAINING NEURAL NETWORKS WITH IMAGE PATCHES

**A Thesis Submitted to**
**the Graduate School of Engineering and Sciences of**
**İzmir Institute of Technology**
**in Partial Fulfillment of the Requirements for the Degree of**

**MASTER OF SCIENCE**

**in Computer Engineering**

**by**
**Semih ORHAN**

**December 2017**
**İZMİR**

We approve the thesis of **Semih ORHAN**

Examining Committee Members:

---

**Assoc. Prof. Dr. Derya BİRANT**
Department of Computer Engineering, Dokuz Eylul University

---

**Asst. Prof. Dr. Nesli ERDOĞMUŞ**
Department of Computer Engineering, İzmir Institute of Technology

---

**Assoc. Prof. Dr. Yalın BAŞTANLAR**
Department of Computer Engineering, İzmir Institute of Technology

**21 December 2017**

---

**Assoc. Prof. Dr. Yalın BAŞTANLAR**
Supervisor, Department of Computer Engineering
İzmir Institute of Technology

---

**Assoc. Prof. Dr. Yusuf Murat ERTEN**
Head of the Department of
Computer Engineering

---

**Prof. Dr. Aysun SOFUOĞLU**
Dean of the Graduate School of
Engineering and Sciences

# ACKNOWLEDGMENTS

I would like to thank to my supervisor Assoc. Prof. Dr. Yalın Baştanlar for all his effort. He was very supportive, caring, and helpful while writing my thesis. I accomplished this work with his sharing knowledge and his support.

I would like to thank my all friends and Computer Vision Research Group (CVRG) members for their friendship.

I also to thank Asst. Prof. Dr. Nesli Erdoğmuş and Assoc. Prof. Dr. Derya Birant, sparing their valuable time for evaluating this work.

I am grateful to my family for their love and support.

This thesis work is supported by The Scientific and Technical Research Council of Turkey (TUBITAK) with project number 115E918.

# ABSTRACT

LOCALIZATION OF CERTAIN ANIMAL SPECIES IN IMAGES VIA TRAINING NEURAL NETWORKS WITH IMAGE PATCHES

Object detection is one of the most important tasks for computer vision systems. Varying object size, varying view angle, illumination conditions, occlusion etc. effect the success rate. In recent years, convolutional neural networks (CNNs) have shown great performance in different problems of computer vision including object detection and localization. In this work, we propose a novel training approach for CNNs to localize some animal species whose bodies have distinctive pattern, such as speckles of leopards, black-white lines of zebras, etc. To learn characteristic patterns, small patches are taken from different body parts of animals and they are used to train models. To find object location, in a test image, all locations are visited in a sliding window fashion. Crops are fed to CNN, then classification scores of all patches are recorded. To illustrate object location, heat map is generated by the classification scores of the patches. Afterwards, heat maps are converted to binary images and end up with bounding box estimates of objects. The localization performance of our Patch-based training is compared with Faster R-CNN – a state-of-the-art CNN-based object detection and localization algorithm. While evaluating the performances, in addition to the standard precision-recall metric, we use area-precision and area-recall which represent the potential of Patch-based Model better. Experiment results show that the proposed training method has better performance than Faster R-CNN for most of the evaluated classes. We also showed that Patch-based Model can be used with Faster R-CNN to increase its localization performance.

# ÖZET


İMGE PARÇALARI KULLANILARAK EĞİTİLEN YAPAY SİNİR AĞLARI İLE İMGELERDE BELİRLİ HAYVAN TÜRLERİNİN KONUMLANDIRILMASI

Nesne bulma bilgisayarla görü sistemlerinin en önemli görevlerinden biridir. Değişen nesne boyutu, değişen bakış açısı, ortam aydınlatması, örtüşen nesneler ve benzeri etkenler başarım üzerinde etkilidir. Son yıllarda, Evrişimli Yapay Sinir Ağları (EYSA) birçok bilgisayarla görü problemlerinde (nesne konumlandırma ve nesne tespiti) çok iyi bir performans göstermiştir. Bu çalışmada, bedeni üzerinde ayırt edici bir desene sahip hayvanların, örneğin: benekli leoparlar, siyah beyaz çizgili zebralar gibi, konumunu bulmak için EYSA kullanan yeni bir yaklaşım öneriyoruz. Desen özelliklerini öğrenmek için, vücudun çeşitli bölgelerinden küçük parçalar alınır ve modelleri eğitmek için kullanılır. Test imgelerinde nesne konumunu bulmak için bütün konumlara kayan pencere yaklaşımı ile uğranır. Parçalar EYSA'na verilir ve tüm parçaların sınıflandırma skorları kaydedilir. Nesne konumlarını görselleştirmek için tüm parçaların sınıflandırma skorları kullanılarak sıcaklık haritası üretilir. Daha sonra sıcaklık haritaları ikili imgelere çevrilir ve nesneyi kapsayan kutu tahmini yapılarak süreç sonuçlanır. Önerdiğimiz Parça-tabanlı eğitim yönteminin nesne konumlandırma performansını EYSA kullanan güncel algoritmalardan biri olan Faster R-CNN ile karşılaştırdık. Performans değerlendirmesi yaparken standart kesinlik-anma metriğine ek olarak, Parça-tabanlı yöntemi daha iyi ifade ettiği için alan-kesinlik ve alan-anma metriğini de kullandık. Deney sonuçlarına göre önerilen eğitim yöntemi Faster R-CNN'e göre neredeyse değerlendirilen tüm sınıflar için daha iyi bir performans göstermektedir. Aynı zamanda, Parça-tabanlı yöntem Faster R-CNN ile kullanılarak Faster R-CNN'in konumlandırma başarısının artırılabileceği de gösterilmiştir.

# TABLE OF CONTENTS

# LIST OF FIGURES

**Figure** **Page**

# LIST OF TABLES

# LIST OF SYMBOLS

$P_{AR}$ . . . . . Area Precision

$R_{AR}$ . . . . . Area Recall

$PatchCont$ . . . . . Patch-based Model Contribution

# LIST OF ABBREVIATIONS

BoW . . . . . Bag of visual Words
NNs . . . . . Neural Networks
CNNs . . . . . Convolutional Neural Networks
R-CNN . . . . . Region-based Convolutional Neural Networks
RPN . . . . . Region Proposal Network
YOLO . . . . . You Only Look Once
ReLU . . . . . Rectified Linear Unit
SVM . . . . . Support Vector Machine
RGB . . . . . Red Green Blue
MLP . . . . . Multilayer Perceptron
IoU . . . . . Intersection over Union
EYSA . . . . . Evrişimli Yapar Sinir Ağı
ROC . . . . . Receiver Operating Characteristic
MNIST . . . . . Modified National Institute of Standards and Technology
ResNet . . . . . Deep Residual Network
TP . . . . . True Positive
FP . . . . . False Positive
FN . . . . . False Negative

# CHAPTER 1

# INTRODUCTION

In the last decades, visual data has grown exponentially. Thus computer vision applications have become a part of our society, such as self-driving car, autonomous robotic systems, image understanding, face detection, object tracking, etc. The core of these applications may require image classification, object detection, and image segmentation.

In our daily life, visual object classification arises whenever we meet an object. And also most of industrial works highly depend on image classification. Desired results are obtained by precisely classification of objects. To solve this problem, scientist have made great effort to develop several approaches [8], [9], [1], [43].

Object detection aims a true classification of an object and also well fitting to its bounding box. During 2000s, many algorithms have been proposed for object detection, such as Bag of visual Words (BoW) [3], Histogram of oriented gradients [4], Deformable Part Models [6]. In recent years, artificial neural networks became popular in visual object detection and other related tasks due to their great success.

Actually artificial neural networks is not a new research area. In 1943, McCulloch and Pitts built a model that demonstrate how neuron works in brain [19]. Computers became more sophisticated in 1950s, this improvement gave people to simulate theoretical neural networks. Marvin Minsky, who was founder of MIT AI Lab, and Seymour Papert wrote a book that is related to analysis on limitation of Perceptrons [20]. In this book, this approach of AI was thought to have a dead-end due to lack of trace on the system and its critical nature. This conclusion caused to freeze funding and publication to AI. Most people believed that this paper caused a AI winter. Paul Werbos proposed that backpropagation can be used in neural networks [41]. He has solved how to train multilayer neural networks in his PhD thesis. But due to the AI winter, it required a decade for researchers to work in this area. In 1986, this approach became popular [32]. First time in 1989, it was applied to a computer vision task which is handwritten digit classification [14]. It has demonstrated an excellent performance. Again it took more than a decade for computers to handle more complex tasks and to learn from huge amount of image data.

An outstanding performance was observed in 2012. AlexNet [13] got the $1^{st}$ place

in ImageNet 2012 classification task with achieving 16.4% error rate. There was a huge difference between $1^{st}$ place (16.4%) and $2^{nd}$ place (26.1%). Several factors were responsible for gaining this outstanding performance. (i) Training dataset reasonably extended, (ii) GPU computing has been used, (iii) better training method which employs "dropout" [13] has been implemented.

## 1.1. Related Works

Object detection with CNNs is a highly popular research area. There exists many approaches, for instance OverFeat [34], Faster R-CNN [30], and Oquab et al.[25]. We are especially interested in the ones that focus on object localization task. In [34], OverFeat, convolutional neural network was trained as a classifier first. After that, softmax layer was removed and regression layer was implemented to be able to estimate object location. Objects are searched using efficient way of sliding windows. In R-CNN [7], potential object regions are proposed by an algorithm called 'selective search' [40] before classification. Faster R-CNN [30] implemented region proposal step as a neural network, called Region Proposal Network (RPN), which reduced the region proposal time significantly. Objects are searched only in the proposed regions. In [24], Oquab et al. explained that CNNs, even trained for image classification, carry good amount of information of object location. By using this observation, they proposed an approach to detect object location using CNNs [25]. They trained their algorithm using weakly-supervised learning, instead of supervised learning. In supervised learning systems, object location is explicitly defined. This helps algorithm to learn more explicit way. But annotation of millions of boxes are not feasible for large-scale works. Unlike supervised learning, only contained object label is given to the weakly-supervised learning systems.

Proposed after Faster R-CNN, You Only Look Once (YOLO) [28], did not increase the detection performance but reached to real time object detection speed. YOLO uses single CNN for both classification and object detection. Input image is divided into grid cell, and you get $n$ bounding box prediction for each cell. Confidence score defines how far we are close to predict an object in bounding box. Each bounding box, and each cell also predict a class score. Confidence score and class score are combined, and final score is obtained.

## 1.2. Thesis' Aim and Objectives

In this work, we propose to find object location using a state-of-the-art CNN [11] and Patch-based Training Method. Patch-based Training Method is based on training neural network with object patches. Some example patches can be seen at Fig.1.1. To find object location, in a test image, all locations are visited in a sliding window fashion. Crops are fed to CNN, then classification scores of all patches are recorded. To illustrate object location, heat map is generated by the classification scores of the patches. After that, heat maps are converted to binary images at varying thresholds. Final object location estimates are generated after employing a connected component analysis algorithm on binary images.

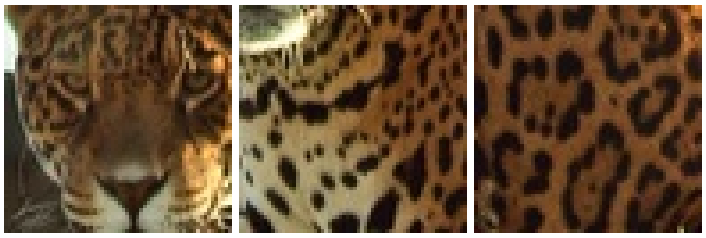


Figure 1.1. Example Patches from Dataset.

Our experiment results showed that Patch-based Training approach has better performance for area-precision and area-recall metrics than Faster R-CNN. We also observed that Patch-based Model can be employed with Faster R-CNN to increase its localization performance. We will call it "Combined Model". Faster R-CNN success is increased by the help of Patch-based training method.

## 1.3. Organization of Thesis

The organization of the rest of this thesis is as follows: Chapter 2 provides basic information about NNs; difference between CNNs and NNs; different types of activation functions and CNN architectures. How Patch-based Model works is described in Chapter 3. Experiments and evaluation metrics are given in Chapter 4. Conclusions are given in Chapter 5.

# CHAPTER 2

# BACKGROUND

This chapter provides basic information about the structure of neural networks (NNs) and different types of NNs, which will be beneficial to understand the following chapters. Neural networks are inspired by brain. They are composed of a number of neurons and layers. First NN which is Perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt [31]. Single layer perceptron can only work on linearly separable problems due to its linear nature. To overcome this problem, multilayer perceptron (MLP) was invented. It consists of three or more layers, and uses non-linear activation functions. An example architecture of multilayer perceptron can be seen at Fig.2.1.

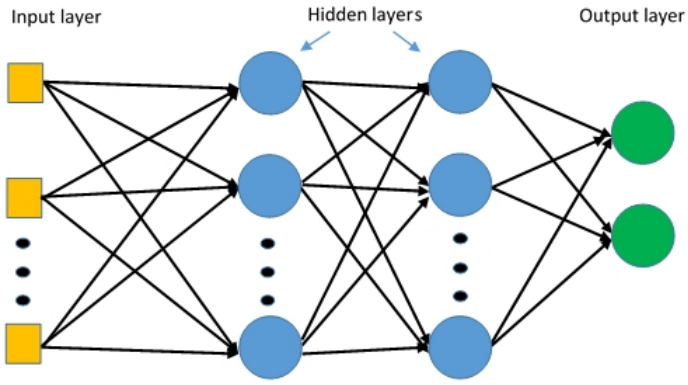



Figure 2.1. Multilayer Perceptron [21].

Neurons are one of the key component for NNs. At some point they can be fired, like in brain, using with activation functions. Activation functions are also known as transfer functions. They transfer input values to output values in specific manner. For example, given inputs and outputs can be seen at Fig.2.2. In this figure, Leaky Rectified Linear Unit (Leaky ReLU) function was used, but there are also different types of activation functions, such as sigmoid, tanh, ReLU, etc. Output of neurons may or may not give a continuous value (that can be transmitted to next neuron of other layers). These structure leads neural networks to have highly non-convex structure. Having highly non-convex structures

cause many local minima and it makes algorithm hard to train. Because we may stuck at a minima which may not be a good one (compared to global), it makes neural networks hard to generalize. And its high complex structure require so much computation power for large scale works. In NNs, each input pixel is connected to each neuron weights. This phenomenon is not an efficient way of using weights when the input is an image. First, the number of weights increase enormously when each pixel is a neuron. Second, close pixels in images contain similar information. Inspiring that idea, instead of connecting each input pixel to weights, convolution operation is applied on input pixels. This phenomenon drastically decrease the number of weights and makes computation much faster. For the last ten years, thanks to advancement in computation power and approach, they enable us to train CNNs on large scale data in less required time. Basic CNNs compose of convolutional layers, fully connected layers, activation functions, and loss functions (SVM, softmax).

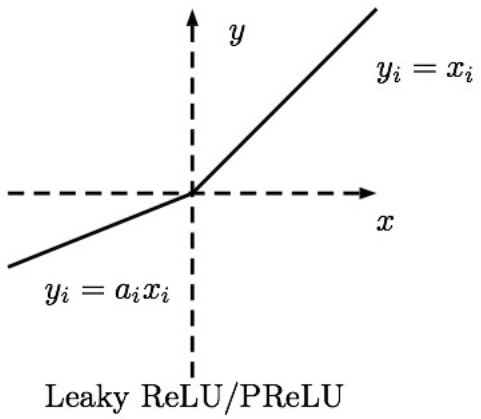


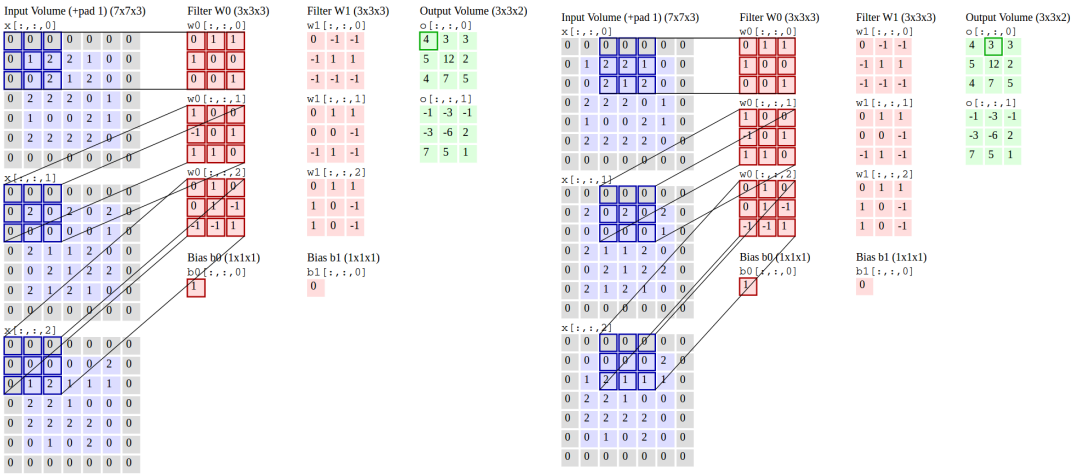

Figure 2.2. Leaky ReLU [15].

## 2.1. Layers of CNNs

### 2.1.1. Convolutional Layer

Convolutional layer is a core building of convolutional neural networks. It contains plenty of learn-able filters (or kernels). Each filter is convolved across width and height of input images. At the end of training process, filters of network are able to

identify specific types of images which contain same types of shapes, same type of spatial positions, etc. One mathematical example is given to illustrate how convolutional layers work. In this example, 5x5 RGB image is given to the Neural Network. It is convolved with two kernels that are 3x3x3 (height, weight, and depth). Convolution is applied with stride size of 2. During the convolution, zero padding is added to enlarge image size. First convolution operation can be seen at Fig.2.3(a). Element wise addition is applied in each convolution phase. For example in the first convolution, result is $[(0\times0+1\times0+1\times0+1\times0+0\times1+0\times2+0\times0+0\times0+1\times2)+(1\times0+0\times0+0\times0+-1\times0+0\times2+1\times0+1\times0+1\times0+0\times0)+(0\times0+1\times0+0\times0+0\times0+1\times0+(-1)\times0+(-1)\times0+(-1)\times1+1\times2)+1(bias)=4]$.




(a) (b)

Figure 2.3. (a) First convolution operation applied with filter W0. Computation gives us the top-left member of next layer, (b) Second convolution operation. Again applied with filter W0. Stride is equal to 2 [2].

## 2.1.2. Fully Connected Layer

The fully connected layer is a traditional Multi Layer Perceptrons (MLP). Each neurons of layer is connected to next layer of each neurons. Last convolutional layer contains high dimensional features. An example of neural network is shown at Fig.2.4. In this example, we have four classes. Output of last convolutional layer is connected to

fully connected layer. Final layer contains low-dimensional data, which can be given to softmax function for classification.

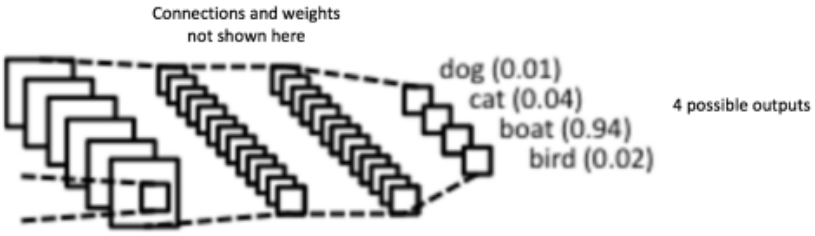



Figure 2.4. Neural Network [22].

### 2.1.3. Activation Functions

Neural network is inspired from how a neuron works in brain. Due to this reason, we need an activation function to make it similar. Activation function calculates weighted sum of input neurons and then adds a bias. This operation can be seen at Eq.2.1.

$$y = f(\sum w_i * x_i + b) \tag{2.1}$$

where $y$ is output, $w_i$ is weight of neurons, $x_i$ is inputs, and $b$ is bias. We need to decide at some point whether neuron is activated or not.

#### 2.1.3.1. Step Function

This is generally used by Perceptrons. If input value is higher than threshold, we get output $1$. This characteristic can be seen at Fig.2.5. So it can activate/fire the neuron. This function generally used for binary classification. There is a major drawback. What if we have more than two classes? Let's assume that we have four different classes which are cat, dog, boat and bird. Given input may be activated for more than one classes, such as: cat, dog, and boat classes can be activated. At this point, we can not decided what certain input class is. To solve this problem we need an analog activation function.

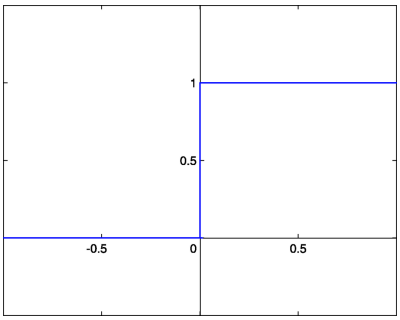



Figure 2.5. Step Function [37].

### 2.1.3.2. Linear Function

Function output is linearly proportional to input value (Fig.2.6). The problem with the linear function is that the gradient of the function is constant. When there is a loss, we get the constant gradient value without considering of its input value.

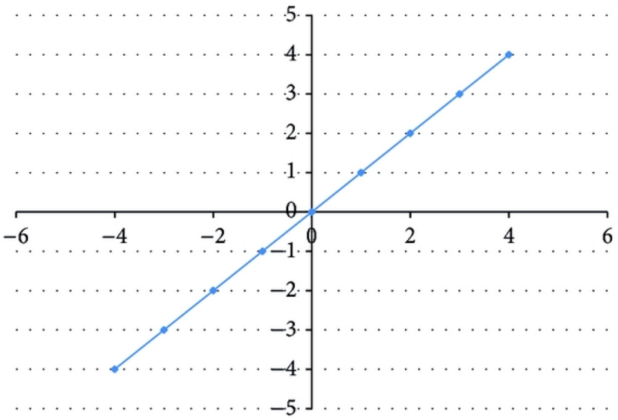



Figure 2.6. Linear Function [17].

We can not get a non-linear function using with combinations of linear functions. In most cases, linear function is not good enough to solve complex problem.

### 2.1.3.3. Sigmoid Function

Sigmoid is a non-linear function (Fig.2.7) which means that combination of function is also non-linear. It also squash output value between $0$ to $1$. It is good, because in linear function, output value can be anything from $-inf$ to $+inf$. At horizontal axis between $-2$ to $+2$, output value significantly change. However, there are some problem exist with sigmoid function too. Rate of change is too small after value $4$ with respect to Fig.2.7. This low rate of change cause to vanishing gradient problem. Second problem is that gradient calculation of sigmoid function is not easy.

$$y = f(wx) = \frac{1}{1 + e^{-wx}} \tag{2.2}$$

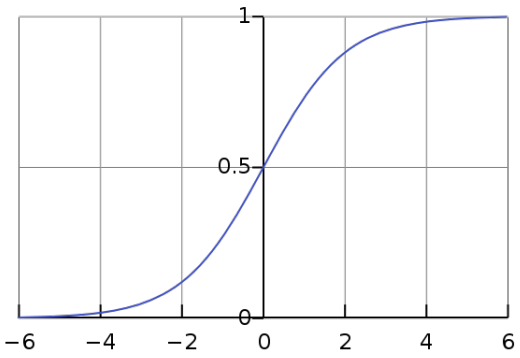



Figure 2.7. Sigmoid Function [35].

### 2.1.3.4. Tanh

Tanh is another activation function of neural networks. Its characteristic is similar to sigmoid function expect its range between $-1$ to $+1$. Rate of change is stronger than sigmoid function. Again it is widely used.

$$y = f(wx) = tanh(wx) = \frac{2}{1 + e^{-2wx}} - 1 \tag{2.3}$$

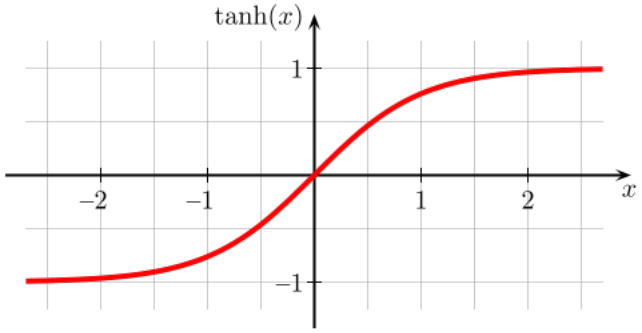



Figure 2.8. Tanh Function [39].

### 2.1.3.5. Rectified Linear Unit

Rectified Linear Unit (ReLU) is another activation function of neural networks. Its characteristic can be seen at Fig.2.9. ReLU is a non-linear function. Output value can be between $0$ to $+inf$. While using sigmoid and tanh function, almost all neurons of network one way or another are activated. It is costly. We would like to activate neurons sparsely. ReLU enable us to use network more effective manner in which some neurons are not activated while random initializing. Its computation cost is less than sigmoid and tanh. A problem with ReLU function is that, while calculation gradient and updating weights, values horizontal line does not respond to gradient and important part of network may become not updating during backpropagation. This problem is called dying neurons. This problem can be partly achieved using with Leaky ReLU. Leaky ReLU formula can be seen at Eq.2.5, where $\wp$ is a small constant.

$$f(x) = x^{+} = max(0, x) \tag{2.4}$$

$$f(x) = 1(x < 0) * (\wp x) + 1(x >= 0)(x) \tag{2.5}$$

## 2.1.4. Max-Pooling Layer

Max-Pooling layer is commonly used between convolutional layers to reduce parameter sizes. It makes algorithm much faster, and also it helps us to prevent from over-

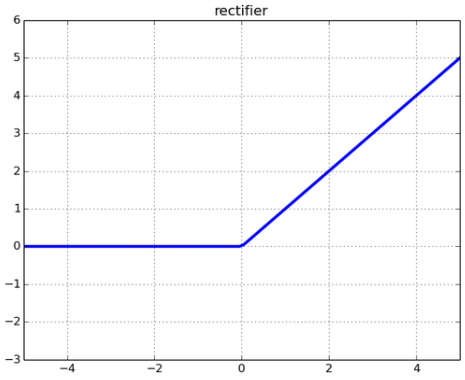



Figure 2.9. Rectified Linear Units [29].

fitting. Example input image, whose size are $4x4$, and its output size after applying max-pooling operation is shown at Fig.2.10.

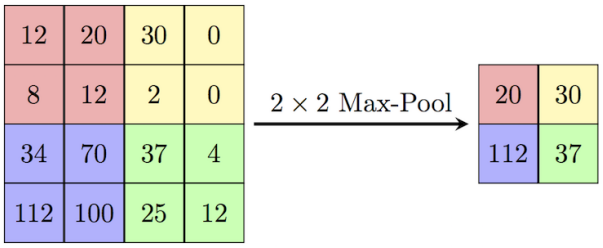



Figure 2.10. Max Pooling Operation [18].

### 2.1.5. Dropout

Dropout is a simple regularization technique for neural network. Randomly chosen some neurons are dropped out during training. It helps us to prevent overfitting. This learning method is similar to ensemble learning. Instead of generating one solid network, different networks are generated dropping out some neurons during the training. After end of the training, all neurons are combined for testing.

## 2.2. Different Architectures of Convolutional Neural Networks

### 2.2.1. Standard Convolutional Neural Networks

Convolutional Neural Networks (CNNs) consist of convolutional layers, max-pooling layers, fully connected layers, and loss function (softmax/SVM). CNNs are similar to standard neural networks. Instead of connecting weights to all input pixels, convolution is applied on each input. It enable us to scale weights in more efficient way. An example of a standard CNN can be seen at Fig.2.11.

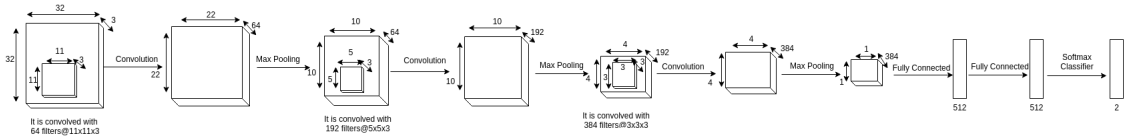



Figure 2.11. Architecture of Standard CNN [26].

### 2.2.2. Vanishing Gradient Problem

After observing huge success of convolutional neural network [13], number of layers in neural networks have increased year by year [36][38]. Is it possible to increase number of layer in neural networks as we want? Answer of this question, accuracy of neural networks which have more than tens of layers is going to saturated after few iterations. How depth affects to neural networks is examined in [23]. Example code can be found at following link: https://github.com/mnielsen/neural-networks-and-deep-learning.git. Model is trained on Modified National Institute of Standards and Technology (MNIST) handwritten dataset [16]. Several models, which have different number of layers, have been trained. Classification accuracy can be seen at Table 2.1.

It is observed that, more than few layers does not have positive effect on accuracy. Accuracy of model even may decrease with respect to its depth. To get an idea about why vanishing gradient problem is occurs, simple neural network is given at Fig.2.12.
where $w_1, w_2, ...$ are weights, $b_1, b_2, ...$ are biases. $a_j$ is $\sigma(z_j)$, where $\sigma$ indicates activation function which can be sigmoid, rectified linear unit etc., and $z_j = w_j * a_{j-1} + b_j$ is

| Classification accuracy % | |
|---|---|
| Model that has 1-hidden layer | 96.48 |
| Model that has 2-hidden layers | 96.90 |
| Model that has 3-hidden layers | 96.57 |
| Model that has 4-hidden layers | 96.53 |

Table 2.1. Classification accuracy of models that are trained with MNIST dataset.

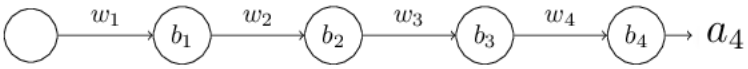



Figure 2.12. Simple Neural Network [23].

weighted input to neuron (Fig.2.12). If predicted output, which is $a_4$ in this example, is close to actual output then cost will come near to zero. Otherwise, it will be high. Gradient equation that is related to first hidden neuron is shown below, where $C$ is the cost function.

$$\frac{\partial C}{\partial b_1} = \sigma' z_1 \times w_2 \times \sigma' z_2 \times w_3 \times \sigma' z_3 \times w_4 \times \sigma' z_4 \times \frac{\partial C}{\partial a_4} \tag{2.6}$$

Derivative of sigmoid function equals to maximum 25% of its previous value (Fig.2.13). Thus, weights are usually satisfy at $|w_j \times \sigma'(z_j)| < 1/4$. As a result of that, products are decrease exponentially.

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times \overbrace{w_2 \times \sigma'(z_2)}^{1/4} \times \overbrace{w_3 \times \sigma'(z_3)}^{1/4} \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4} \tag{2.7}$$

As shown in Eq.2.6, magnitude of weights is decrease to 25% of its previous value at end of each layer. So, gradient of $\partial C/\partial b_1$ usually become 16 times smaller (or bigger) than gradient of $\partial C/\partial b_3$. As a result, vanishing gradient phenomenon occurs and NNs starts learning very slow.

To solve this problem, ReLU activation function is used mostly for todays CNNs. Derivation of ReLU is shown at Eq.2.8. While backpropagation, according to Eq.2.8, there is no attenuation of gradient that is caused by ReLU function.

$$f(x)' = \begin{cases} 0, \ for \ x < 0 \\ 1, \ for \ x \geq 0 \end{cases} \tag{2.8}$$

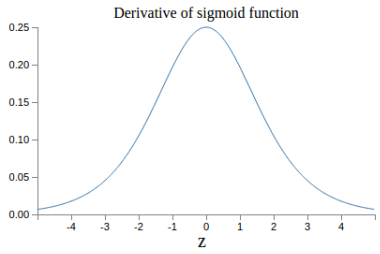



Figure 2.13. Derivation of Sigmoid Function [23].

### 2.2.3. Deep Residual Network

Adding tens of layers makes NNs harder to optimize, and that cause higher training error rate. It has been reported with increased depth in networks accuracy not only saturates but also degrades after a point [10]. This is called as 'degradation' problem and this is not caused by overfitting. To deal with degradation problem, deep residual learning was proposed [11]. Deep residual network consists of many "Residual Units". Residual block can be applied at regular intervals. General equation is shown at Eq.2.9 [12]:

$$y = F(x, W_i) + x \tag{2.9}$$

where $x$ is the input and $y$ is the output vector. $F(x, W_i)$ symbolize residual block. Operation of $F(x) + x$ is an element-wise addition. The formulation of $F(x) + x$ can be seen at Fig.2.14[11].

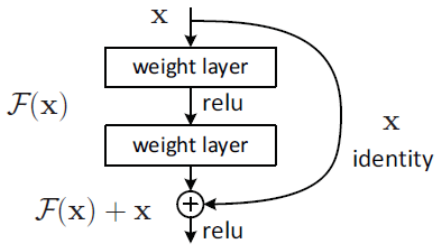



Figure 2.14. Residual Learning block [11].

Residual block does not increase the complexity of the model. This is the major advantage of residual learning. Both dimension $x$ and $F(x)$ should be equal to apply element-wise addition. If dimension of $x$ and $F(x)$ are not equal, lower dimension can be

increased with this Eq.2.10 [11]:

$$y_l = F(x_l, W_l) + W_s x_l \tag{2.10}$$

$F(x_l, W_l)$, the residual block, can be made by any number of layers. Generally, it is used with two or three layers. In our work, we employed ResNet [11], a deep residual network, for our Patch-based Training approach.

### 2.2.4. Faster R-CNN

Faster R-CNN [30] is a state-of-the-art CNN-based object detection and localization approach. It basically combines a region proposal network (RPN) with a CNN-based classifier. Instead of sliding window on whole feature maps at classification step, only proposed regions are classified in Faster R-CNN. It makes algorithm much faster than previous approaches ([25], [34]). 3x3 sliding windows are applied on the last convolutional layer of CNN. Each sliding window generates $k$ anchor boxes. With different scales and different width-height ratios, all these $k$ boxes are centered at the location of the current sliding window. At each location, a $256$-d feature vector is computed. Two estimations are performed using this vector; a classification score (being object or not) and a regression value (object may shift to right/left with respect to the reference location or may get smaller or bigger in $x$ or $y$ direction with respect to the reference size). These operations are illustrated in Fig.2.15. The procedure explained above generates about $20000$ region proposals for an image. Most of these are eliminated with a series of steps according to their objectness scores and their intersection areas with each other. Classification is performed for the remaining regions only. It helps to decrease computation load. Whole architecture of Faster R-CNN is shown at Fig.2.16.

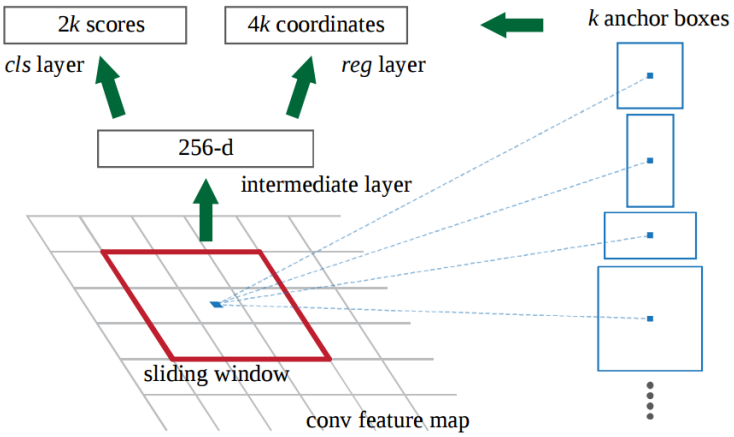



Figure 2.15. RPN, which is placed on top of the last convolutional layer of CNN [30].

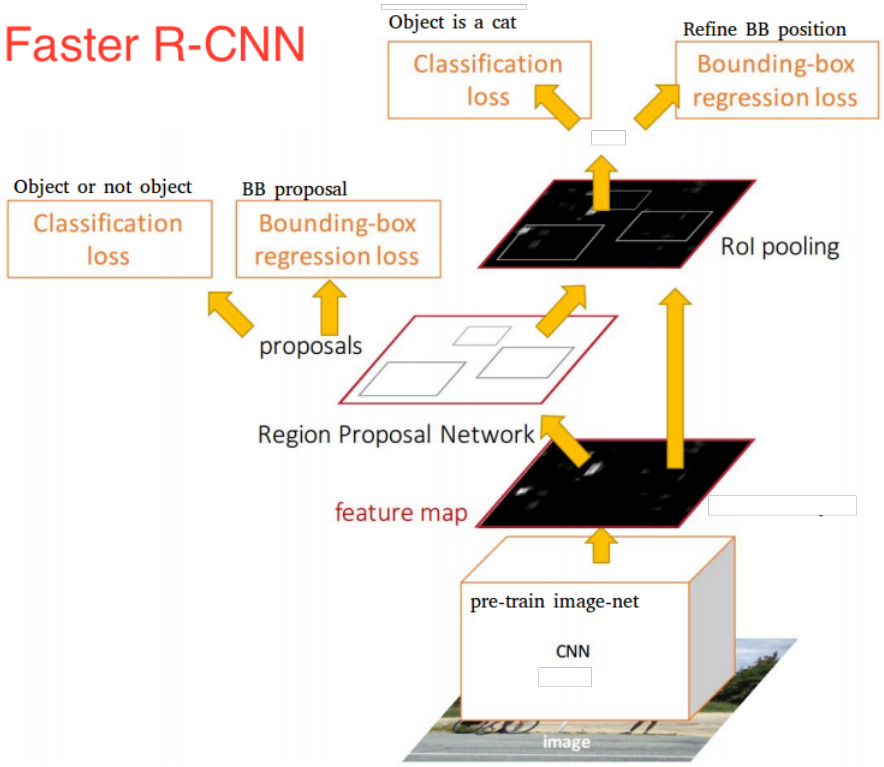



Figure 2.16. Faster R-CNN Architecture [5].

# CHAPTER 3

# PATCH BASED TRAINING FOR OBJECT LOCALIZATION

ResNet, a deep residual network, is selected as the CNN-based classification algorithm in our work. Firstly, we evaluated its image classification performance with experiments. To do that, we chose three different networks. First two of them are plain networks which contain a few convolutional layers [26], third one is ResNet [11] with different depths. Plain-Network-A (a very similar architecture can be seen in Fig.2.11) consists of 3 convolution, 2 fully connected layers. Plain-Network-B consists of 5 convolution, 2 fully connected layers. To train these models, two classes which are leopard and background were identified. Images are taken from ImageNet dataset [33]. We used 700 images for each class to train the models, and we tested their performance on 120 leopard, 120 background images. To evaluate the performance for all possible threshold values, we plot Receiver operating characteristic (ROC) curves for all models. ROC curves are shown at Fig.3.1. There is an obvious performance difference between plain networks and ResNet. ResNet with 56 layers had a perfect performance. Since it has such high classification performance, we decided to use ResNet for our Patch-based Training approach.

## 3.1. Dataset Preparation & Training

To find location of animals using Patch Based Model, 5 different classes are defined which are leopard, zebra, elephant, bear, and background. Stride size is chosen as 32 pixels, and $64$x$64$ crops are taken from bodies of animals. Example patches can be seen at Fig.3.2. Each class approximately contains one thousand image patches. Negative class (which is background) patches are taken from the same images but from the regions that do not contain any object parts. To train Patch Based Model, ResNet-Depth-50 model [11] is chosen as classifier network. We use MxNet as framework.[1]

[1] https://mxnet.apache.org/

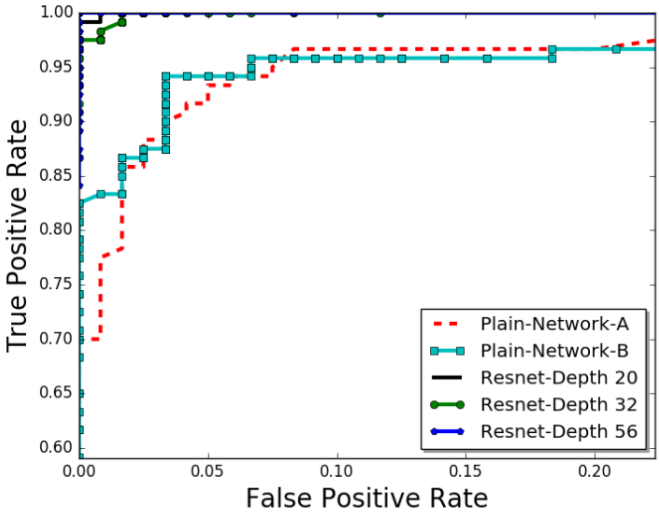



Figure 3.1. ROC Curve of Trained Models.

## 3.2. Generating Heat Map

To find the relevant patches in a test image, sliding window approach was applied. For Patch-based Model, 64x64 patches whose stride size of 32 pixels, were given ResNet-Depth-50 [11] as inputs. For each patch, probability of belonging to one of the trained classes is saved and then heat map is generated for a target class with respect to these results. Example input image and generated heat map can be seen at Fig.3.3. Red color which has the highest score means that location has been classified as the target animal for all encompassing sliding windows. Blue color means that none of the sliding windows including that image location was classified as the target. Heat map is generated for each class for each input images. Maximum probability value of each 32x32 pixel area can be 4 due to intersection of four boxes. After that, number of 4 is normalized to 1.

## 3.3. Estimating the Bounding Box of an Object

To draw a bounding box estimate of an object, heat map of an image converted to binary image according to the threshold values (scores) used in the test. Some image processing techniques are benefited in this part. To eliminate small dot noises: opening morphological operation is used. To connect separated parts: closing morphological oper-

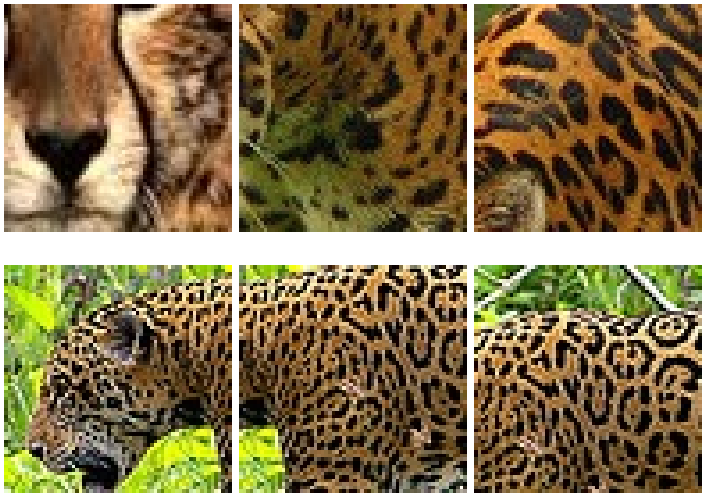
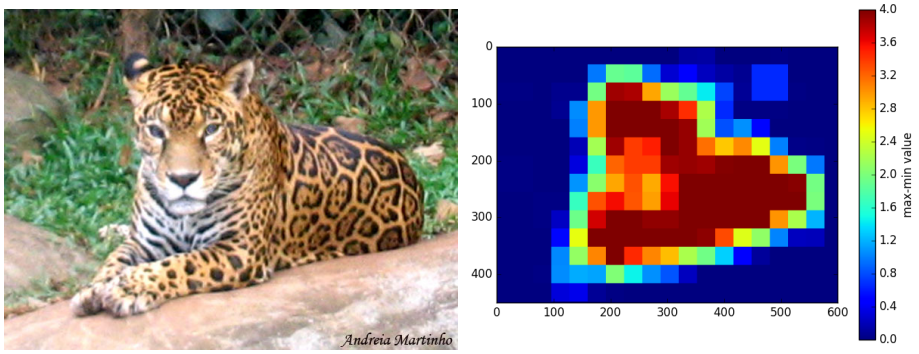


Figure 3.2. Instance of Patch-based Training Dataset.




(a) (b)

Figure 3.3. (a) An example input image (600x450 pixels), and (b) its generated heat map.

ation is applied. After applying morphological operations, connected component analysis algorithm is used to find object contours. Whole process can be seen on an example in Fig.3.4, and as a flowchart in Fig.3.5. Some example input images and their generated heat maps can be seen at Fig.3.6.

The work presented in Chapter 3 was presented in a conference [27] just for the leopard class. Later on, three other animals classes are added and the experiments, results of which are given in Chapter 4, are conducted.

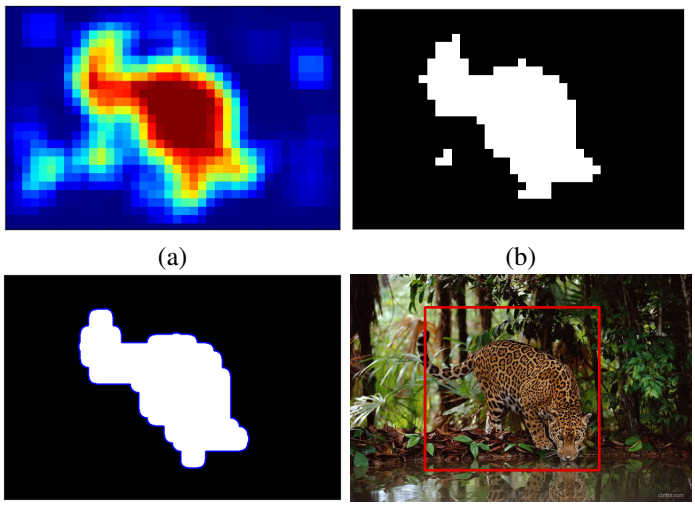



Figure 3.4. (a) Heat map of an input image, (b) binary image of the heat map , (c) result of opening and closing morphological operation, (d) predicted object bounding box.

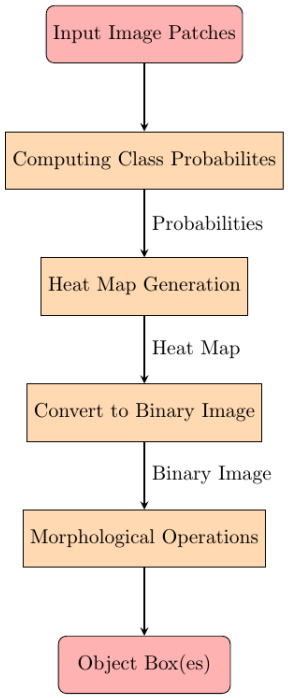



Figure 3.5. Flowchart of Patch-based Model for object detection.

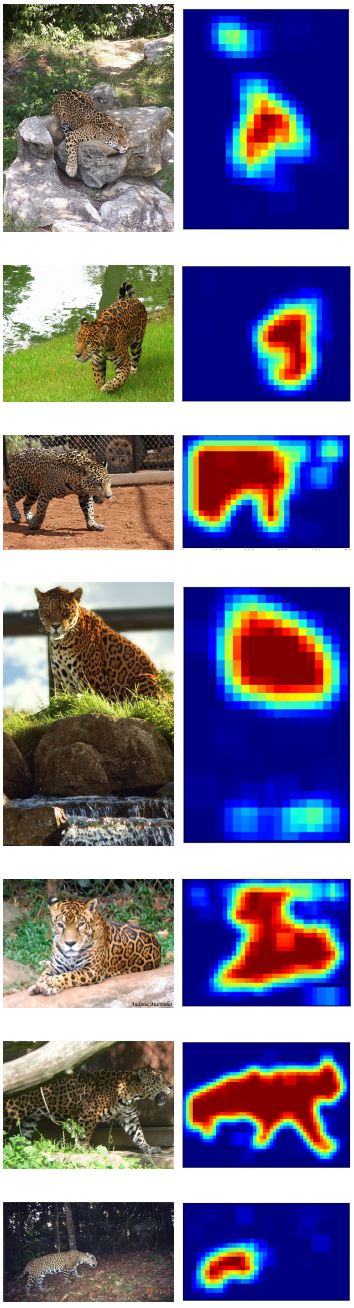


Figure 3.6. Input images are shown at first column, generated heat maps by Patch-based Model are shown at second column.

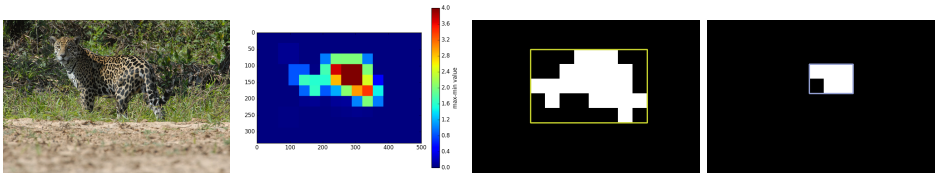
# CHAPTER 4

# EXPERIMENTS

In this chapter, we compare the results of proposed Patch-based training method with Faster R-CNN [11]. Also we present the improvement that could be gained by combining these two approaches. Section 4.1 explains the metrics that we used for evaluation which serve as background to understand the following subsections.

## 4.1. Evaluation Metrics

To evaluate performance of object detection algorithms, generally precision-recall metric is used in literature. Precision-recall formula can been seen at Eq.4.1. Detected object is classified as a true positive if it is correctly labeled and Intersection our Union (IoU) rate is higher than $0.5$. IoU formula can be seen at Eq.4.2. Faster R-CNN, and most of object detection algorithms, have high precision sensitivity while generating object boxes at high threshold values. Their box sizes neither shrink nor enlarge with respect to confidence value of boxes. Unlike general object detection algorithms, box sizes of Patch-based Method change according to threshold values. At low confidence scores: it has bigger boxes; at high confidence scores: it has smaller boxes. If we detect too big boxes, according to IoU equation (Eq.4.2), we get false positive. If we detect too small bounding boxes, again we likely to get false positive. To illustrate what is happening more precisely, with respect to different threshold values, example of detected boxes can be seen at Fig.4.1. This causing bumping effect on the precision-recall curve. Example output can be seen at Fig.4.2. As threshold increases, after a point, precision starts to decrease for Patch-based training since high confidence scores are always obtained for small boxes and they tend to be eliminated by IoU criterion. According to the definition of precision metric, at higher threshold values we need to localize objects more precisely. But this definition does not fit to Patch-based training method. Even if we more precisely localize an object, we get false positive according to Eq.4.1.



(a) (b) (c) (d)

Figure 4.1. (a) Input Image, (b) Generated Heat Map (maximum value four indicates that four boxes can be overlapped at the same pixel), (c) Predicted Bounding Box at threshold value 30, (d) Predicted Bounding Box at threshold value 90.

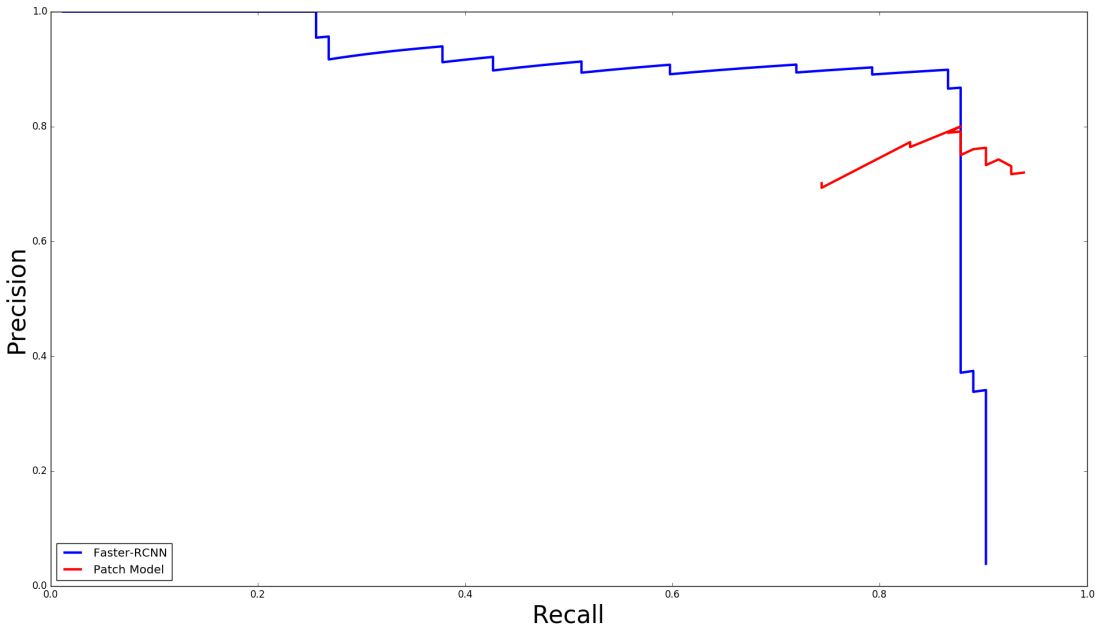



Figure 4.2. Precision-Recall Curve on Leopard Images.

$$\begin{aligned} \textit{Precision} &= \frac{TP}{TP+FP} \\ \textit{Recall} &= \frac{TP}{TP+FN} \end{aligned} \tag{4.1}$$

$$\textit{IoU} = \frac{Box_{\text{predicted}} \cap Box_{\text{groundtruth}}}{Box_{\text{predicted}} \cup Box_{\text{groundtruth}}} \tag{4.2}$$

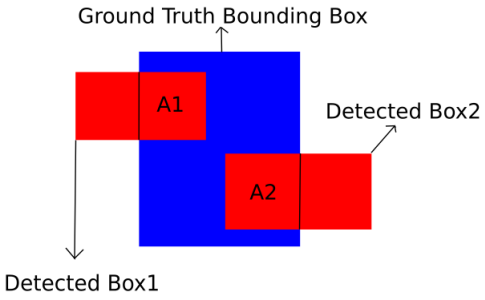



Figure 4.3. Area Precision & Area Recall Example.

$$P_{AR}(G, D) = \frac{Area(G \cap D)}{Area(D)}$$
$$R_{AR}(G, D) = \frac{Area(G \cap D)}{Area(G)} \tag{4.3}$$

Although high confidence score windows by Patch-based model fit in the object ground truth boxes, they are considered as false positive. Therefore, this precision-recall metric is not very appropriate to present the potential of Patch-based approach. As a metric suitable for our case, we used area-precision & area-recall metric [42] that is shown at Eq.4.3. $G$ is a ground truth rectangle, where $D$ is a list of detected rectangles, where $D_j = 1, ..., |D|$. In $Recall$ axis, proportion of detected ground truth area is measured. In $Precision$ axis, proportion of true and false overlapped area is measured. An area-precision & area-recall example given in Fig.4.3 to make it more clear. Area-precision & area-recall calculation of Fig.4.3 is shown below:

$$P_{AR}(G, D) = \frac{Area(A_1 + A_2)}{Area(DetectedBox_1 + DetectedBox_2)}$$
$$R_{AR}(G, D) = \frac{Area(A_1 + A_2)}{Area(GroundTruth)}$$

Precision-recall metric (Eq.4.1) works like a binary classification, given input is classified as true-positive (TP) or false-positive (FP), whereas area-precision & area-recall metric works like an analogue classifier. There is no true or false values.

## 4.2. Test Sets

### 4.2.1. Testing on Two Classes

Training set composes of 400 leopard, 450 zebra images for Faster R-CNN Number of images used for Patch-based Training is 50 for leopard and 50 for zebra classes. From these images approximately a thousand patches are extracted for each class. We tested performance of patch based training model and Faster R-CNN on 82 leopard, and 61 zebra images. Area-precision & area-recall curve of leopard class can be seen at Fig.4.4, area-precision & area-recall curves of zebra class is shown at Fig.4.5. According to area-precision & area-recall curves, Patch-based Training Method has show better performance than Faster R-CNN for both classes.

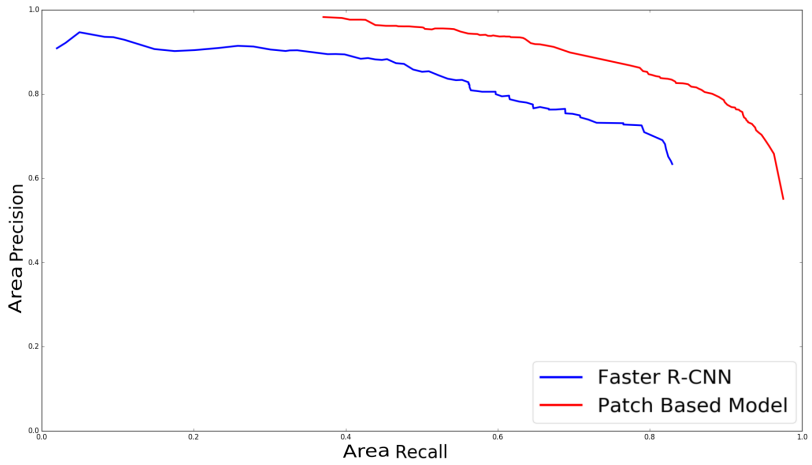



Figure 4.4. Area-Precision & Area-Recall Curve on Leopard Images.

### 4.2.2. Testing on Four Classes

We extended our first dataset to measure performance on different classes. Second dataset consists of bear, elephant, leopard, and zebra classes. Bear and elephant do not have distinctive patterns on their body as leopard and zebra do. These classes are intentionally chosen to observe if there is a performance decrease for them. Faster R-CNN is trained with 446 bear images, 351 elephant images, 400 leopard images, and 450 zebra

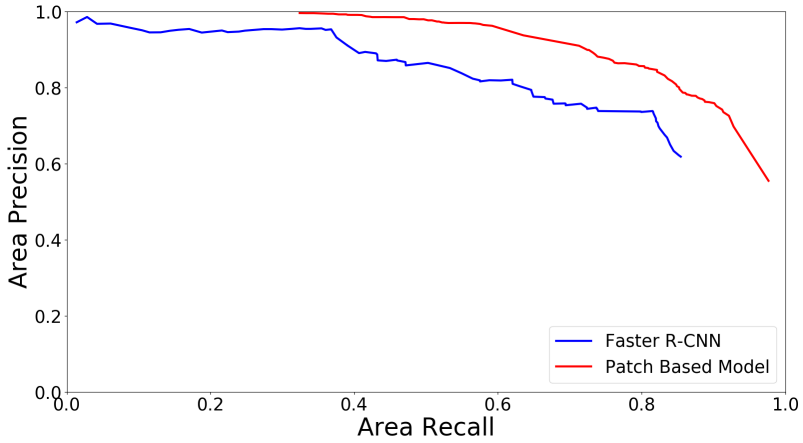



Figure 4.5. Area-Precision & Area-Recall Curve on Zebra Images.

images. Patch Based Training Model requires much less data than Faster R-CNN. Number of 35 bear images, number of 20 elephant images, number of 50 leopard images, and number of 50 zebra images are used to train Patch-based Model. Again, approximately a thousand image patches are extracted for each class. Area-precision & area-recall curves of bear, elephant, leopard, and zebra classes ares show at Fig.4.6, Fig.4.7, Fig.4.8, and Fig.4.9 respectively.

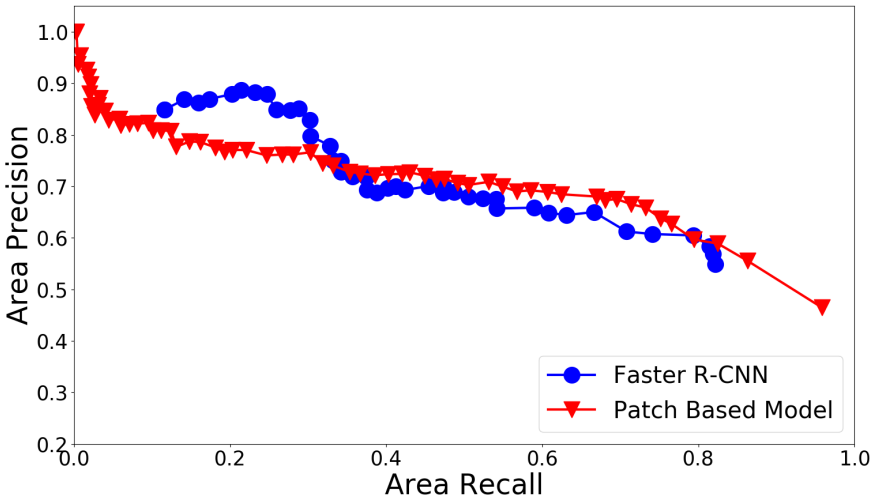



Figure 4.6. Area-Precision & Area-Recall Curve on Bear Images.

Patch-based Training Model outperform Faster R-CNN for elephant, leopard and zebra classes. Both model did not perform well on bear images; Faster R-CNN has shown slightly better performance than Patch-based Training Model. Some false predicted ex-

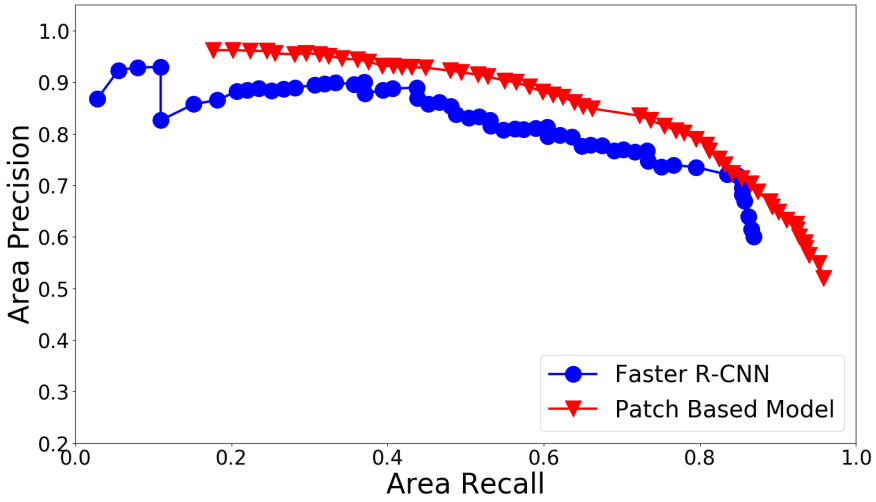



Figure 4.7. Area-Precision & Area-Recall Curve on Elephant Images.

amples of Patch-based Training Model are shown at Fig.4.10, Fig.4.11, Fig.4.12, and Fig.4.13 respectively. In Fig.4.10, Fig.4.11, and Fig.4.12, Patch-based Model prone to predict bear patches as elephant class patches. In Fig.4.11 and Fig.4.13, it is also a little bit confused with background class and divides the probability to three classes: background, bear, and elephant. This causes to get low scores (probability) for bear at actual bear location. With these results we have seen that performance of Patch-based Training is very good for classes with distinctive patterns (leopard & zebra). For elephant class, success slightly decrease. Finally for bear class where the body can be both mixed with elephants and background, the localization performance becomes lower than Faster R-CNN.

### 4.2.3. Combined Model Testing on Four Classes

In this section, we investigate the performance of a new model. The new model is a combination of Patch Based and Faster R-CNN models. Each model was trained on four classes, which are bear, elephant, leopard, and zebra, independently. Patch Based Model is used as reference classifier. Because it has a better localization sensitivity but with small size of predicted boxes. For a given input, object detection is made by both models independently. Probability of Faster R-CNN detected boxes are re-evaluated using with Eq.4.5. Contribution is calculated with respect to Eq.4.4 between Patch-based Model and

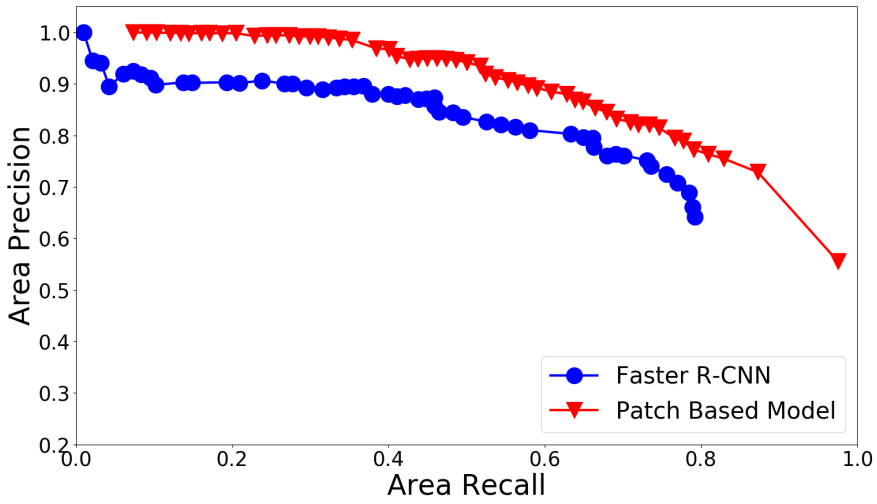



Figure 4.8. Area-Precision & Area-Recall Curve on Leopard Images.

Faster R-CNN boxes, where $n$ is the number of boxes obtained with Patch-based training model.

$$PatchCont_j = \frac{\sum_{i=1}^{n} Area(PatchBasedModelBox_i \cap FasterR-CNN_j)}{\sum_{i=1}^{n} Area(PatchBasedModelBox_i)} \tag{4.4}$$

$$P_{FasterR-CNN_j} = \frac{P_{FasterR-CNN_j} + PatchCont_j}{2} \tag{4.5}$$

In Eq.4.5, $P_{FasterR-CNN}$ is a list of Faster R-CNN predicted box scores, where $j = 1, ..., |P_{FasterR-CNN}|$. $PatchCont_j$ is a contribution factor to Faster R-CNN predicted boxes. Tests are applied three different confidence threshold scores of Patch-based model, where scores are $0.25$, $0.50$, and $0.75$ when full range of the heat map is normalized to $[0, 1]$. Area-precision & area-recall results of Faster R-CNN and Combined Models are shown at Fig.4.14, Fig.4.15, Fig.4.16, and Fig.4.17 for bear, elephant, leopard and zebra classes respectively.

According to area-precision & area-recall results (Fig.4.14, Fig.4.15, Fig.4.16, and Fig.4.17), Combined model has shown better performance than naive Faster R-CNN model for all classes, and it has shown less performance than Patch-based Model for leopard and zebra classes. A conclusion here is that, Patch-based model alone was not performing well on bear images, but combined model performed well.

By observing area precision & area recall results, we also wanted to check performance of Combined and Faster R-CNN Model using with classic precision-recall metric

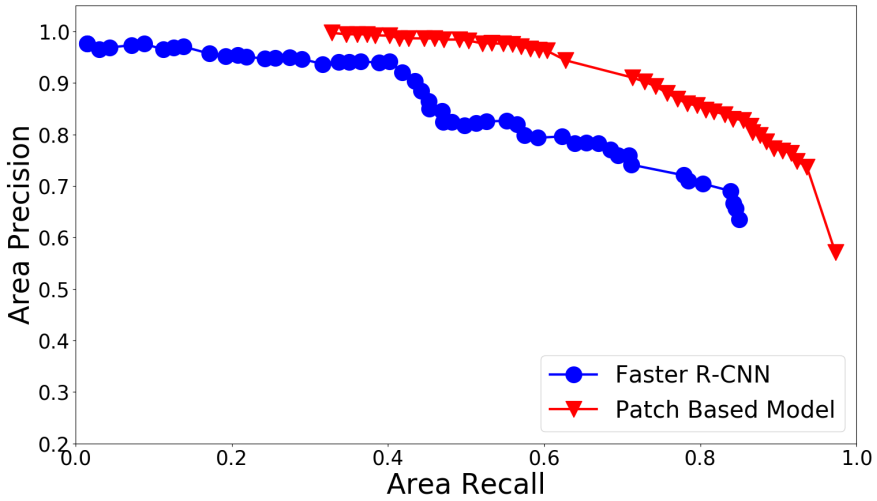



Figure 4.9. Area-Precision & Area-Recall Curve on Zebra Images.

(Eq.4.1). Tests are applied three different threshold scores of Patch-based Model where threshold scores are $0.25$, $0.50$, and $0.75$. Precision-recall results on test set are shown at Fig.4.18, Fig.4.19, Fig.4.20, and Fig.4.21 respectively. For all classes, positive effect of Combined Model has been observed. Combined Model ($@0.25$) outperforms than Faster R-CNN for all classes.

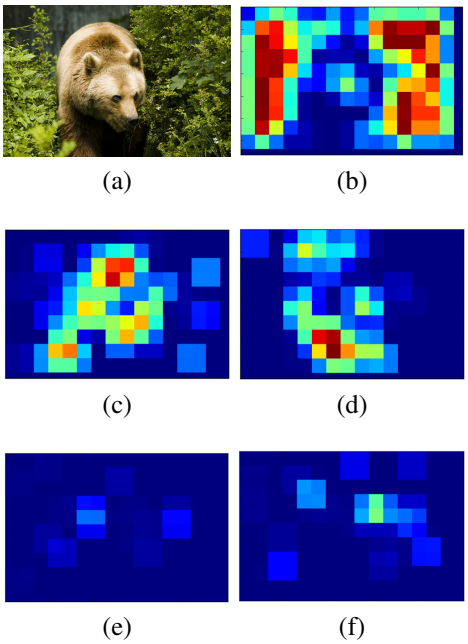


Figure 4.10. (a) Input Image, (b) Heat Map of Background, (c) Heat Map of Bear Class, (d) Heat Map of Elephant Class, (e) Heat Map of Leopard Class, (f) Heat Map of Zebra Class

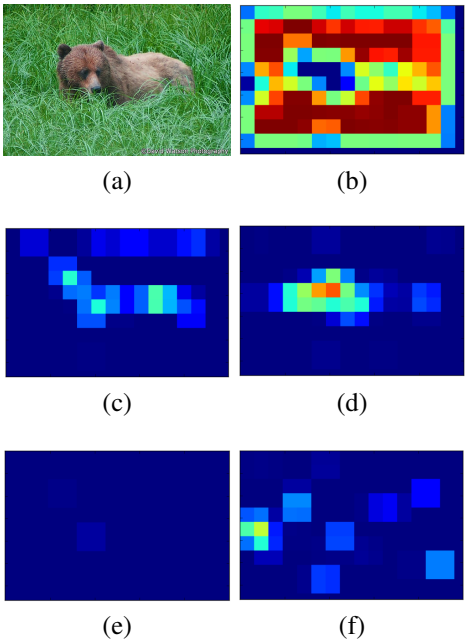
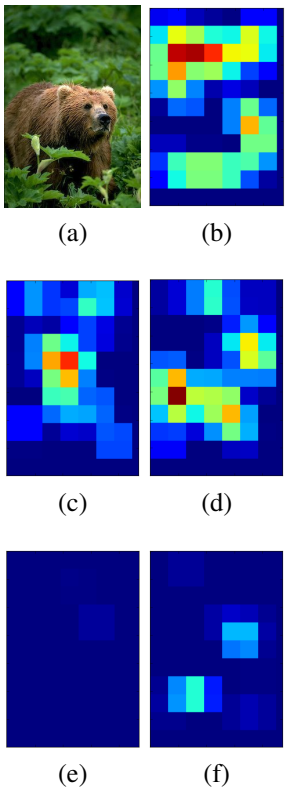


Figure 4.11. (a) Input Image, (b) Heat Map of Background, (c) Heat Map of Bear Class, (d) Heat Map of Elephant Class, (e) Heat Map of Leopard Class, (f) Heat Map of Zebra Class

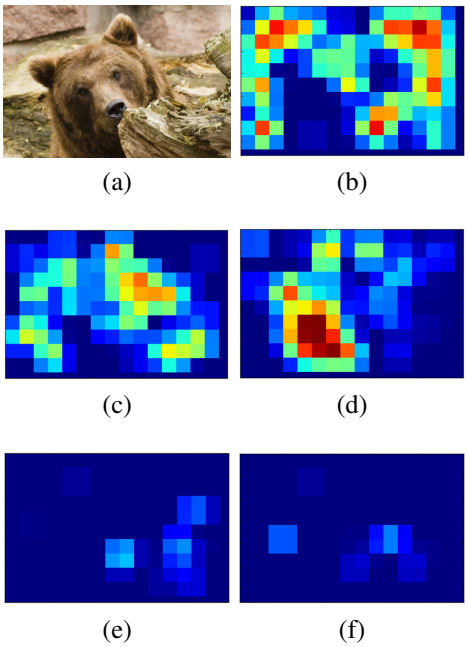


(a) (b)

(c) (d)

(e) (f)

Figure 4.12. (a) Input Image, (b) Heat Map of Background, (c) Heat Map of Bear Class, (d) Heat Map of Elephant Class, (e) Heat Map of Leopard Class, (f) Heat Map of Zebra Class



(a) (b)

(c) (d)

(e) (f)

Figure 4.13. (a) Input Image, (b) Heat Map of Background, (c) Heat Map of Bear Class, (d) Heat Map of Elephant Class, (e) Heat Map of Leopard Class, (f) Heat Map of Zebra Class

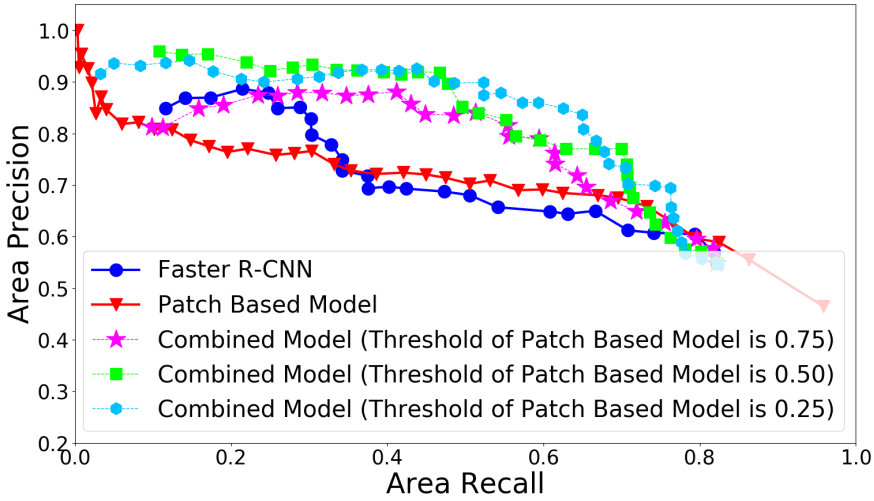



Figure 4.14. Combined Model and Faster R-CNN Area-Precision & Area-Recall Result on Bear Images.

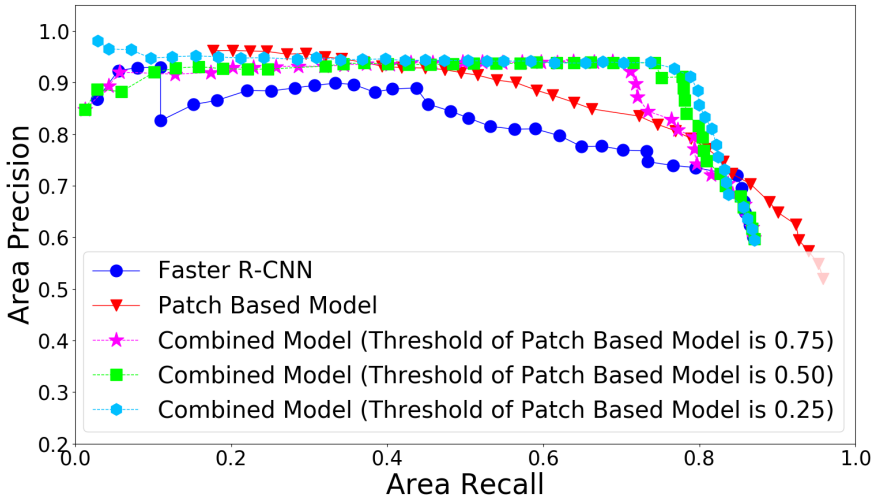



Figure 4.15. Combined Model and Faster R-CNN Area-Precision & Area-Recall Result on Elephant Images.

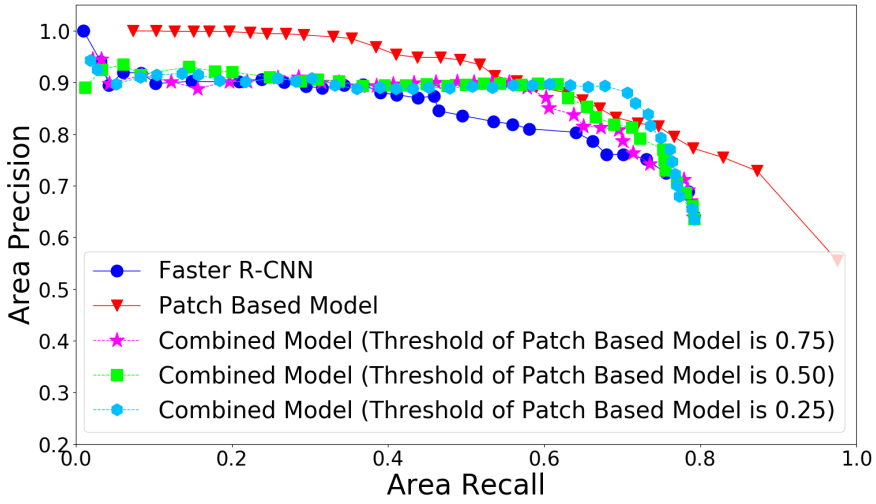



Figure 4.16. Combined Model and Faster R-CNN Area-Precision & Area-Recall Result on Leopard Images.

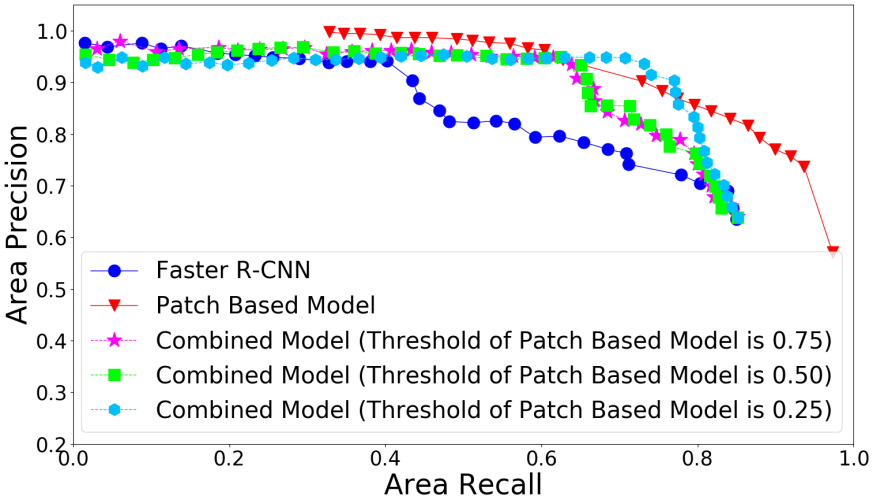



Figure 4.17. Combined Model and Faster R-CNN Area-Precision & Area-Recall Result on Zebra Images.

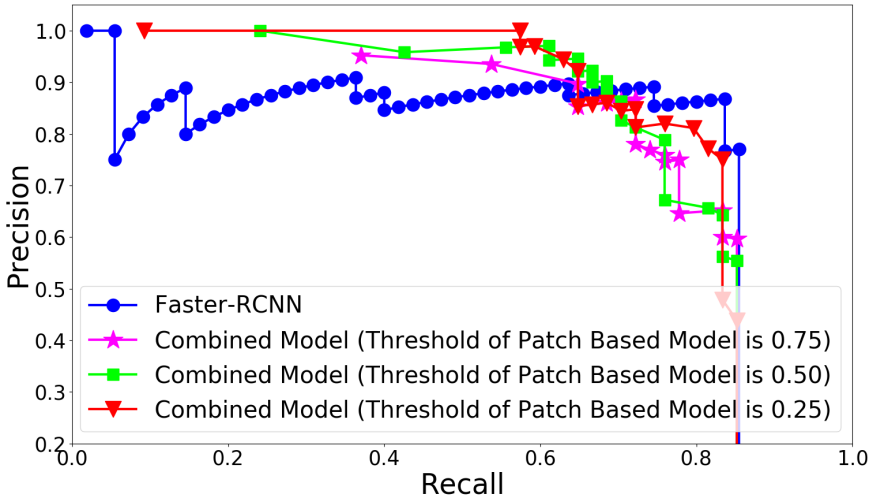



Figure 4.18. Combined Model and Faster R-CNN Precision & Recall Result on Bear Images.

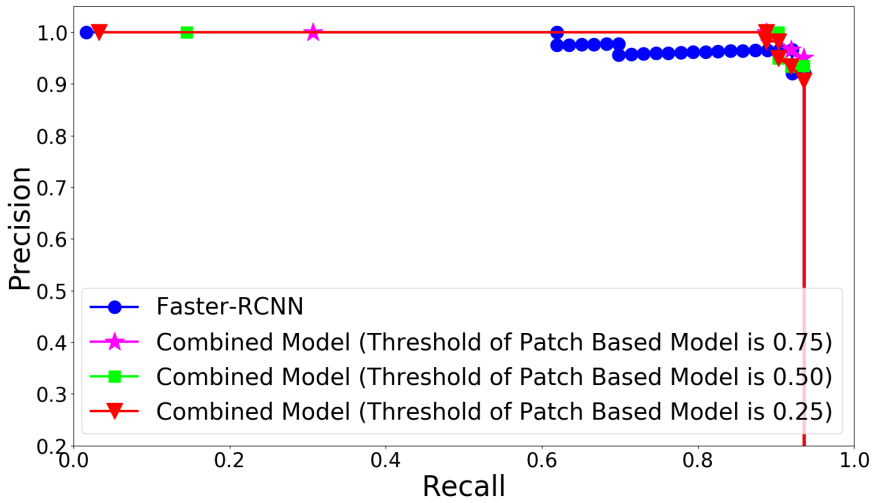



Figure 4.19. Combined Model and Faster R-CNN Precision & Recall Result on Elephant Images.

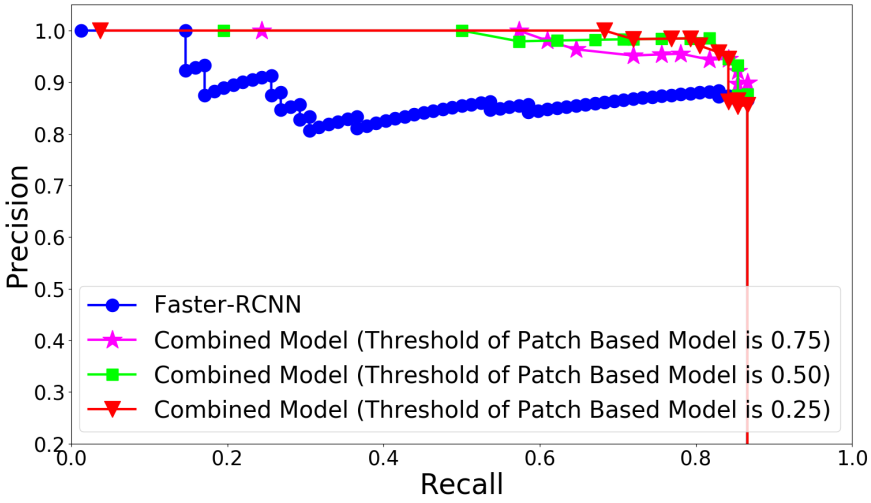



Figure 4.20. Combined Model and Faster R-CNN Precision & Recall Result on Leopard Images.

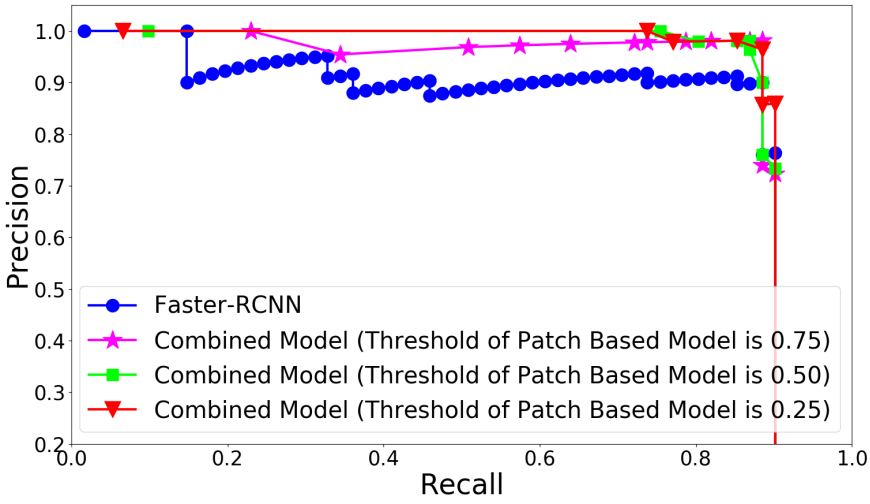



Figure 4.21. Combined Model and Faster R-CNN Precision & Recall Result on Zebra Images.

# CHAPTER 5

# CONCLUSIONS

In this work, we propose a Patch-based Training approach for CNNs to localize some animal species whose bodies have distinctive pattern, such as speckles of leopards, black-white lines of zebras etc. CNN is trained with patches cropped from image locations where animals exist. At test time, all image locations are visited in a sliding window fashion, and a heat-map is generated using the classification scores of the patches. After a series of morphological operations, heat-maps are converted to bounding box estimates.

We compared Patch-based Training localization performance using two different metric. First one, and well know, is Precision & Recall metric (Eq.4.1), and second one is Area-Precision &Area-Recall metric (Eq.4.3). Precision & Recall metric works like a binary classifier: these is true-positive or false-positive prediction. This metric is not very appropriate to represent the potential of Patch-based Model. Because at high threshold scores, Patch-based Model get false-positive prediction due to making prediction with small boxes even if it makes a high precise localization. To overcome this problem, we decided to use Area-Precision & Area-Recall metric. Area precision-recall metric works like an analogue classifier. There is no true-positive or false-positive prediction.

We tested object localization performance of Patch-based Model and Faster R-CNN on four animal classes, which are bear, elephant, leopard, and zebra classes. We observed that, on leopard and zebra classes Patch-based Model worked well, on elephant class it's performance is above Faster R-CNN, and for bear class it is not good. This is due to bear and elephant do not have a distinctive body pattern as leopard and zebra do. However, as shown by experiments Patch-based Model is able to locate the object more precisely when the detected class is correct and when used in combination with Faster R-CNN it increases the localization performance of Faster R-CNN. This is even true for bear and elephant classes.

Another advantage of Patch-based approach is that significantly less number of images are adequate for training. For instance, 50 zebra images instead of 450 images. This may become a reason of choice if the available dataset has a limited size.

Currently, we crop patches from test images in a sliding window fashion which

requires a great amount of time to extract features for all patches (80 seconds). Since cropped patches overlap each other, running time can be decreased by applying convolutions only on the non-overlapping parts of the image patches.

For future work, Patch-based Method also can be used as a box regressors for more challenger dataset. Because it makes a high precise localization, however with small boxes.

# REFERENCES


[1] O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[2] Convolution Image. `http://cs231n.github.io/convolutional-networks`.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[5] Faster R-CNN. `https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html`.

[6] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] P. Gong, D. J. Marceau, and P. J. Howarth. A comparison of spatial feature extraction algorithms for land-use classification with spot hrv data. *Remote sensing of environment*, 40(2):137–151, 1992.

[9] R. M. Haralick, K. Shanmugam, et al. Textural features for image classification. *IEEE*

*Transactions on systems, man, and cybernetics*, pages 610–621, 1973.

[10] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Y. Le Cun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[15] Leaky ReLU. `http://www.jefkine.com/deep/2016/08/08/initialization-of-deep-networks-case-of-rectifiers/`.

[16] Y. LeCun, C. Cortes, and C. J. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[17] V. Lo Brano, G. Ciulla, and M. Di Falco. Artificial neural networks to predict the power output of a pv panel. *International Journal of Photoenergy*, 2014, 2014.

[18] Max Pooling Image. `https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html`.

[19] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[20] M. L. Minsky and S. Papert. *Perceptions: An Introduction to Computational Geomry*. MIT press, 1969.

[21] Multi Layer Perceptron Image. `https://www.safaribooksonline.com/library/view/getting-started-with/9781786468574/ch04s04.html`.

[22] Neural Network Image. `https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/2`.

[23] M. Nielsen. Neural network and deep learning,why are deep neural networks hard to train? http://neuralnetworksanddeeplearning.com/chap5.html.

[24] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[25] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[26] S. Orhan and Y. Bastanlar. Detecting photos with leopards using convolutional neural networks. In *International Conference on Computer Science (UBMK), 2016 1th*, pages 1–4. UBMK, 2016.

[27] S. Orhan and Y. Bastanlar. Effect of patch based training on object localization with convolutional neural networks. In *Signal Processing and Communications Applications Conference (SIU), 2017 25th*, pages 1–4. IEEE, 2017.

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-

time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[29] ReLU Image. https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks.

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[31] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[32] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[33] 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)*, 115(3):211–252, 2015.

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

[35] Sigmoid Function Image. https://en.wikipedia.org/wiki/Sigmoid_function.

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

[37] Step Function . https://www.quora.com/What-is-the-sigmoid-process.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke,

and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[39] Tanh Image. `https://www.quora.com/When-would-one-use-a/-tanh-transfer-function-in-the-hidden-layer-/of-a-neural-network`.

[40] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[41] P. J. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Doctoral Dissertation, Applied Mathematics, Harvard University, MA*, 1974.

[42] C. Wolf and J.-M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR)*, 8(4):280–296, 2006.

[43] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.