5-2017

Object Recognition in Videos Utilizing Hierarchical and Temporal Objectness with Deep Neural Networks

Liang Peng
Utah State University

OBJECT RECOGNITION IN VIDEOS UTILIZING HIERARCHICAL AND TEMPORAL OBJECTNESS WITH DEEP NEURAL NETWORKS

by

Liang Peng

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

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Xiaojun Qi, Ph.D. Major Professor	Haitao Wang, Ph.D. Committee Member
Vicki Allan, Ph.D. Committee Member	Stephen Clyde, Ph.D. Committee Member
Adele Cutler, Ph.D. Committee Member	Mark R. McLellan, Ph.D. Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2017

ABSTRACT

Object Recognition in Videos Utilizing Hierarchical and Temporal Objectness with Deep Neural Networks

by

Liang Peng, Doctor of Philosophy

Utah State University, 2017

Major Professor: Xiaojun Qi, Ph.D.
Department: Computer Science

This dissertation develops a novel system for object recognition in videos. The input of the system is a set of unconstrained videos containing a known set of objects. The output is the locations and categories for each object in each frame across all videos. Initially, a shot boundary detection algorithm is applied to the videos to divide them into multiple sequences separated by the identified shot boundaries. Since each of these sequences still contains moderate content variations, we further use a cost optimization-based key frame extraction method to select key frames in each sequence and use these key frames to divide the videos into shorter sub-sequences with little content variations. Next, we learn object proposals on the first frame of each sub-sequence. Building upon the state-of-the-art object detection algorithms, we develop a tree-based hierarchical model to improve the object detection. Using the learned object proposals as the initial object positions in the first frame of each sub-sequence, we apply the SPOT tracker to track the object proposals and re-rank them using the proposed temporal objectness to obtain object proposals tubes by removing unlikely objects. Finally, we employ the deep Convolution Neural Network (CNN) to perform classification on these tubes. Experiments show that the proposed system significantly improves the object detection rate of the learned proposals when comparing with some state-of-the-art object detectors. Due to the improvement in object detection, the proposed system also achieves higher mean average precision at the stage of proposal classification than state-of-the-art methods.

(111 pages)

PUBLIC ABSTRACT

Object Recognition in Videos Utilizing Hierarchical and Temporal Objectness with Deep Neural Networks

Liang Peng

As the growth of mobile devices and social networks has been faster than ever, online image and video content have become truly ubiquitous today. Understanding of these images and videos, called vision, is one of the most primary ways for the human being to perceive the world. Computer vision, which refers to the study of enabling machines to see and understand the visual world, is fundamental in advancing Artificial Intelligence.

Object recognition, which is defined as the task of locating and recognizing object categories in images and videos, is a major research field in computer vision. Recent research in object recognition has achieved some significant improvement utilizing larger labeled data (e.g., ImageNet) and deep architecture of neural network algorithms (e.g., Convolution Neutral Network, Restricted Boltzmann Machine, etc.). However, object recognition research using deep architectures has been mainly focused on images. Little has been done in videos, one of the fastest growing types of multimedia content. Video understanding, especially large-scale object detection in video, has applications in brand awareness, autonomous cars, augmented reality, etc.

The research presented in this dissertation proposes and demonstrates a novel system that automatically recognizes objects in videos by incorporating tracking, object detection and classification using deep neural networks. By utilizing temporal and spatial information, the proposed approach achieved the better object recognition performance than the prior state-of-the-art methods in terms of the average precision.

This work is dedicated to my parents Fuping Wen, Bihe Peng, and my fiancee Zheng Han.

ACKNOWLEDGMENTS

I would like to express my deep appreciation for my major professor, Dr. Xiaojun Qi, for her tremendous amount of effort on guiding me through my Ph.D. research. I got interested in image processing and computer vision by taking the very first course of the Ph.D. program: image processing and computer vision by Dr. Qi. During these five years, she spent a lot of effort on discussing research ideas with me, designing the experimental settings, and polishing the writing for each paper. She drove to school under big snow in Christmas break just to pass me a co-authored paper with her hand-written edits.

I would like to thank my committee members, Dr. Haitao Wang, Dr. Vicki Allan, Dr. Stephen Clyde, and Dr. Adele Cutler for their valuable feedback on my proposal to further improve my research and dissertation.

Last but not least, I want to thank my parents and fiancee for their support and encouragement during my five years of Ph.D. studies. Sometimes I got a little disappointed and depressed when struggling with the unexpected experimental results. My long-distance fiancee often encouraged me and told me to be patient with the research on the phone. My parents always reminded me of doing exercise and taking breaks to keep good health during the busy work. Also, thank all my friends and colleagues for their support.

I could not have done it without all of you.

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Objects are ubiquitous all over the world and we see them everywhere in our daily life. Humans can quickly recognize any object from billions of different objects within a millisecond using their vision systems. After evolving three thousand decades, human vision is one of the most complex visual systems among all animals. The process of recognizing objects by human vision could be considered as extracting high-level concepts by looking at the objects followed by processing them in the brain. The objects could be in the 3-D form as seen in the physical word, or could be in the 2-D form such as objects in images or videos. More specifically, human eyes take in the physical stimuli of light rays and convert them into electrical and chemical signals in the brain. Then a complex process going through the central nervous system in the brain finally constructs and interprets the images and extracts high-level concepts of objects from the images.

1.1 Object Recognition: Definition and Applications

Computer vision, a research field in computer science, aims at using computers to serve similar functions of human vision. It includes a wide range of subtopics such as object detection, object recognition [1–8], image classification [9–16], image retrieval [17–23], and many others [24–26]. Among these subtopics, object recognition, as the task of locating and recognizing object categories in images and videos, is arguably one of the most important and challenging tasks in computer vision.

Object recognition could be defined as follows: given a digital image or a digital video, which is essentially a sequence of digital images, containing the unknown object(s), the computer outputs the position(s) and category label(s) of the object(s) in the image or the video.

Computer vision is a major component of Artificial Intelligence (AI) [27]. Object recognition serves as an essential part in computer vision and there is a wide range

Figure 1.1. Examples of object recognition in images

of existing and potential applications driven by object recognition. For instance, for Augmented Reality (AR) [28], if we want to develop some hardware devices (e.g., wearable glasses) [29] with the capability of generating the assistive information (e.g., object names, merchandise names, etc.) which can be integrated with the reality, we need to recognize these objects first. Robotics [30–32] also requires object recognition. For any type of robots that want to perform some tasks (e.g., walking, running, and picking up items) by interacting with the environments, perceiving the objects in their surroundings is a precondition. Another application is autonomous driving [33], [34]. Even though the sensors on the car could detect the physical objects around them, it is difficult for them to distinguish the objects of similar distances, sizes, and shapes. Hence, it is necessary to utilize computer vision-based object recognition techniques to understand the objects via the video signal captured by the cameras, which are installed and spinning on the top of the self-driving car. Object recognition also plays an important role in automating the manufacturing process [35], [36]. Recognizing the mechanical parts and defects of parts could make the manufacturing pipeline automated for some hardware industries. Therefore, object recognition plays a crucial role in developing a large number of applications and pushing the frontiers of AI.

1.2 Object Recognition Challenges

Object recognition by computers is a great challenge due to the following reasons: First, a digital color image is essentially a 3-D array, which contains three 2-D matrices corresponding to red, green, and blue channels and a gray-scale image is a 2-D matrix with each element representing the intensity value of the corresponding pixel in the image. Second, an image could contain one object, multiple similar objects, or no objects at all. Third, when an image contains one or more objects, each object could appear

in any size, shape, orientation, and lighting condition in the image. A small change in any of these factors would have great impact on the values of matrices by which images are represented. Accurately recognizing objects by computers requires capturing the patterns of objects with effective algorithms and features, which are invariant to changes in size, shape, orientation, and illumination for objects of the same category and are discriminative to objects from different categories. Fourth, the background of objects in images or videos could be anything in any complex visual appearance. This adds additional difficulty to extract objects from images or videos.

To address these challenges, machine learning techniques have been heavily applied in the research problem of detecting and recognizing objects in images and videos. Both object detection and object recognition could be treated as classification problems in the machine learning context. Due to the possibility that an object may appear in any size anywhere in an image, there are two questions we need to answer: First, where are the objects? Second, which category does an object belong to? For the first question, if we know a certain category of objects we are looking for, this problem refers to an object detection problem. More specifically, in object detection, the input is a model of an object (e.g., dog), and images possibly containing the object of interest. The output is the position(s) containing that type of the target objects if they are present in the image. The position is typically represented by an estimated bounding box to cover the target objects with the least amount of background. For the second question, the answer relies on object recognition. Actually, for object recognition, we need to answer both aforementioned questions assuming that we are looking for multiple categories of objects. The input is an image containing object(s) of unknown categories and a set of object categorical labels each object possibly belongs to. The output is the position(s) and object category (or categories) of object(s) in the unknown (new) image.

To find out the answers to both questions, a large number of rectangular boxes of varying sizes and aspect ratios can be used to scan over the image and classify each box to contain a certain object category or non-object. For object detection, since we know the type of objects we are looking for (e.g., dog), a binary classifier is usually employed to assign the class label (e.g., dog or non-dog) to each box. Each box is classified to a class either containing an object (e.g., dog) or not containing an object. For object

recognition, a multi-class classifier is usually employed to assign the appropriate class label to each box. For example, having known that an image possibly contains objects from the following categories: people, bird, horse, dog and bike, we need to not only find out which boxes may contain objects, but also find the categorical label from the five categories for each box.

Two stages, namely, training and classification, are typically involved in machine learning based classification problems. In the training stage, a set of data points with known labels are used to generate a model; In the classification stage, the learned model is applied to a set of unseen data points to classify each data point to a certain class label. Depending on the specific classification tasks, certain features are extracted from each data as a representation of data, which are used as the inputs to both learning and classification. As a result, the performance of a machine learning classification task typically depends on two factors: feature representation and classifier. Feature representations should be discriminative to different object classes and classifiers should be powerful to generate distinct models for different object classes. In the past few decades, researchers have been developing various features [37–39] and classifiers to improve machine learning performance for different tasks in various domains.

Considering object detection and object recognition in the context of machine learning-based classification, each bounding box among all scanned boxes could be considered as a data point. For object detection, in the training stage, we collect a set of images with labeled objects, which are enclosed by bounding boxes. We select a set of bounding boxes that contain the objects as positive examples and a set of bounding boxes that do not contain the objects as negative examples. Distinguishable features are then extracted from the bounding box to represent each example, and appropriate classifiers are employed on the extracted features to generate a model. In the classification stage, the generated model is applied on the same feature representation for each unknown bounding box to assign its class label. For object recognition, we still treat it as a classification problem. Various specific learning strategies have been developed to recognize objects, which will be discussed in the following sub-section.

1.3 Related Work

A lot of research work is carried out in object detection and recognition, and achieves the significant progress in recent two decades. Object detection and recognition methods can be roughly classified into two categories: multi-class approaches and class-agnostic approaches. Multi-class approaches treat the problem as a binary classification problem for each object category versus the non-object category on a huge number of windows. Several representative techniques in this category include: sliding window-based exhaustive search methods, boosting-based cascade methods, segmentation-based methods, and saliency-based methods. Class-agnostic approaches generate proposals, which are likely to contain objects of interests. A (n + 1)-way classification with n object categories and one background category is then followed to classify each object.

Sliding window-based exhaustive search methods aim to locate each category of objects by using template windows. Since the presence of the objects could be of any size and position in an image, a lot of candidate bounding boxes of different sizes at different positions need to be examined. One naive way is to use a large number of sub-windows of different sizes at different positions to scan over the image. Regarding the size of the sub-window, the smallest size could be the smallest box that the objects could appear in an image (e.g., 5x5). The size of the sub-windows in either width or height can be gradually increased by one pixel at a time until one dimension reaches the width or height of the image or the size of the bounding box equals to the size of the image itself. Next, on each sub-window, binary classification is performed to see if the bounding box contains an object (i.e., positive) or not (i.e., negative). Since this type of methods only address one type of objects, and the human face is a type of objects with great of interest, early object detection work mostly focused on face detection [40]. Sung and Poggio [41] proposes a face detection method based on examples of face images and non-face images. It collected a set of face templates and non-face templates as the training set, uses a "Mahalanobis-like" metric as the feature representation for each image, and employs a Multi-Layer Perceptron (MLP) net as the classifier to decide if each scanned window is a face or non-face. It can detect faces of different scales under various illuminations. However, it can only detect frontal faces with high computational cost. Rowley et. al. [42] introduce a similar face detector by a "bootstrap" technique. It

avoids the difficulty of manually selecting diverse non-faces samples by selectively adding the false detection result to the training set as the training progresses. Comparing with [41], it reduces the false detection rate. Souheil Ben-Yacoub [43] propose another face detection method with faster speed. Building upon the MLP classifier of sliding windows, it performs the convolution in the frequency domain to increase detection speed by 8 to 14 times with the same detection rate. There are many other sliding window-based approaches [44–46]. The advantages of these sliding window-based approaches are two-fold: First, they do not miss any searching location due to the exhaustive search. Second, they decompose the difficult problem (detecting multiple categories of objects) into multiple simpler problems. However, the search space is huge even for detecting one category of objects and most sub-windows do not contain objects. Furthermore, the computational complexity grows linearly as the number of categories increases and it becomes impractical when detecting objects from a large number of objects.

Boosting-based cascade methods aim to boost the speed of original sliding window-based methods by improving the image representation, features evaluation, and classifiers. The most representative approach under this category is Viola Jones's face detection method [47], which offers three major novelties. First, it introduced the integral image representation, which enabled the fast computation for features. An integral image is pre-generated so the sum of pixel values within any window can be quickly computed to extract features. Second, it uses a set of weak classifiers with each weak classifier only dependent on a single feature. Third, it combines the set of weak classifiers in a cascade structure to form a strong classifier so the windows that are unlikely to contain objects are eliminated very early and most computation focuses on object-like regions. Boosting-based cascade methods significantly increase the detection speed and make real-time detection for some categories possible (e.g., face and car). However, similar to the sliding-window-based approaches, the number of windows that need to be checked at the beginning is still very large and this number is much larger than the number of pixels in the images.

Unlike sliding window-based approaches, segmentation-based methods and saliency-based methods [48–50] aim to isolate objects-like regions quickly from pixel levels, followed by other techniques such as codebook, contour, or part-based model [51–53].

Segmentation refers to the task of partitioning an image into multiple regions with each having some unique characteristics. More specifically, segmentation is considered as the process of assigning pixels into different regions so that the pixels within the same region share the same characteristic. Consider a simple image which has two regions, one for foreground (e.g., an object), and the other for background. Top-down segmentation algorithms compute the conditional probability that each pixel belongs to the figure (i.e., foreground) given a hypothesis corresponding to the object category. Similarly, the conditional probability for each pixel to belong to the ground (i.e., background) is also computed. As a result, a probability map is generated and the segmentation result is then obtained based on the likelihood ratio between figure and ground. Once the segmentation is done, the detection focuses around the figure region. The codebook approach has been proposed to learn object proposals. It first generates some interesting regions (e.g., key points) from the figures in the training images and extract some features from these interesting regions. A clustering algorithm is then applied to group the interesting regions into n clusters. This clustering result is called a codebook (i.e., dictionary). For a new image, segmentation is first applied to find the figure. The same interesting region generation procedure is applied to the figure and each of these regions is assigned to one cluster in the codebook based on the Euclidean distance. In this way, the distribution of the newly given region is computed based on the codebook. This distribution is treated as the feature representation of the region. Finally, each region is classified as a certain category based on the labeled regions from the training set. This completes the object detection process. The contour-based method detects edges from the segmentation results and then generates contours of the objects. It builds the object category model using the contour-based geometric shape of objects to perform object detection. The advantage of segmentation-based methods is that it can quickly eliminate a large portion of images in the search space from the pixels level, so rest of techniques can focus on the object-like regions. Saliency detection serves as a preprocessing step for segmentation. Once the saliency map is generated, segmentation can be done by thresholding the map.

Selective Search (SS) [54] is another segmentation-based method for object detection. It combines the advantages of exhaustive search and segmentation methods to

capture sliding windows at all scales and significantly reduce the number of bounding boxes. It achieves these properties by the following steps: First, it uses diversifying image properties such as color, texture, and contours to segment out some object regions. Second, it uses these regions as the initial regions and starts merging the regions using a bottom-up approach. This merging process picks the two regions that have the most similarity to merge and continues this merging strategy until there is only one merged region left. The bounding boxes in this iterative merging process are likely to contain the true objects. Another successful segmentation-based object detection method is called Deformable Parts Model (DPM) [55]. It describes the objects by a collection of parts and the connection between them. An energy function is defined to model the sum of these two parts. Specifically, the wellness of matches between each part of the testing image and each part of a standard template of each object is measured by Histogram of Gradient (HOG) feature. Then the relationship between these parts is formed by a classic spring model to measure how much stretch occurs for the spring to match the standard template object and the test object. The DPM scans parts on a pyramid to minimize the energy function to find the possible object regions.

All these aforementioned ones are methods of detecting one category of objects at a time. For tasks of detecting multiple categories of objects, the computational cost grows linearly with the number of object categories. To address this issue, The class-agnostic approaches [56], [57] have been introduced to generate proposals, which are likely to contain objects of interest. It starts with a localization stage to rank all possible sub-windows based on the likeliness of including an object regardless of object categories and select top k windows as object proposals. It then proceeds to the classification stage, where a $(n+1)$-way classification with n object classes and 1 background class is followed to classify each object proposal. Two types of learning algorithms, namely, training-free and training-required, are commonly used for localization. One of the best training-free methods for learning generic object proposals is EdgeBox [58], which achieves a fairly high detection rate in a quick time. It uses the number of contours that are entirely enclosed in a bounding box as the likeliness of the box containing an object. To this end, it uses an edge detector to detect edges from each image and computes the contours based on the proposed affinity from edges. The objectness measure is computed as

the affinity score within each bounding box scanned across the entire image. Many training-required algorithms are also employed for learning object proposals. In recent years, due to the development of the computing power (e.g., GPU) and rapid growth of the labeled data, artificial neural network algorithms [49, 59, 60] with deep architecture have reached state-of-the-art performance for object detection. Region Convolution Neural Networks (RCNNs) [61–63] use raw pixels from images to feed in as the input layer and gradually learn the meaningful features to differentiate the objects from non-objects. Forward feeding and backpropagation are used in training to learn weights and biases of the network. Unlike the engineered features in the traditional machine learning paradigm, features are learned in the training process are globally optimized to reach the high object detection rate. The downside is that the training process is slow due to the intensive computation from a large number of forward and backward iterations for learning weights and biases. Therefore, the class-agnostic generation of proposals is expensive if using deep neural networks to detect object proposals and classify them.

From what has been discussed above, generating object proposals remains a bottleneck in object recognition. Quickly and accurately generating object proposals has been of a topic of great interest. A new BING-feature-based objectiveness measure [64] has been proposed to quickly generate object proposals (at 300 frames per second) with a high detection rate (i.e., 90% with 50 proposals and 96.2% with 1000 proposals on the VOC 07 data set [65]). EdgeBox we mentioned above achieves the state-of-the-art object recall rate (e.g., 75% recall at Intersection over Union (IoU) of 0.7 using 1000 proposals on the VOC 07 data set [65]) with the speed of approximate 0.25 seconds per image.

Most of aforementioned work has been focusing on generating object proposals for images. Video consisting of a sequence of dense images (e.g., 25 frames per second) raises additional challenges for methods that fit for images, due to high complexity and lack of temporal consistency if detectors were directly applied on individual frames. Some recent studies use temporal information and image-based detectors on videos. Sharir and Tuytelaars [66] apply object proposals in each frame and link them over frames into spatiotemporal object hypothesis. The detection on individual frames is still costly. Oneata et al. [67] propose the supervoxel method using hierarchical clustering of

superpixels in a graph with spatial and temporal connections. Hua et al. [68] incorporate tracking and detection to improve the consistency and reduce the cost of detection on individual frames. Some motion-based methods [69], [70] for object detection in video can only address moving objects but not static objects in video. In addition, unlike constrained video, which has constant background, unconstrained videos refer to the type of video that contains both static and moving objects with frequent changes of background. Even though breakthrough has been achieved on object recognition in images along with the development of deep neural networks, object recognition for both static and moving objects in an unconstrained video remains a challenging task in computer vision.

1.4 Overview of the Proposed System

In this section, we provide a brief overview of the proposed system, emphasizing the rationale for developing such a system. In chapter 5, we will describe the system in detail by presenting the input, output, and method for each module, and showing preliminary results on YouTube-Objects database.

As mentioned earlier, quickly and accurately generating object proposals in images remains a bottleneck for object recognition in general. Even though breakthrough results have been achieved on images for object recognition, it is computationally impractical to recognize objects by directly employing image-based object recognition methods in individual images for videos since a video is composed of a dense sequence of images. To address these two limitations, we propose a novel object recognition approach in video, which takes advantages of temporal and spatial properties of video to accurately and quickly recognize objects.

The proposed framework to recognize objects in video offers the following contributions:

- We formulate a novel cost function, which ensures good local representation and good content variation coverage in a video, and apply dynamic programming on this cost function to select key frames from the video. Applying this key frame extraction method on each shot of a long unconstrained video, the long unconstrained video can be decomposed into multiple stable frame sequences with little

content variation in each frame sequence. The effective decomposition of videos made it possible to apply image-based object recognition approaches to videos with dramatic reduction of computational cost and little loss in accuracy.

- We develop a compact feature representation to map each object's visual appearance into a tree-based feature representation by taking advantages of internal hierarchical structures of objects. Using this representation, we formulate a domain-specific learning schema to bridge the gap between generation of object proposals and classification of proposals existing in the traditional frameworks to improve the quality of object proposals.

- We propose a model-free method that is able to dynamically measure the objectness of each object proposal by exploiting temporal consistency within each optical flow during tracking. This method does not require any training and can simultaneously track and learn object proposals while further improving the quality of object proposals.

- We integrate the three proposed modules into a single large framework to quickly learn more accurate object proposals and apply Convolution Neural Network (CNN) to classify object proposals to achieve higher accuracy of recognizing objects in videos. To the best of our knowledge, it is the first generic framework that is capable of recognizing both static and moving objects in every single frame in unconstrained videos.

As we mentioned in section 1.2, detecting and recognizing objects in individual frames of a video is impractical due to extremely high computational complexity. It is well known that an unconstrained video typically has large content variations across all frames in a relatively long time range; On the other hand, adjacent frames within a relatively short time range have small content variations. We call each short sequence of frames with small content variations as "a stable frame sequence". The rationale of the entire proposed system is as follows: First, we develop an approach to divide each video into many "stable frame sequences"; Second, we only generate object proposals on the first frame of each "stable frame sequence" and use these proposals as initial

possible positions of objects to initialize multiple object trackers to track through each "stable frame sequence". This step produces a lot of object proposal tracks, which are a sequence of object proposals extracted from a sequence of frames. Due to the small variations in each "stable frame sequence", we assume each object proposal track has the same categorical label. The total number of possible categories include n object categories defined in the problem domain and one background category (i.e., does not contain any object). In the last step, we need to classify these object tracks into one of the $n + 1$ categories to produce labeled videos as output.

Following this rationale, there are three major components in the proposed system as shown in Figure 1.2: Frame sequence generation, object proposals learning, and RCNN [49] classification.

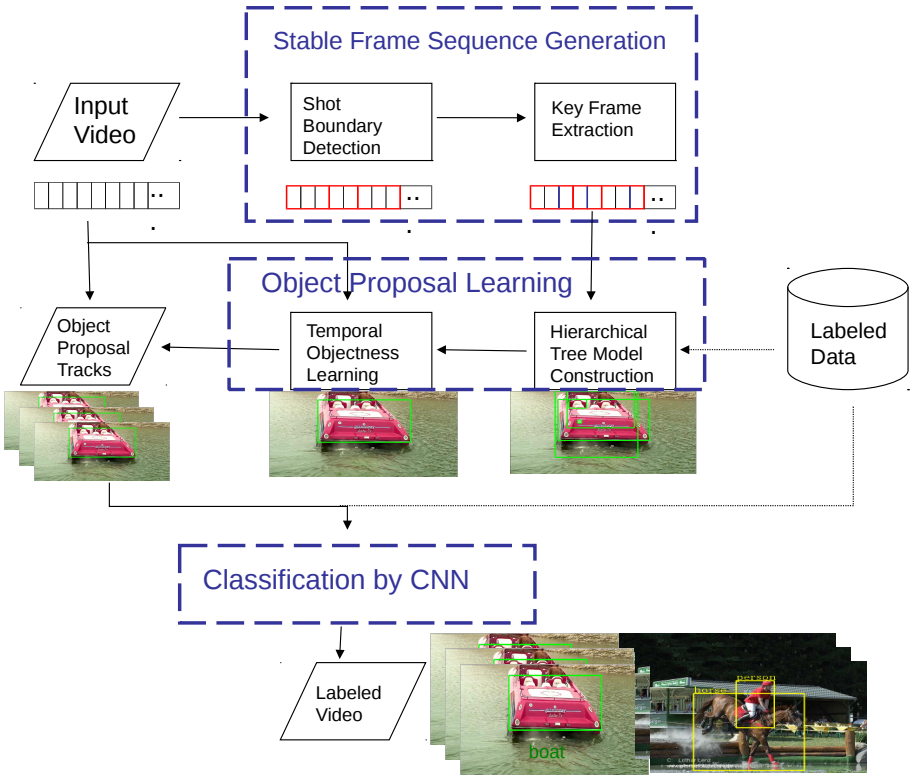

Figure 1.2. System overview of object recognition in video

In the frame sequences generation module, we apply a shot boundary detection algorithm [71] to divide the entire video into many shots and propose the key frames

extraction algorithm to extract key frames from each shot. In the object proposal learning module, we develop a hierarchical model to generate high quality object proposals on each of the key frames and use these object proposals to initialize trackers to track and refine object proposals based on the proposed temporal objectness measure. In the CNN classification module, we apply a two-stage (i.e., train and fine-tune) learning schema to train a model on a large labeled data set and fine tune the trained model using the domain-specific data set. We then classify each of the object proposal tracks into a specific object category or background to produce labeled object proposal tracks.

Extensive experiments have been conducted on benchmark datasets to evaluate each proposed module as well as the entire framework by comparing with the state-of-the-art methods. The proposed method achieved significant improvement over state-of-the-art methods in terms of the average precision of each individual category as well as the mean average precision overall.

The rest of the dissertation is organized as follows: In chapter 2, we describe frame sequence generation for video in detail. In chapter 3, we introduce the proposed hierarchical model and temporal objectness to learn object proposals in video. We also present the experimental result on each individual component in chapter 4, we illustrate the classification of object proposals using deep neural networks. In chapter 5, we describe the entire integrated framework and workflow, and present the corresponding experimental results. Finally in chapter 6, we present the conclusion and the future work.

CHAPTER 2

STABLE FRAME SEQUENCES GENERATION

As cameras and social networks became more and more popular in the recent years [72], a large proportion of the online multimedia content is in the form of videos [73]. There are two types of videos: constrained and unconstrained. Constrained videos refer to the type of videos that contain relatively fixed or stable background. The foreground objects that are moving while the background is constant. For example, the videos that are captured from the cameras inside the elevators or buses are constrained videos. By contrast, unconstrained videos refer to the type of videos with frequent changes of background. For example, if we hold a camera to shoot 360-degree scenes around our houses, the resultant video is unconstrained. Another example of unconstrained videos would be a movie that contains different scenes.

It is much more challenging to recognize objects in unconstrained videos than in constrained videos because the frequent changes in the background bring up a lot of noise in the temporal domain, which makes it difficult to segment out the foreground objects and track them.

Our proposed system aims to address the problems of object recognition in videos. Since an unconstrained video includes frequent changes of visual content, we propose a method to divide the entire video into many stable frame sequences such that each frame sequence has little content variations. Then we develop an object detection algorithm building upon the existing state-of-the-art object detectors to generate object proposals on the first frame for each of these sequences. Third, we use these detected object proposals to initialize multiple trackers to estimate the optical flow of each object proposal over each stable frame sequence to generate the object patch tubes. Last, we assign a label to each of these tubes by deep neural networks based classification.

In this chapter, we describe the proposed method of generating stable frame sequences. The proposed method consists of two modules: shot boundary detection and

key frame extraction. Shot boundary detection aims to divide the video into multiple shots by finding the boundary frames in the video. It should be noted that moderate content variations may still exist within each shot. As a result, we propose an algorithm to select key frames from each shot, where each key frame is significantly distinct from each other. In this way, each frame sequence between two selected key frames has little content variations. Hence, we can use these key frames to divide the videos into a set of stable frame sequences.

2.1 Shot Boundary Detection

Hard-cut and gradual changes are two common shot changes in a video. The hard-cut changes refer to the sharp changes between two adjacent frames with one belonging to the previous shot and the next one belonging to the next shot. The gradual changes mean a transition consisting of several frames, which belong to two shots. Common gradual shot changes include fade, wipe, and dissolve. Fig. 2.1 shows one example for gradual change and one example for hard cut change. The aeroplane images sequence in the top two rows shows an example of gradual change. More specifically, it is a dissolve. The bird image sequence in the bottom two rows shows an example of hard cut change, as we can see that the last image in row 3 suddenly changes to a quite different image, which is the first one in row 4.

We introduce two shot boundary detection methods in the following two sections. The first method is able to quickly detect hard cut shot boundaries with a pre-set threshold. The second method is able to detect both hard cut shot boundaries and gradual changes using an adaptive threshold. Depending on the types of the videos and shot changes, each method has its own suitable scenario. For the type of videos with only slow movements and only hard cut changes, it is better to use the method with pre-set threshold since it is very fast and there is no need to compute the distribution of frame differences. For the type of videos containing both gradual changes and hard cut with fast movements, adaptive thresholding is more suitable since the threshold is adjusted based on different types of movements. In Fig. 2.1, the aeroplane image sequence is divided by two shots by the fixed-threshold method (row 1 contains images in shot 1 and row 2 contains images in shot 2). the bird image sequence is divided by two shots

by the shot boundary detection method with the adaptive thresholding (row 3 contains the images in shot 1 and row 4 contains the images in shot 2). We describe how each method works in the following two sections.

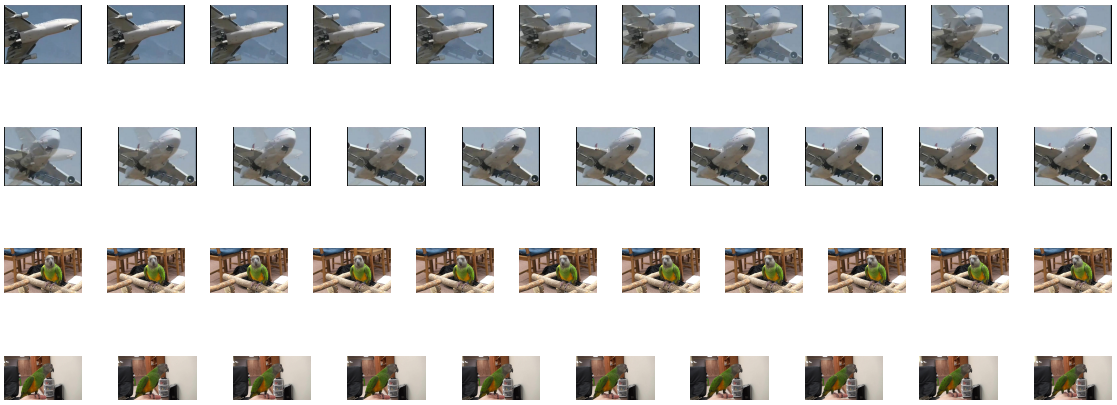

Figure 2.1. Sample images of gradual and hard cuts changes: images of gradual change (row 1 and row 2) from a aeroplane video and hard cuts change (row 3 and row 4) from a bird video

2.1.1 Hard Cut Detection Using the Fixed Threshold

For a given video, we apply a simple shot boundary detection algorithm to divide the video into multiple shots. For this module, the input is a given video, and the output is a set of shots, which are indicated by the starting and ending frame number in each shot. The first shot starts at the first frame, and the last frame in each shot is the previous frame of the next shot. A content-aware detector [74] is used to detect shot boundaries and it computes the difference between every pair of subsequent frames in a video by averaging the differences of every corresponding pixels between two adjacent frames in red, green, and blue channels. This differences is compared with a pre-set threshold. Two frames are separated into different shots if the difference is greater than or equal to the threshold. Two frames are grouped into the same shot if the difference is less than the threshold.

2.1.2 Hard Cut and Gradual Changes Detection Using the Adaptive Threshold

Content-aware detector uses the pre-set threshold to decide shot boundaries. Even though this method achieves the reasonable shot boundary detection results, it does not

work well for videos with both fast and slow movements. Since the portion in video with faster movement should have a larger threshold than the portion in video with slower movement, one fixed threshold cannot work well on both types of movements in a video. To this end, we develop a more sophisticated approach using an adaptive threshold to detect not only hard cut shot boundaries but also gradual changes regardless the type of movements in the video.

This sophisticated approach sets an adaptive threshold in a local temporal range of frames rather than a global range. Specifically, to detect the shot boundaries with hard cut changes, it computes the discontinuity of each frame by calculating the change between the current frame and its previous frame using their corresponding feature vectors. It then uses a moving window of size 32 to slide across the video. At each sliding location, if the middle frame located at the center of the window has the maximum discontinuity which is also significantly greater than the discontinuity of the other frames within the window, this middle frame is defined a shot boundary. To detect the shot boundaries with gradual changes, we develop a method using the adaptive threshold. To this end, we use a moving temporal window of size 32 to compute the difference of standard deviation between each pair of adjacent frames across the video. Second, we calculate the differences between minimum and maximum change of standard deviations within each window (e.g., the mean discontinuity of the other frames plus twice the standard deviation of the discontinuity values) . The middle frame in the window with a large difference, whose value is both the maximum in the current window and also greater than an adaptive threshold, is set as a shot boundary. By empirical studies, we found these differences follow exponential distribution after conducting the Kolmogorov-Smirnov test [75] for goodness of fit. Fig. 2.2 shows the histogram of one video (i.e., 0001.mp4) from the aeroplane category as an example. This histogram can be approximated by the exponential distribution. The Probability Density Function (PDF) for exponential distribution is $\lambda e^{-\lambda x}$ and its Cumulative Density Function (CDF) is $1 - e^{-\lambda x}$. We estimate the parameter λ by fitting the data. Then we set the threshold by letting $1 - e^{-\lambda x} = 95\%$ and solve for x so that the adaptive threshold is set to be a value, which leads to the value of CDF to be greater than 95% of all differences.

In terms of the computational complexity, suppose we compute a histogram for

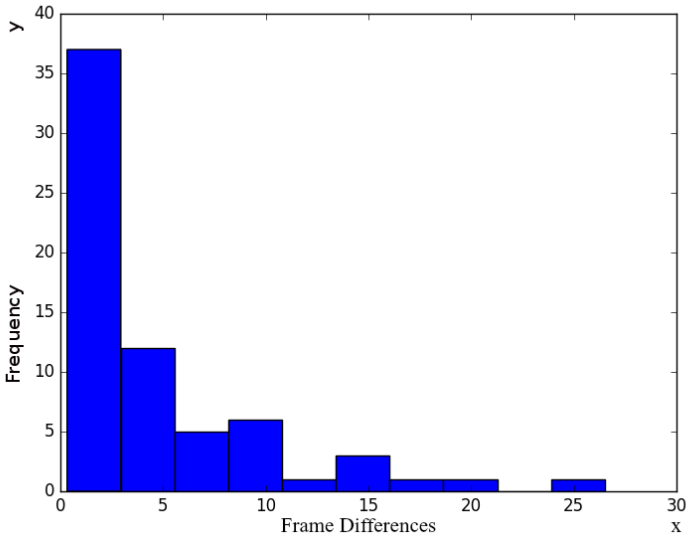

Figure 2.2. Distribution of adjacent frame differences for aeroplane: adjacent frame differences for aeroplane video 0001.mp4 (X: frame differences; Y: Frequency)

each frame (i.e., an image with width m and height n) as its feature vector. The time complexity for calculating the feature is $O(mn)$. Suppose there is total of k frames, the time computational complexity of calculating the discontinuity is $O(k)$.

2.2 Key Frame Extraction

After shot boundary detection, we propose a method [76], [77] to find key frames to divide each shot into a set of "stable frame sequences" with little content variation. Key frames refer to a set of frames selected from a video (could be either a shot or a long video) that can best summarize and represent the content of the entire video. We formulate the problem of key frame extraction as a cost optimization problem and provide the solution to this optimization.

2.2.1 Problem Formulation

The key frames should possess two properties: 1) Good local representation: Each selected key frame has good representativeness for its neighboring frames; and 2) Good content variation coverage: The selected frames cover large content variations from the whole shot or video. Based on these two properties, we formulate the key frame

extraction problem as a cost optimization problem. Let N represent the total number of frames in a shot or video to be summarized, and $\{F_i\}$ $(i = 1, ..., N)$ represent the set of frames from the entire shot or video. The goal is to select M frames $\{a_i\}(i = 1, ..., M)$ from $\{F_i\}$, which are the best representative frames to summarize the shot or video.

Good local representation means that each selected frame has large visual similarity with its neighboring frames (i.e., each selected frame is similar enough to represent its neighboring frames from the original shot or video). Let H_i stand for the 1-D feature vector of the ith frame, F_i. Let Sim represent a function that computes the similarity between two 1-D vectors. The local representation of the ith frame F_i is defined as:

$$P(i) = \begin{cases} Sim(H_i, H_{i+1}) & \text{if } i = 1 \\ \frac{Sim(H_{i-1}, H_i) + Sim(H_i, H_{i+1})}{2} & \text{if } 1 < i < N \\ Sim(H_{i-1}, H_i) & \text{if } i = N \end{cases} \tag{2.1}$$

That is, the local representation of F_i is computed as the average similarity between its previous frame F_{i-1} and its next frame F_{i+1} (except for the first and last frames whose local representation is computed by its similarity with the second frame and its similarity with the second to the last frame, respectively).

A good content variation coverage can be interpreted as that consecutively selected frames have a large dissimilarity. Here, we define the similarity of the two key frames containing the objects of interest by:

$$Q(k, j) = Sim(H_k, H_j), \tag{2.2}$$
$$1 \leq k \leq N,\ 1 \leq j \leq N,\ k \neq j,$$

where H_k and H_j are the 1-D feature vectors of two selected adjacent key frames from F_k and F_j, respectively.

A good summary of the video requires larger $\sum_{i=1}^{M} P(a_i)$ (i.e., better local representation), and smaller $\sum_{i=2}^{M} Q(a_{i-1}, a_i)$ (i.e., better content variation coverage). Then

we create a total cost function C:

$$\begin{aligned} & C(a_1, a_2, ..., a_M) \\ = & \sum_{i=1}^{M} \alpha[1 - P(a_i)] + \sum_{i=2}^{M} (1-\alpha) Q(a_{i-1}, a_i), \end{aligned} \tag{2.3}$$

where α is a weighting factor with its value in $[0, 1]$. This total cost function consists of cost for local representation based on a single frame and its neighboring frames and content variation coverage based on a pair of consecutive key frames.

2.2.2 Solution to Key Frame Extraction

The goal is to find the solution set $(a_1^*, \cdots, a_M^*)$ to minimize the cost function (2.3), which can be solved by dynamic programming [78]. Let $\Omega_i[a_i]$ denote the cost based on the optimal selection of the first i frames, i.e., $\Omega_i[a_i] = \text{Minimize } C(a_1, \cdots, a_i)$. For selecting the $1st$ key frame, the cost function only involves cost for local representation since the content variation coverage cost is zero. Hence, we have

$$\Omega_1[a_1] = \alpha(1 - P(a_1)) \tag{2.4}$$

as the base case. In addition, by definition of $\Omega_i[a_i]$, we have

$$\Omega_i[a_i] = \Omega_{i-1}[a_{i-1}] + \alpha(1 - P(a_i)) + (1-\alpha) Q(a_{i-1}, a_i) \tag{2.5}$$

which shows that the selection of the next key frame index a_i is independent of the selection of previous frames for a given cost function. Once Ω_i is computed, the optimal solution to the whole problem can be obtained by taking:

$$a_M^* = \underset{a_M}{\text{argmin}}\ \Omega_M[a_M] \tag{2.6}$$

and tracking back in order of decreasing i until the base case (2.4) is satisfied. In other words,

$$\begin{aligned} a_{i-1}^{*} &= \underset{a_{i-1}}{\operatorname{argmin}}\{\Omega_{i-1}\left[a_{i-1}\right] + \alpha(1 - P(a_i^{*})) \\ &\qquad +(1-\alpha)Q(a_{i-1}, a_i^{*})\} \end{aligned} \tag{2.7}$$

The computational complexity of this dynamic programming approach is $O(MN^2)$ where M is number of the key frames and N is the number of frames from the original shot or video.

After dividing the video into multiple shots and extracting key frames from each shot, we have formed a set of key frames by merging all key frames from each shot. This set could be considered as the key frames of the entire video. Then we use key frames to divide the entire video into multiple "stable frame sequences" by using the ith key frame as the first frame and using the frame before $i + 1th$ key frame as the last frame in the ith stable frame sequence. The number of resultant stable frame sequences in the video is equal to the total number of key frames.

CHAPTER 3

OBJECT PROPOSALS LEARNING

As we mentioned in chapter 1, object proposal generation remains a bottleneck for object recognition in video. This chapter focuses on presenting the proposed novel methods to improve the quality of object proposals. First, we propose a tree-based hierarchical model to capture the internal structural properties of each category of objects and develop a learning schema to improve object proposal generation. Second, taking advantages of object consistency in the optical flow, we formulate a temporal objectness measure to further reduce false positive object proposals during tracking.

To test the effectiveness of the proposed tree-based hierarchical model and the temporal objectness measure, we design separate experiments to test each module. Specifically, for evaluating the tree-based hierarchical model, we use EdgeBox [58] to generate object proposals and use the proposed model to re-rank them to see if higher recalls are achieved. For evaluating the temporal objectness, we first use EdgeBox to generate the top 50 proposals on each shot representative frame obtained by applying key frame extraction on each shot to compute the precision-recall curve. Then we apply the SPOT trackers to track each object for all the frames in each stable frame sequence, and re-rank the object proposal tracks based on the temporal objectness to compute the precision-recall curve for comparison. Furthermore, we evaluate the integrated system to see if the object recognition performance is improved by combining all proposed modules including shot boundary detection, key frame extraction, tree-based hierarchical model, and temporal objectness. It should be noted that the specific parameter settings for testing the integrated system are slightly different from those used to test each individual module based on empirical studies and running time. The details of the methods and experiment settings are described in each subsection.

3.1 Learning Object Proposals Using a Hierarchical Tree Model

3.1.1 Introduction

Existing methods for generating object proposals (e.g., BING [64] and EdgeBox [58]) typically use measures of likelihood of a bounding box containing objects to retrieve top n bounding boxes as candidate proposals. Due to the class-agnostic nature of these methods, all generated object proposals are generic objects. On the other hand, during the classification stage, the object categories are usually bounded in a certain domain, which is dependent on the labeled training data sets. For example, the VOC 07 data set [65] has 20 object categories. This difference means that the object proposals usually contain additional categories of objects that may not be in labeled categories for classification. For example, a "propeller" of an aeroplane might be generated by object proposal generation methods as an object, but the labeled categories for classification only contains a category such as "aeroplane" instead of "propeller". Similarly, a "bicycle wheel" might be detected as a proposal whereas the classification labels only contain "bicycle" not "bicycle wheel"; A "train window" might be detected as a proposal but the classification labels only contain "train" instead of "train window". The false positives caused by these gaps have significant negative impact on both detection recall and precision of object detection. Consequently, it will further affect object recognition accuracy in the object proposal classification stage. For example, in the classification stage, usually all object proposals from the generation stage, which are not in the labeled categories for classification are treated as the "background" category. However, part of these "background" objects are in some out-of-domain categories (i.e., some object categories are not in the set of object categories in the domain of problems being addressed) and are not truly "background". Hence, reducing false positives in object proposal generation stage will improve object detection precision, retrieve more candidate proposals to improve recall and further improve classification accuracy.

With the goals of reducing false positives and improving the recall rate for any domain-specific object detection task, we observe that each object category differs from others not only by its visual appearance as a whole part, but also by its intrinsic structure, such as its sub-parts and complexity of its structures. For example, a bicycle differs

from a train when being looked at as a whole, which corresponds to its visual appearance as a whole entity. In addition, a bicycle contains two "wheels" and one "frame" whereas a train contains "multiple rectangular windows". A bounding box of blue "sky" object may not contain any sub-parts as an object since it is relatively pure and it is true background. Hence, in additional to visual appearance, sub-part structures and the complexity of an object can be employed to construct richer objectness to better represent the objects.

Driven by this intuition, we develop a model which takes an object's visual appearance as a whole and its parts-based hierarchical structure into account. There are three major novelties for the proposed work. First, we develop a compact 1-D feature vector of length 23 as the visual appearance representation of an object; Second, we develop an algorithm to map each object to a hierarchical tree structure representation, which captures the relationships among sub-parts and the complexity of the object structure. Third, we formalize a learning schema to measure each proposal's objectness by comparing with visual features and tree structures using the nearest neighbor method.

3.1.2 The Proposed Hierarchical Tree Method

The proposed method builds upon the existing object proposal generation methods to improve the object detection precision and recall. As a high-level view, we first employ the existing object proposal generator on each image to obtain top n object proposals. Since many of these proposals include generic objects and backgrounds and only a small portion of them include the objects that belong to domain-specific categories in labeled training data, we develop a learning schema to compute a confidence (i.e., objectness) for each object and re-rank top n objects in each image, with the aim of ranking category-specific objects towards the top to improve the recall rate for different levels of n. In other words, we want to select a fewer number of proposals from the top n object proposals to achieve the same recall as obtained by n proposals of the state-of-the-art methods. The subsections below are organized as follows: Section 3.1.2.1 introduces the proposed compact feature for object's appearance as a whole. Section 3.1.2.2 describes the methodology to map each object to a hierarchical structure represented by a tree. Section 3.1.2.3 formalizes the proposed learning schema to compute confidence of each

object using the compact feature and the tree-based hierarchical model.

3.1.2.1 Compact Feature Representation

Even though approaches based on deep neural networks can learn feature representations with high discriminative power, it is quite costly in terms of time. For object proposal generation, we want to develop a fast learning schema to eventually rank object proposals from the category-specific dataset higher than object proposals of the other categories. Hence, the constructed compact feature should have high discriminative power to distinguish objects from non-objects and moderate discriminative power across different categories of objects.

We observe that object and non-object proposals differ a lot in terms of their properties. First, non-object proposals usually contain background and object proposals usually have one dominate object present within the bounding box. Second, background proposals are generally more evenly distributed and have less variations in terms of their pixel intensities. In contrast, object proposals are generally more skewness and have more intensity variations due to the few dominate colors of the main object and a small background region outside of the object boundary. Third, from the spatial point of view, background proposals are usually more evenly distributed among different color pixel intensities inside the bounding box while object proposals would have a few dominant colors occupying the majority region of the bounding box. Hence, the variation-related and spatial-related properties can be used to distinguish object proposals from non-object proposals.

To this end, we define the first category of features to measure the color variations by computing variances in red, green, and blue channels (three values) and entropy of the luminance-based histograms within the proposal (one value). We then define the second category of features to measure the spatial evenness of the color distribution. Specifically, we convert the proposals into a histogram in red, green, and blue color channels. We empirically choose the top five bins with the highest frequencies and compute their normalized frequencies. This would produce 15 features (five normalized frequencies for each of the three channels). Finally, we observe that the positions of the object proposals are not evenly distributed within an image. For example, the

probability for object proposals near the four corners or the edge of the image is far less than the probability for object proposals near the center. Some aspect ratios of the object proposals may occur much more frequently than other aspect ratios due to the limited shapes of objects in the entire set of object categories. Following this intuition, we incorporate the distribution of the relative locations and aspect ratios of proposals into extracted features to represent each proposal. Since the image dimension (i.e., the width and height) may vary among different images, we use the relative positions and dimensions of the proposals within an image to define four additional features as follows: the relative width (ratio of the width of the proposal to the width of the image), the relative height (ratio of the height of the proposal to the height of the image), the relative horizontal coordinate, and the relative vertical coordinate. The relative horizontal coordinate is the ratio of the horizontal coordinate of the upper-left corner of the proposal to the width of the image. The relative vertical coordinate is the vertical coordinate of the upper left corner of the proposal to the height of the image. These four features capture the properties regarding the relative aspect ratio and position of a proposal. In all, the compact feature of the length of 23 is extracted for a proposal and will be used to classify object and non-object proposals.

To clearly explain the proposed features, we provide the mathematical definition for some of these features. For a given image I with width W and height H and a proposal B in the image with x and y representing the horizontal and vertical coordinates of the upper-left corner of B, respectively, let w and h represent the width and height of B and $P_{i,j,c}$ represent the pixel value at the horizontal and vertical coordinates (i, j) of I in the channel c of the RGB color space, where $c \in \{R, G, B\}$.

For the first category of the proposed features, the variance of B in a red color channel could be expressed as:

$$\frac{\sum_{j=y}^{y+h}\sum_{i=x}^{x+w}(P_{i,j,c=R} - \frac{\sum_{j=y}^{y+h}\sum_{i=x}^{x+w}P_{i,j,c=R}}{w\times h})^2}{(w \times h - 1)} \tag{3.1}$$

The variance of B in green and blue channels can be computed similarly using the above formula by substituting $c = G$ and $c = B$, respectively.

To compute the entropy, we first compute the histogram based on the pixel counts

for each of 768 bins (i.e., 256 possible values in each color channel times the total number of channels), and then apply the standard entropy formula on relative frequencies of 768 bins.

For the second category of the proposed features, we use 64-bin histogram on each of the RGB channels to compute the five highest normalized frequencies. Firstly, we compute the histogram based on the pixel counts of 64 bins on each of RGB channels. Secondly, for each channel, we normalize the frequency of each bin by dividing the total pixel counts in the channel. Lastly, we rank relative frequencies and select top 5 relative frequencies in each channel to produce a total of 15 values.

For the third category of the proposed features, we compute Xr (the relative horizontal coordinate), Yr (the relative vertical coordinate), Wr (the relative width), Hr (the relative height) as follows:

$$Xr = \frac{x}{W} \tag{3.2}$$

$$Yr = \frac{y}{H} \tag{3.3}$$

$$Wr = \frac{w}{W} \tag{3.4}$$

$$Hr = \frac{h}{H} \tag{3.5}$$

3.1.2.2 Hierarchical Tree Structure

A total of n object proposal bounding boxes may be retrieved by the proposal generator for an image. Since each object or non-object bounding box may include smaller bounding boxes corresponding to sub-parts of an object, we convert all bounding boxes on each image to several trees in the following way: each node represents a bounding box. If a bounding box B_a is mostly included in another bounding box B_b (indicated by a ratio close to 1's, defined as overlap of B_a and B_b over B_a), the corresponding tree node T_a should be a child node of the tree node T_b. This condition will be applied to all bounding boxes in an image. Figure 3.1 shows two examples of this type of mapping.

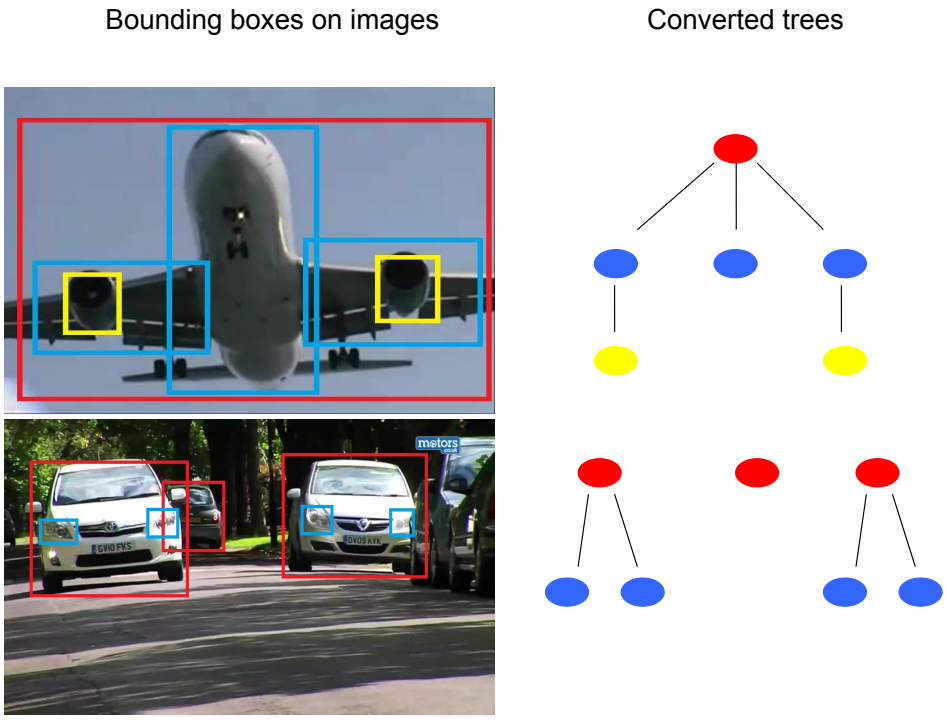

Figure 3.1. Examples of bounding boxes and their corresponding trees

We then propose a method [79] to build multiple trees by considering all bounding boxes in an image. First, we sort all bounding boxes by their areas (i.e., width * height) in descending order. Second, we use the first bounding box (i.e., the biggest one) to create a root node as the initial tree and gradually process the remaining bounding boxes one-by-one. For each bounding box, we check if it can be inserted in any of the existing trees by calculating the ratio of the overlap between the bounding box of the root of the tree and the candidate bounding box to the candidate bounding box. If the computed ratio is larger than a pre-set threshold, we insert the candidate bounding box into the tree. Otherwise, the bounding box cannot be inserted into any of the existing trees. We then use this bounding box to create a new tree by setting it as the root node of the new tree. The detail of the algorithm is shown in Algorithm 1.

The detail of the tree insertion function is summarized in Algorithm 2, where the input is a bounding box, the root node of a tree (i.e., treeRoot), and a predefined overlapping threshold. The output is the altered tree after the insertion. It should be

Algorithm 1 Convert multiple bounding boxes in an image to multiple trees

input: *bbs*: multiple bounding boxes; *thresh*: a pre-defined threshold for overlap
output: *trees*: multiple trees built based on bounding boxes
function name: convertBbsToTrees(bbs, thresh)
sortedBbsDescend ← sortBySizes(*bbs*) /*sort bbs by sizes in descending order*/
trees ← new List /*initialize a new list to place all the root nodes of resultant trees*/
for *bb* in *sortedBbsDescend* **do**
  insertSuccess ← False
  for *treeRoot* in *trees* **do**
    if the ratio of the overlap between *treeRoot.Bb* and *bb* to *bb* ¿ *thresh* **then**
      insert(*treeRoot*, *bb*, *thresh*) /*see Algorithm 2 for details of this function*/
      insertSuccess ← True
      break
    end if
  end for
  if *insertSuccess* is Ture **then**
    continue /* process next bounding box */
  else
    newTreeRoot ← createNewTreeNode(*bb*)
    trees.append(*newTreeRoot*)
  end if
end for
return *trees*

Algorithm 2 Insert a bounding box into an existing tree

input: *bb*: a bounding box; *treeRoot*: the root node of a tree; *thresh*: a pre-defined threshold for overlap
output: altered tree after insertion
pre-condition: the ratio of overlap of *treeNode.rootBb* and *bb* to *bb* ¿ *thresh*
function name: insert(*treeRoot*, *bb*, *thresh*)
if *treeRoot.children* is NULL /*base case*/ **then**
  newTreeRoot ← createNewTreeNode(*bb*)
  treeRoot.addChild(*newTreeRoot*)
  return
end if
for each child *c* of *treeRoot* **do**
  if the ratio of overlap of *c.Bb* and *bb* to *bb* ¿ *thresh* **then**
    insert(*c*, *bb*, *thresh*) /*recursive call*/
  end if
end for
newTreeRoot ← createNewTreeNode(*bb*) /* all children of treeNode do not have sufficient overlap with *bb* */
treeRoot.addChild(*newTreeRoot*) /* Add *newTreeRoot* as a child of *treeRoot* */
return

noted that the input bounding box is mostly included in the bounding box corresponding to the input root of the tree based on pre-condition in Algorithm 1. As a result, calling this function will insert the input bounding box at the right location of the tree, where its parent corresponds to the smallest bounding box which has the sufficient overlap with the inserted bounding box. We use recursion to implement the insert function. Let us denote the function by insert(treeRoot, bb, thresh). The base case is that the treeRoot has no children. In this case, we simply create a new node for bb and add this new node as a child of treeRoot. If the treeRoot has children, we will check if bb is tightly included in the bounding box corresponding to any of the children. If yes, we make a recursive call by passing this child, bb and thresh as parameters. If bb is not tightly included in the bounding box corresponding to any of the children, we simply create a new node using bb and add this new node as the child of treeRoot. The details of this algorithm are shown in Algorithm 2

The time complexity to compute the visual feature representation for an image with width m and height n is $O(mn)$. The time complexity to convert all the bounding boxes in an image to the structure of the hierarchical trees is $O(klog(k))$, when there are total of k bounding boxes in an image. Specifically, sorting these k bounding boxes takes $O(klog(k))$ in time complexity. Inserting sorted bounding boxes into an appropriate position in the trees takes $O(klog(k))$ in time complexity.

3.1.2.3 Learning Schema to Compute Confidence

Based on sections 3.1.2.1 and 3.1.2.2, each object proposal bounding box has a tree associated with it. If the root node of a tree T corresponds to a bounding box B, the children of T correspond to all bounding boxes inside of B. We compute feature F from box B as the root node feature. Hence, for each bounding box B_i, F_i represents its visual feature and tree T_i with a certain number of children representing its hierarchical structure, whose root node corresponds to bounding box B_i.

Since each bounding box maps to a tree, we compare two bounding boxes by comparing their corresponding trees and their visual features. We define the distance between two bounding boxes as:

$$\text{Dist}(\text{B}_i, B_j) = b \times FeatureDist(F_i, F_j) + (1 - b) \times TreeDist(T_i, T_j)$$

where B_x, F_x, and T_x represent xth Bounding Box, Feature, and Tree, respectively. b is the balancing factor between the overall visual feature distance and the hierarchical tree distance. $FeatureDist(F_i, F_j)$ is a distance measure between two feature vectors and is computed by normalized Euclidean distance (i.e., Mahalanobis distance) with the Euclidean distance defined as follows:

$\text{FeatureDist}(\text{F}_i, F_j) = \sqrt{\sum_{k=1}^{23}(F_{i,k} - F_{j,k})^2}$

$TreeDist(T_i, T_j)$ is a distance measure between two general trees. The edit distance used to measure between two trees as defined in 3.1.2.3. Here we use the Euclidean distance to compare two visual features and use the edit distance to compare two trees.

$\text{Tree}(\text{T}_i, T_j) = A(T_i, T_j) + D(T_i, T_j) + R(T_i, T_j) where \text{A}(\text{T}_i, Tj)$ represents the number of additions, $D(T_i, Tj)$ represents the number of deletions, and $R(T_i, T_j)$ represents the number of replacements.

The edit distance computes the minimum number of operations needed to convert from one tree to another tree using three types of operations: adding a node, deleting a node, replacing a node.

For each object bounding box B_i, we compute its distance to each bounding box B_j in the training data using $Dist(B_i, B_j)$ and find the n closest B_js. Among n B_js, there may be p objects and q non-objects (i.e., $p + q = n$). So, we define a confidence score by:

$$Conf_B = \frac{p}{n} \tag{3.6}$$

The higher the confidence score, the more likely the proposal being an object. Last, for each image, we order all object proposals by their confidence scores in the descending order. The time complexity of computing the confidence is the same as K-Nearest-Neighbor (KNN) algorithm (e.g., $O(kgd)$) where g is the cardinality of the training set and d the dimension of each proposal or sample. Specifically, computing Euclidean distance for one testing tree with all g training trees takes $O(g)$ in time and computing the edit distance for one testing tree with all g training trees also takes $O(g)$ in time.

3.1.3 Experimental Results

To evaluate the proposed method, experiments are conducted on YouTube-Objects dataset V2.2 [80]. The dataset is composed of the videos that are collected from YouTube and from names of 10 object classes of the PASCAL VOC Challenge [65]. There are 9 to 24 videos for each object class. The duration of each video varies between 30 seconds and 3 minutes. The videos are weakly annotated, where each video contains at least one object of the corresponding class.

The entire dataset contains a total of 720,000 frames with 6975 bounding-box annotations. The annotated frames are divided into the training set and the testing set. One instance has been annotated per frame in the training set and all instances have been annotated in the testing set. In total, there are 4306 annotated frames with 4306 bounding box annotations in the training set. There are 1781 annotated frames with 2669 bounding box annotations in the testing set. Each video contains multiple shots with their annotated starting and ending frame numbers. Table 3.1 lists the 10 classes of objects (e.g., rigid and non-rigid objects) together with the number of videos and shots for each class.

Table 3.1. The number of videos and shots for each class of objects.

Rigid objects	Non-Rigid objects
aeroplane: 13 videos, 482 shots	bird: 16 videos, 175 shots
boat: 17 videos, 191 shots	cat: 21 videos, 245 shots
car: 9 videos, 212 shots	cow: 11 videos, 70 shots
motorbike: 14 videos, 444 shots	dog: 24 videos, 217 shots
train: 15 videos, 324 shots	horse: 15 videos, 151 shots

We perform two experiments to evaluate the performance of the proposed system. The first experiment is to compare the detection rate of the proposed hierarchical model and the EdgeBox detector using the same selected frames from annotated training and testing frames. We consider this as the baseline comparison. The second experiment is to compare the proposed hierarchical model and the EdgeBox in the context of running the entire proposed workflow which includes stable frame sequence generation, EdgeBox proposal generation, and hierarchical model with parameters tuned for the whole system. To this end, the input frames for learning object proposals are the output frames from

the stable sequence generation module. The aim of this experiment is to tune the parameters of the hierarchical model and fit the hierarchical model in the context of the entire system. We describe the details of these two experiments in the next two subsections.

3.1.3.1 Baseline Comparison

Due to the nature of the video dataset, the frame sequence within the same shot changes gradually and the differences between frames within the same shot are small. The YouTube-Objects V2.2 dataset has the beginning frame number and ending frame number for each shot. Hence, we select the first labeled image within each shot as the representative image from that shot. Some shots do not have any labeled image, we simply skip them. As a result, all images we select are from different shots. We call these labeled images shot representative images. Among these shot representative images, 870 images are from the training set and 334 images are from the testing set. We use these 870 as training data and these 334 images as testing data to evaluate the hierarchical model as baseline.

Several parameters affect the performance of both EdgeBox and the proposed method. First, Intersection Over Union (IoU) is commonly used to measure the overlap of two bounding boxes. For two bounding boxes in an image, suppose the area of their overlap is S_1 and the area of their union is S_2, IoU $= \frac{S_1}{S_2}$. To decide whether a predicted bounding box correctly captures a true object, typically an IoU is computed between the predicted bounding box and the ground truth bounding box, which includes the object. If IoU is bigger than a pre-defined threshold, we conclude that the predicted bounding box correctly detects the object. The higher value the IoU threshold is, the more strict the correct detection is defined. Fig 3.2 shows the positive and the negative detected bounding boxes on the same images under three IoU values: 0.5, 0.7 and 0.9. We can see from the figure that the number of positive bounding boxes decreases as the IoU value increases. In other words, the detection rate decreases as the IoU threshold increases. Another parameter that affects the proposed system is the weight b in equation 3.1.2.3.

First, we apply EdgeBox [58] on every shot representative image in training and testing sets and generate top 50 bounding boxes per image. Since this is the video

Figure 3.2. The detected bounding boxes under three IoUs: positive and negative bounding boxes under three IoUs on the same images. Yellow bounding boxes show the ground truth on every image. Three columns from left to right show the detected bounding boxes under IoU values of 0.5, 0.7, and 0.9, respectively. The top row shows the positive bounding boxes and the bottom row shows the negative bounding boxes

dataset and the number of objects is relatively large in each image, top 50 bounding boxes generated by EdgeBox would detect nearly 70% of true objects. We use top $n = 50$ in our experiment. The values of EdgeBox parameters are: the step size of sliding window search is 0.8; the non-maximum suppression threshold is set to be 0.55, and the min score of boxes to detect is 0.01. EdgeBox will produce 50 bounding boxes together with their confidence scores. The balancing weight b in Eq. (3.1.2.3) is set to 0.5 and IoU is set to be 0.7 is used. To evaluate performance, we compute the recall rate of shot representative images based on their EdgeBox output and annotated ground truth labels in testing images. Here, the recall rate is computed by Eq.(3.7)

$$Recall = \frac{TP}{TP + FN} \tag{3.7}$$

where TP stands for the number of true positives and FN represents the number of false negatives.

We further employ the proposed tree-based hierarchical model on top n proposals to recompute the confidence score by using Eq. (3.6). The recall rate of shot representative images based the tree model is similarly computed.

Table 3.2. Comparison of the hierarchical method and the EdgeBox: the recall rates of the tree-based hierarchical method and EdgeBox for top n proposals

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
EdgeBox	.10	**.18**	.22	**.27**	.29	.31	.33	.35	.36	.37	.39	.41	.42	.43	.43	.44
Proposed	**.11**	.17	**.22**	.26	**.29**	**.32**	**.37**	**.41**	**.44**	**.46**	**.47**	**.49**	**.51**	**.53**	**.54**	**.54**
n	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
EdgeBox	.46	.48	.49	.50	.51	.51	.52	.53	.55	.55	.57	.58	.58	.59	.59	.60
Proposed	**.54**	**.55**	**.55**	**.55**	**.57**	**.58**	**.58**	**.59**	**.59**	**.60**	**.60**	**.60**	**.61**	**.62**	**.62**	**.63**
n	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
EdgeBox	.60	.61	.61	.62	.62	.62	.63	.63	.64	.64	.64	.65	.65	.65	.65	.66
Proposed	**.63**	**.64**	**.64**	**.64**	**.64**	**.64**	**.65**	**.66**	**.66**	**.66**	**.66**	**.66**	**.67**	**.67**	**.67**	**.67**
n	49	50														
EdgeBox	.66	.67														
Proposed	**.67**	**.67**														

Table 3.2 shows the comparison of the recall rate of EdgeBox and the proposed method using top 1 to 50 boxes. We clearly see that the proposed method outperforms EdgeBox for most values of n. The convergence at n=50 is due to the fact we use all 50 boxes from EdgeBox to re-order proposals. It shows that the proposed method is able to use fewer top proposals to achieve the same recall as achieved by top 50 proposals generated by EdgeBox.

3.1.3.2 Comparison of the System Performance Under Different Parameters

For this experiment, instead of selecting the first frame from each shot based on the ground truth, we apply stable sequence generation as described in chapter 2 to get all stable frame sequences. For key frame extraction component in stable sequence generation, we set the parameter κ (i.e., the proportion of key frames over the total number of original frames) to be 0.04. Among the selected key frames, we only keep the key frames that have ground truth labels of objects. The original dataset has a training set and a testing set for annotated frames. Hence, the key frames with the annotation in the original training set are chosen as the training frames for the hierarchical model. Similarly, we choose key frames with the annotation in the original testing set as the testing frames for the tree-based hierarchical model. Consequently, we have 4240 frames in the training set and 65 frames in the testing set for evaluating the hierarchical model.

We use the 50 bounding boxes generated from EdgeBox as the input to the hierarchical method. When designing the experiments, we want to test the relationship between the IoU values and the recalls. In addition, we also want to test the performance of the proposed method by using different balancing weights b defined in Eq. (3.1.2.3).

From what has been discussed above, we choose three typical values for IoU threshold: 0.5, 0.7 and 0.9, in the experiments. Also, we choose three values of balancing weight which are 0.4, 0.5 and 0.6, in the experiments. Therefore, we design three sets of experiments to compare the performance of EdgeBox and the proposed hierarchical method.

In the first experiment, we label each of these 50 bounding boxes in the training set as positive (i.e., contains object) if the IoU is greater than or equal to 0.5 or as negative (i.e., does not contain object) if IoU is less than 0.5. The same labeling procedures are applied to the testing data to obtain the ground truth in order to compute the recalls. We then apply the proposed method with three different values of b (0.4, 0.5 and 0.6) to compute a new confidence score for every proposal and re-order them in the descending order for every testing image. The recall rate can then be computed by Eq. (3.7) based on the labeled ground truth. The comparison of results is shown in Fig. 3.3. We can see that the overall recall rates for the hierarchical model regardless the weight values are higher than the recall rates for EdgeBox. The three recall curves corresponding to the three weights are very close and the recall curve with the weight value of 0.4 is the best. The recall curve with the weight value of 0.5 is slightly higher than the recall curve with the weight value of 0.6.

In the second experiment, we label each of these 50 bounding boxes in the training set as positive (i.e., contains object) if IoU is greater than or equal to 0.7 or as negative (i.e., does not contain object) if IoU is less than 0.5. For boxes with IoU between 0.5 and 0.7, since they do not show obvious characteristics of object or non-object, we ignore these boxes to achieve better performance as suggested by literature, we simply ignore them. The same labeling procedures are applied to the testing data to obtain the ground truth and compute recall. Similar to the first setting, we apply the proposed method with three weight values such as 0.4, 0.5 and 0.6 to compute a new confidence score for every proposal and re-order them in the descending order for every testing image. The

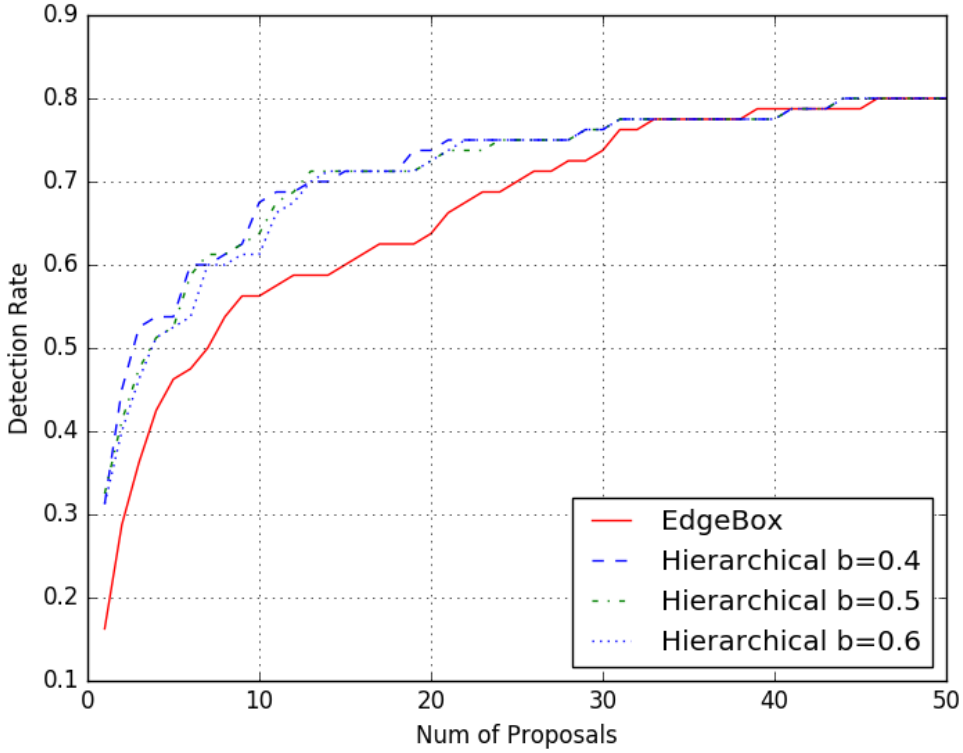

Figure 3.3. Comparison 1 of the EdgeBox and the proposed method: the recalls of EdgeBox and the proposed method when $IoU = 0.5$ and $b = 0.4, 0.5, 0.6$

recall rate can then be computed by Eq. (3.7) based on the labeled ground truth. The results are shown in Fig. 3.4. Comparing with the results in Fig. 3.3, we can see that the overall recall rates of IoU = 0.7 are lower than the recalls for IoU = 0.5. This is consistent with the definition of IoU. Fig. 3.4 clearly shows that the hierarchical model with different weights have higher recall than EdgeBox. All three recall curves are close to each other. Recalls with the weight value of 0.6 are slightly better than recalls with the other two weights when the number of proposals is greater than 20.

In the third experiment, for the training set, we label each of these 50 bounding boxes in the training set as positive (i.e., contains object) if the IoU is greater than or equal to 0.9 and or negative (i.e., does not contain object) if the IoU is less than 0.9. The same labeling procedures are applied to the testing data to obtain the ground truth and compute recall. We then apply the proposed method with b's of 0.4, 0.5 and 0.6 to compute a new confidence score in the same way as we do in the first two settings.

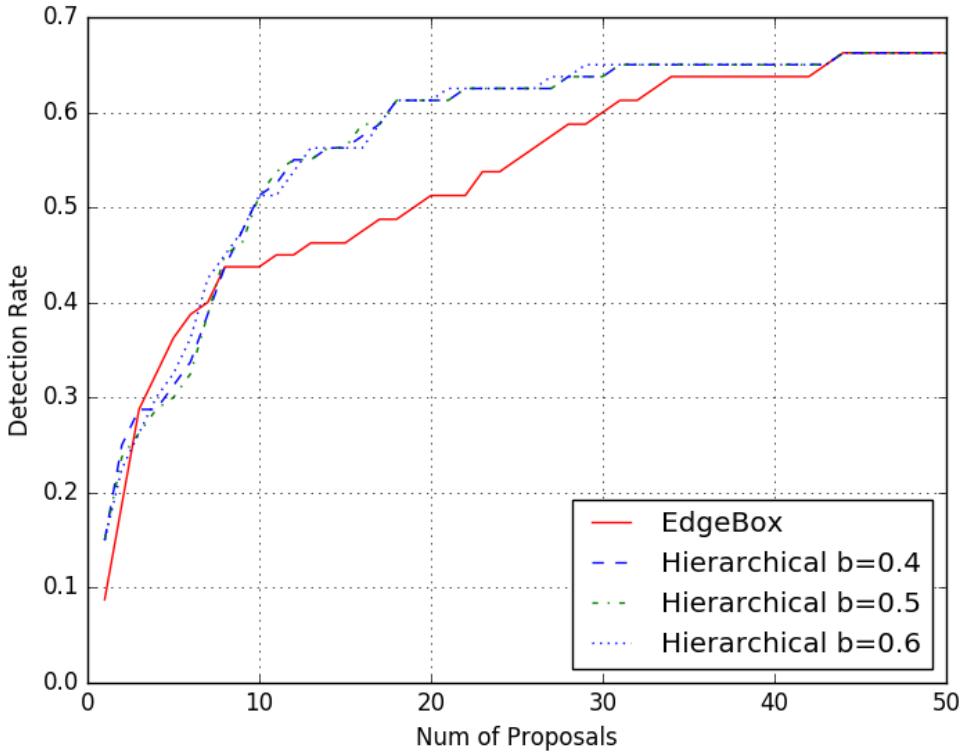

Figure 3.4. Comparison 2 of the EdgeBox and the proposed method: the recalls of EdgeBox and the proposed method when $IoU = 0.7$ and $b = 0.4, 0.5, 0.6$

The recall rate can then be computed based on the labeled ground truth. The results are shown in Fig. 3.5. Comparing with Figures 3.3 and 3.4, we can see that the overall recall rates of IoU = 0.9 are significant lower than the recall for IoU = 0.5 and IoU = 0.7. This matches the intuition based on the definition of IoU. In Fig. 3.5, overall all three curves from the hierarchical model are close to each other and show higher recall than the curve of EdgeBox. Recalls with weight of 0.6 are slightly higher than recalls with the other two weights when the number of proposals is greater than 25.

3.1.4 Summary

Generating class-agnostic object proposals followed by classification has become a common paradigm for object recognition in recent research. Current state-of-the-art approaches typically generate generic objects, which serve candidates for object categories classification. Since these objects proposals are generic whereas the categories for

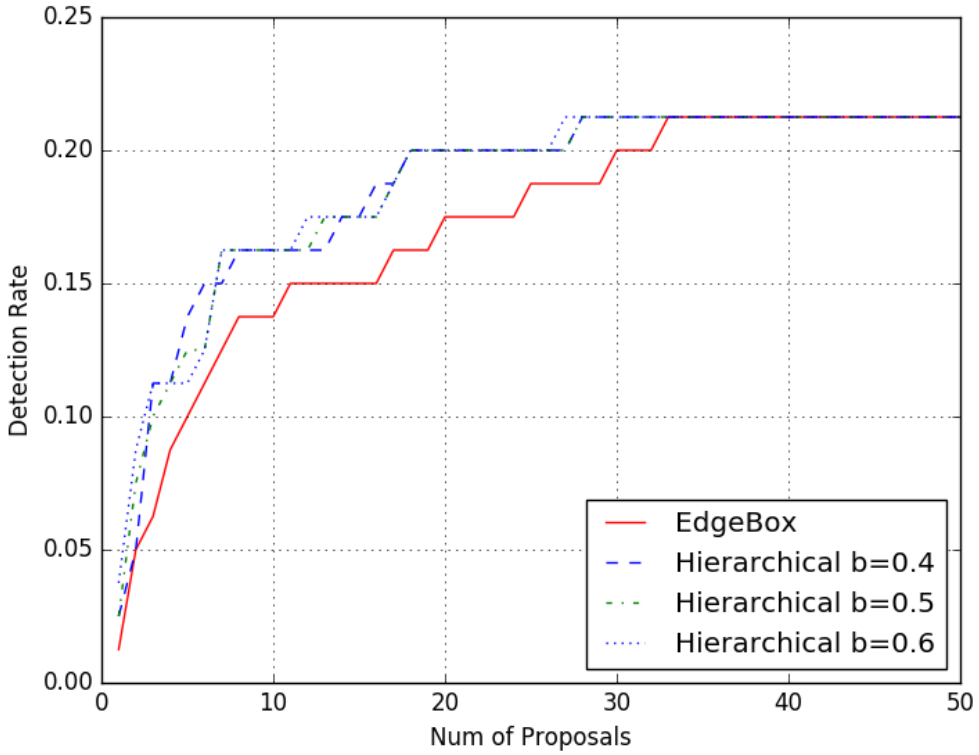

Figure 3.5. Comparison 3 of the EdgeBox and the proposed method: the recalls of EdgeBox and the proposed method when $IoU = 0.9$ and $b = 0.4, 0.5, 0.6$

classification are domain specific, there is a gap between these two steps. In this section, by taking advantages of the intrinsic structure and the complexity of each object's category, we propose a novel tree-based hierarchical model to learn object proposals based on the generic proposals generated by the existing object proposal generation methods. First, we develop a tree-structure representation for each object to capture its intrinsic hierarchical structure. Second, we propose a compact feature to efficiently represent the object's visual appearance. Third, we formulate a learning schema to evaluate the objectness of each proposal from the perspectives of visual appearances and complexity of sub-parts. Experiments show the significant improvement of our proposed approach over the state-of-the-art methods for object detection rate.

3.2 Learning Object Proposal Tracks Using Temporal Objectness

After detecting the object proposals in the first frame of each stable frame sequence,

we can use the positions of these object proposals as initial positions of candidate objects to estimate the object proposals in the following frames of each sequence. We use tracking methods to estimate the optical flow of each object proposal over each sequence. Compared with the approaches of detecting object proposals in every frame, estimating the optical flow of each proposal via tracking offers two advantages. First, since tracking considers the temporal information, the object proposals across different frames are more consistent; Second, detecting object proposals in individual frames could be costly and some existing tracking methods can perform estimation in real-time.

Most tracking methods take the initial positions of objects in the first frame as the input and estimate the corresponding positions of objects in the next frame. They keep estimating the object positions in every single frame based on the estimated positions of objects in the previous frame. A lot of tracking methods have been developed over the past two decades.

Tracking-Learning-Detection (TLD) [81] is one of the state-of-the-art tracking methods. It builds upon the classic Lucas and Kanade method [82] with the three assumptions for the objects' movements in the video. The first assumption is the brightness consistency, which means every pixel's intensity from a frame moves to another position with the approximately same intensity in the next frame by a small displacement distance d. The second assumption is the temporal persistence, which means each pixel moves to another position by a small displacement distance d. The third assumption is the spatial coherence, which means all pixels in the Regions Of Interest (ROIs), e.g., object regions, move coherently in the same direction. Based on these three assumptions, tracking is formulated to a minimization problem of the displacement distance d, which is expressed by pixels in the current frame, pixels in the next frame, and the gradient of pixels in the current frame. For every pixel in the ROIs, tracking predicts its position in the next frame. However, due to the scale changes, the predicted positions for some pixels may not be reliable. Hence, the TLD method develops a forward-backward error measure to refine the tracking result. It uses the Lukas-Kanande method to track forward to estimate all pixel positions in the next frame. Then it uses the same method to track back pixel positions from the next frame to the current frame. The pixels with either large distance or large value change are filtered out during the tracking process

and a bounding box containing the remaining reliable pixels as the predicted position is located in the next frame. Whenever the tracked objects are lost, the TLD algorithm re-detects the objects to re-initialize the tracker. The advantage of the TLD algorithm is that it is robust to track objects with scale changes and occlusion. However, it could only track a single object.

Most visual object tracking methods are able to track a single object in a video sequence. Simultaneously tracking multiple objects in videos is still a challenging problem. Zhang et. al. [83] propose a tracking method called Structure Preserving Object Tracker (SPOT), which is able to track multiple objects with high accuracy. To the best of our knowledge, SPOT track is the first work that can track multiple generic objects. It is a model-free tracker requiring the initial positions of objects in one frame. The SPOT tracker uses HOG features to represent each image patch to be tracked and an SVM classifier to predict the position of each patch in the next frame. It also considers the geometric relationship of the multiple objects that are being tracked. To this end, it proposes an objective function to include not only the similarity of the observed patch features and the classifier weights among all objects but also a penalized deformation score measuring how much a configuration compresses or stretches between the tracked objects. A configuration here means a set of bounding boxes for these multiple objects with fixed positions. By maximizing this objective function, the tracker predicts the positions of each object while preserving the overall structures of multiple tracked objects. Hence, the SPOT tracker performs especially well under the circumstances, where all objects being tracked move in the same direction in the videos.

3.2.1 Introduction

After learning object proposals using the hierarchical model, we generate high quality proposals on key frames. Each of these key frames serves as the first frame of each stable frame sequence as described in Chapter 2. In this section, we use the object proposals generated on each key frame to initialize multiple object trackers to track the objects across each frame sequence. At the same time, we learn object proposal tracks using a novel temporal objectness [84].

Intrinsic natures of different appearances between sub-regions of objects and non-objects in optical flows lead to more visual consistency for object proposals. Hence, visual variations in different sub-regions of a video sequences over time is a good indicator for likeliness of objects. We propose a method that dynamically measures the objectness of each proposal by exploiting temporal consistency within each optical flow during tracking. We develop a block-based feature representation using object's spatial property and define an objectness measure using the temporal changes of this spatial representation. As a result, the proposed temporal objectness learns good object proposals over a short period (e.g., less than 1 second). The proposed method is model-free and can be used to simultaneously learn and track object proposals without training. Experiments on a video dataset shows that the proposed approach significantly outperforms state-of-the-art methods in terms of the precision-recall measure.

3.2.2 The Proposed Temporal Objectness Method

Since applying object detection on individual frames of a video is undesirable, we seek to incorporate detection and tracking to effectively detect objects in each frame of a video. For object detection in static images, some recent generic object detection algorithms (e.g., EdgeBox [58] and BING) could quickly locate the patches that likely contain objects. These patches serve as candidates for true object locations. Since objects differ from non-objects (i.e., pure backgrounds or regions containing partial objects and partial backgrounds) by their structural closed boundary, the fast generic object detector exploits this property to detect objects. For example, EdgeBox exploits the edges of objects and uses the boundaries that are wholly enclosed in a bounding box to measure objectness. BING uses norm of gradients to measure objectness. However, these detectors tend to assign high objectness scores to patches that contain either true objects or partial objects and partial backgrounds, and assign low objectness scores to patches containing relatively pure background. Therefore, further distinguishing the true object patches and patches that contain partial objects and backgrounds can enhance the discriminative power of classifying objects and non-objects. Compared to images, videos contain additional temporal information. Since tracking methods aim to estimate optical flows over temporal frames by assuming that the objects move coherently, the

consistency of patches in optical flows can be a good indicator for objectness. We did some preliminary studies and came up with the following hypothesis: the object patches tend to have more consistency than patches that include partial objects and partial backgrounds. In other words, object patches tend to have less variations than patches that include partial object and partial background in an optical flow. Fig. 3.6 shows an example of the comparison of negative bird patch tube and positive bird patch tube. It clearly demonstrates the less variations in the positive bird patch tube. Driven by this hypothesis, we propose a compact feature representation for spatial appearance of each patch, and formulate a temporal objectness to learn object proposals.

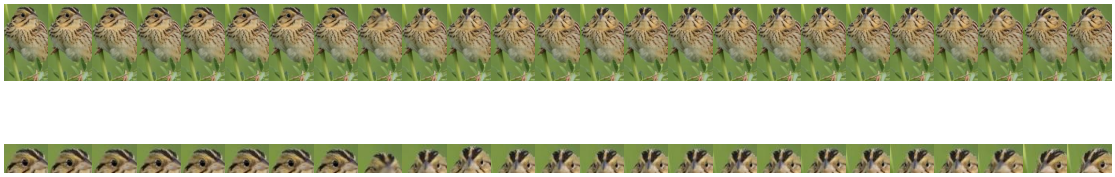

Figure 3.6. Sample images of positive and negative patch tubes: positive bird patch tube (upper) and negative patch tube (lower), where patch tube is a set of patches across frames in the temporal order generated by tracker

3.2.2.1 Block-based Spatial Feature Representation

All objects could be roughly divided into: rigid objects and non-rigid objects. Examples of rigid objects include airplanes, cars, and boats. Example of non-rigid objects include human, birds, and dogs. A sub-region of an image is called a patch. In a video, tracker generates a set of patches across frames in the temporal order. This set of patches is called a patch tube. Our preliminary studies show that patch tubes for objects in optical flow during tracking generally have much higher consistency and lower variations over time than patch tubes for non-objects. Non-rigid objects typically have consistency in optical flow as a whole part but some internal part/parts of the object might move and change appearances more than other parts depending on the structure of deformable objects. For example, the body of a running horse usually does not change appearances as much as the four legs.

To measure the relative consistency in appearances of both rigid and non-rigid objects, we develop the following block-based feature representation. First, we divide

the patch of interest into a 3x3 grid, resulting in a total of 9 blocks. Within each of 9 blocks, we develop a compact and powerful feature vector to represent the local visual properties. Using the variation of pixel values in red, green, and blue channels, the entropy, the spatial distribution of 3 color channels, and the aspect ratio, we define the feature representation as a 20-dimensional feature vector. These 20 values in the feature vector include standard deviations of pixel intensities in rgb color channels (3 values), the entropy of the luminance-based histograms (1 value), relative frequencies of 5 bins with top frequencies in rgb color channels (15 values), and the aspect ratio of the patch (1 value). The first 19 features are the same as the corresponding 19 features described in section 3.1.2.1. The aspect ratio is the new feature introduced and is computed as the ratio of the width to the height of a patch.

Each patch has 9 blocks with each block represented by a 20-dimensional feature vector. For non-rigid objects, the changes in some blocks with moving parts are typically bigger than the changes in other blocks with non-moving parts. Using this block-based spatial feature representation, we capture the relatively steady regions by measuring the changes of each patch in the optical flow represented by the patch tube.

3.2.2.2 Temporal Objectness

We describe how to measure temporal objectness using this block-based spatial feature representation and temporal information in detail. Our goal is to measure how the patch's visual appearance changes over consecutive frames over time by using the block-based spatial features. Since only parts of the object move in the non-rigid object patch tube, we would like to measure the overall variation of consecutive patches at relatively static regions of the object in a patch tube. Hence, we first compute 9 block-wise distances between neighboring patches in the patch tube and keep the least 5 distances. Next, we compute the average of these 5 distances as the neighbor change between two neighboring patches. Finally, we compute the median of all neighbor changes across the patch tube as a measure of overall variation. Since more variation indicates less likeliness of objects, we subtract this median from zero as the temporal objectness. Algorithmically, let P_{t_i} represent a patch in frame t_i, B_{x,y,t_i} represent the spatial feature vector of a block at the xth horizontal partition and yth vertical partition in patch P

of frame t_i where x=1,2,3, y=1,2,3, and t_i = 1,...,n, and n is the number of frames in a stable frame sequence. A tracked patch tube can be represented as $\{P_{t_i}\}$ and the corresponding block features are $\{B_{x,y,t_i}\}$. The Algorithm 3 describes how to compute temporal objectness o for $\{B_{x,y,t_i}\}$.

Algorithm 3 Compute the temporal objectness for a patch tube

input: $\{B_{x,y,t_i}\}$ (a sequence of 20-D feature vectors)
output: o (a value represents temporal objectness)
$D_T \leftarrow$ new List /*temporal distances*/
for $t_i \leftarrow 1$ to $n-1$ **do**
  $D_N \leftarrow$ new List /*neighbor patch distances*/
  for $x \leftarrow 1$ to 3 **do**
    for $y \leftarrow 1$ to 3 **do**
      $d \leftarrow$ EuclideanDist(B_{x,y,t_i}, $B_{x,y,t_{i+1}}$)
      D_N.append(d)
    end for
  end for
  $sortedD_n \leftarrow$ sort(D_N, order=ascending) /*sort 9 values in D_N*/
  $min5D \leftarrow$ first 5 elements of $sortedD_N$
  $aveD \leftarrow$ average($min5D$)
  D_T.append($aveD$)
end for
$o \leftarrow$ -median(D_T) /*find 0 minus median of $n-1$ temporal distances in a patched tube*/

Temporal objectness o represents zero minus the central tendency measure of changes over time sequence for each patch tube. We use the median value on temporal distances of all pairs of adjacent frames to compute temporal objectness since it is more robust than the mean value when handling outliers.

After computing temporal objectness, we use it as a measure of likeness for an object and rank the temporal objectness for all object proposal tracks. Precision and recall using different thresholds ranging from top 1 to top n proposals are then computed to evaluate the performance of the proposed temporal objectness method.

The time complexity for calculating the feature is $O(mn)$ for a patch with width m and height n. The time complexity for calculating the temporal objectiveness is $O(kLog(k))$, where k is the total number of frames in the tube.

3.2.3 Experimental Results

We perform two experiments on YouTube-Objects V2.2 database to evaluate the temporal objectness model. First, based on the ground truth of the shot, we pick a fixed number of frames to individually apply the EdgeBox detector and the proposed temporal objectness measure to generate top object proposals for comparison. We treat this as the baseline. Second, we compare the EdgeBox and the proposed method on the stable frame sequences, which are selected by the proposed stable frame sequence generation model. The purpose for the second setting is to test the performance of the temporal objectness in the context of the entire proposed system. We describe the experiments and results for both settings in the following two sections.

3.2.3.1 Baseline Comparison

Since not all frames are annotated and all instances have been annotated in the testing set for annotated frames, we select up to 10 annotated frames from each shot in each video in the testing set as the initial frame for object detection. Specifically, if a shot has 10 or more annotated frames, we select the first 10 annotated frames. If a shot has fewer than 10 annotated frames, we select all annotated frames. Due to the sparse annotation of frames, the first 10 annotated frames have sufficient temporal gaps along the frame numbers and there are no duplicated frames for initializing the tracker. After selecting up to 10 frames per shot for all shots, this data set contains frames from different shots and therefore is called a shot representative frame set. For each frame in this shot representative frame set, we apply EdgeBox detector [58] to generate object proposals and use EdgeBox objectness to select top 20 proposals, whose locations are used as the initial object locations for tracking. The parameters of EdgeBox are set as follows: the step size of the sliding window is 0.8; the non-maximum suppression threshold is 0.55, the min score of boxes to be detected is 0.01, and the maximum number of boxes to be detected equals 100. After selecting top 20 bounding boxes, we use them as initial positions and apply SPOT tracking [83] for each of boxes for a total of 20 consecutive frames to generate optical flows in video. We chose 20 frames for tracking because empirical studies suggested that it is long enough to have discriminative temporal objectness. The duration of 20 frames is typically less than 1

second so most tracked patch tubes still cover the initial region without drifting, even in an unconstrained video, unless the video shot has big changes or the tracked objects move out of the scene. Since SPOT tracker [83] uses constraints among multiple objects for initialization, we use frames that have at least 2 detected positive objects to initialize tracker. For each object track covered in 20 frames, we compute its temporal objectness as described in Algorithm 3. Since we use the annotated positions of objects in the initial frames to compute recall rate and precision of top n proposals, we need to make sure that each tracked patch tube correctly contains objects without drifting (i.e., an object become a non-object or vice versa). To this end, we manually remove the last few patches to make sure each patch in a tube is consistently tracked (i.e., containing objects from start to end).

In the end, we use 425 frame sequences (i.e., 260 rigid and 165 non-rigid frame sequences) which contain 687 annotated objects (i.e., 362 rigid objects and 325 non-rigid objects) in our experiments. For comparison, we rank patch tubes based on EdgeBox objectness and select top 1 to top 20 patch tubes from each of 20 frames in the frame sequence to compute precision and recall at each level from 1 to 20. We also use the proposed temporal objectness to select from top 1 to top 20 patch tubes from each frame sequence to compute precision and recall. Fig. 3.7 shows the precision-recall (PR) curves of the proposed method versus EdgeBox objectness for all 10 categories of objects. It clearly shows that the proposed method significantly outperforms the state-of-the-art EdgeBox method overall. The curves merge at the top 20 patch tubes because we use all 20 tubes generated from EdgeBox. From this comparison, we can see that the proposed objectness learns better object proposals than EdgeBox within top 20 proposals by considering temporal information. Fig. 3.8 and Fig. 3.9 show the PR curves for rigid and non-rigid objects, respectively. For rigid objects, we can see from Fig. 3.8 that the proposed method outperforms EdgeBox when the precision is greater than 0.26. This corresponds to extracting approximately top 10 proposals per image. For non-rigid objects, we can see from Fig. 3.9 that the proposed method significantly outperforms EdgeBox since its PR curve covers a significantly larger area. It is clear that the proposed method performs especially well on non-rigid objects due to its computation of temporal objectness using the minimum 5 block-wise distances on block-based features. Overall,

the proposed method achieves significant improvement in generating accurate proposals for rigid and non-rigid objects compared with the EdgeBox method. For object detection in videos, which typically employs tracking, temporal objectness can be computed to improve the quality of the object proposals. In other words, incorporating the proposed temporal objectness in video can achieve higher precision (i.e., fewer false positives) with the same number of proposals or the same recall rate with the fewer number of proposals. This improvement is obtained with the model-free learning (i.e., no training is needed) and with little additional cost on detection. Hence, this approach leads to further development of a system that incorporates tracking and detection with improved quality of object proposals. Fig. 3.10 shows a sample of qualitative results using the proposed object proposal learning method.

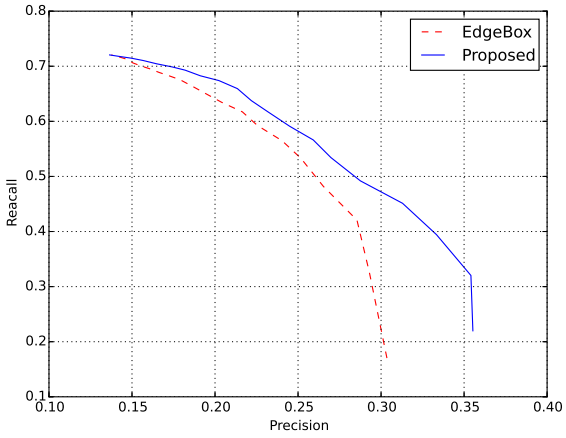

Figure 3.7. PR curves comparison for all objects: curves of the proposed temporal objectness and EdgeBox objectness for top n=1,...,20

3.2.3.2 Comparisons of the System Performance Under Different Parameters

In this experiment, we compare the EdgeBox detector and the proposed temporal objectness in the context of employing shot detection and key frame extraction. In other words, we apply the stable frame sequence generation module first and select the overlapping frames between the ground truth and key frames in the training and testing sets. This step is exactly the same as how we prepare training and testing images in section 3.1.3.2. We used top 20 object proposals on each frame in section

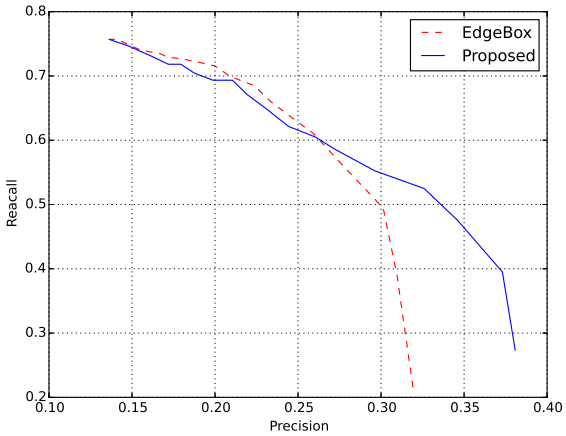

Figure 3.8. PR curves comparison for rigid objects: curves of the proposed temporal objectness and EdgeBox objectness for top n=1,...,20

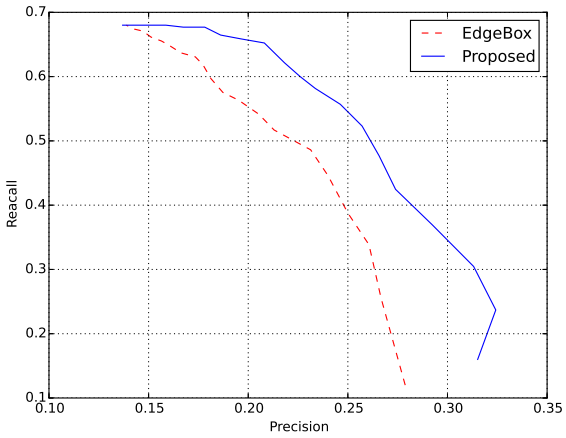

Figure 3.9. PR curves comparison for non-rigid objects: curves of the proposed temporal objectness and EdgeBox objectness for top n=1,...,20

3.1.3.2. Since the proposed temporal objectness method is training-free, we use the 65 frames in the testing set to evaluate the temporal objectness module. For testing the tuned parameters, on one hand, we would like to select more object proposals on each frame to have higher detection rate (i.e., greater than 0.7). On the other hand, if we select too many proposals, the experiments showed it affects the performance of tree-based hierarchical model. Hence, we keep top 50 proposals on each frame based on the ranking from EdgeBox detector. We then apply the proposed temporal objectness to re-rank the top 50 proposals. We finally compare both recall and PR curves for the

Figure 3.10. Qualitative examples of our object proposals: green bounding boxes show matched objects using IoU=0.7, red ones show missed ground truth. For 9 images from top left to bottom right, the number of matched bounding boxes and the number of missed objects are: 3,0; 2,0; 2,0; 2,1; 3,0; 2,0; 2,0; 4,1; and 2,1 respectively

different levels of IoU when defining the match between predicted bounding boxes and ground truth bounding boxes. We choose the values of IoU to be 0.5, 0.7 and 0.9 since they are the typical values for most experiments in prior literature.

Table 3.3 and Fig. 3.11 show the comparison of recalls of the EdgeBox and the proposed temporal objectness using top 50 object proposals under the IoU value of 0.5. It shows the proposed temporal objectness method significantly improves the recall over the EdgeBox for number of proposals from 1 to 50. To reach the recall value of 0.8, the EdgeBox needs 46 proposals and the proposed temporal objectness method only requires 21 object proposals.

Table 3.4 and Fig. 3.12 show the comparison of recalls of the EdgeBox and the proposed temporal objectness using the top 50 object proposals under the IoU value of 0.7. It shows the proposed temporal objectness method reaches much higher recalls than the EdgeBox for the number of proposals from 1 to 50. To reach the recall value of 0.6, the EdgeBox needs 30 proposals and the proposed temporal objectness method

Table 3.3. Detection rates 1 of the EdgeBox and the Temporal Objectness: the detection rates of the EdgeBox and Temporal Objectness for top n proposals under $IoU = 0.5$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
EdgeBox	.16	.28	.36	.42	.46	.48	.50	.54	.56	.56	.58	.59	.59	.59	.60	.61
Temporal Objectness	**.26**	**.39**	**.48**	**.49**	**.54**	**.57**	**.61**	**.62**	**.62**	**.64**	**.66**	**.66**	**.69**	**.69**	**.70**	**.74**
n	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
EdgeBox	.63	.63	.63	.63	.64	.66	.68	.68	.70	.71	.71	.73	.73	.74	.76	.76
Temporal Objectness	**.73**	**.75**	**.77**	**.79**	**.82**	**.82**	**.84**	**.84**	**.85**	**.85**	**.85**	**.85**	**.85**	**.85**	**.85**	**.85**
n	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
EdgeBox	.78	.78	.78	.78	.78	.78	.79	.79	.79	.79	.79	.79	.79	.80	.80	.80
Temporal Objectness	**.85**	**.85**	**.85**	**.85**	**.87**	**.87**	**.87**	**.87**	**.89**	**.90**	**.90**	**.90**	**.92**	**.92**	**.92**	**.92**
n	49	50														
EdgeBox	.80	.80														
Temporal Objectness	**.92**	**.92**														

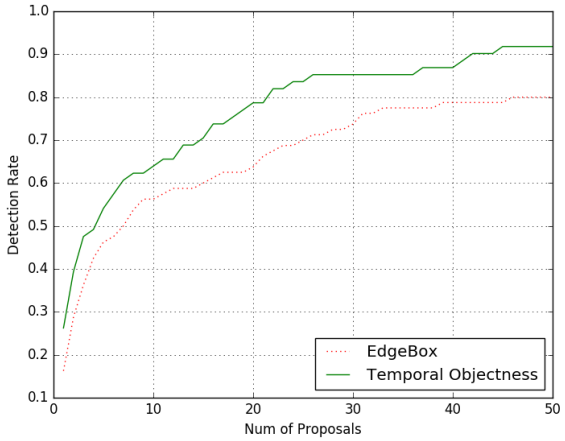

Figure 3.11. Comparison 1 of the EdgeBox and the Temporal Objectness: the recalls using the EdgeBox and Temporal Objectness under IoU = 0.5 and top n=1,...,50

only requires 10 object proposals.

The comparison of PR curves for the EdgeBox and the proposed temporal objectness under IoU = 0.5 is shown in Fig. 3.13. We can see that the proposed temporal objectness method clearly outperforms the EdgeBox detector. The upward shape of both lines are due to the randomness for the small value of top n proposals. In other words, when we only retrieve a few number of top object proposals, it is possible that both precision and recall increase. In this graph, we see that we can reach recall of 0.8 and precision of 0.25 approximately for the proposed temporal objectiveness method when returning top 22 object proposals.

Table 3.4. Detection rates 2 of the EdgeBox and the Temporal Objectness: the detection rates of the EdgeBox and Temporal Objectness for top n proposals under $IoU = 0.7$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
EdgeBox	.09	.19	.29	.32	.36	.39	.40	.44	.44	.44	.45	.45	.46	.46	.46	.48
Temporal Objectness	**.14**	**.21**	**.29**	**.36**	**.40**	**.45**	**.45**	**.50**	**.55**	**.62**	**.62**	**.64**	**.64**	**.69**	**.71**	**.79**
n	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
EdgeBox	.49	.49	.50	.51	.51	.51	.54	.54	.55	.56	.58	.59	.59	.60	.61	.61
Temporal Objectness	**.79**	**.79**	**.79**	**.79**	**.79**	**.79**	**.83**	**.83**	**.83**	**.83**	**.86**	**.86**	**.88**	**.88**	**.88**	**.88**
n	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
EdgeBox	.63	.64	.64	.64	.64	.64	.64	.64	.64	.64	.65	.66	.66	.66	.66	.66
Temporal Objectness	**.88**	**.90**	**.93**	**.93**	**.93**	**.93**	**.93**	**.93**	**.93**	**.93**	**.93**	**.93**	**.95**	**.95**	**.95**	**.95**
n	49	50														
EdgeBox	.66	.66														
Temporal Objectness	**.95**	**.95**														

The comparison of PR curves for the EdgeBox and the proposed temporal objectness under IoU = 0.7 is shown in Fig. 3.14. We can see that the proposed temporal objectness method also clearly has the better performance than the EdgeBox detector. Since we are more strict in defining the correct detection by using IoU of 0.7, the overall PR curves are a little worse than the PR curves under IoU = 0.5. It shows that we can reach the recall of 0.8 and precision of 0.1 approximately using the proposed temporal objectiveness when returning top 23 proposals. When setting the IoU value to be 0.9, the first frame from each sequence has at most one correctly detected proposal. The SPOT tracker could not be applied since it requires at least two objects for initialization.

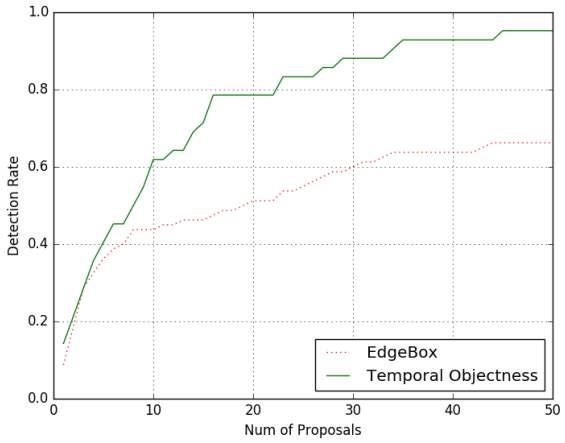

Figure 3.12. Comparison 2 of the EdgeBox and the Temporal Objectness: the recalls using the proposed temporal objectness and EdgeBox objectness under IoU = 0.7 and top n=1,...,50

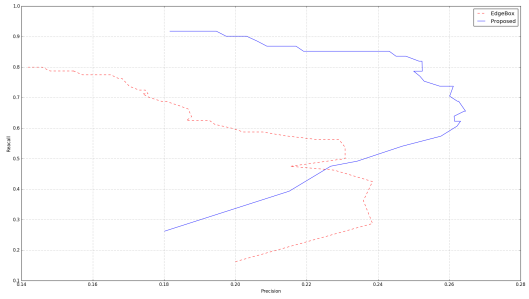

Figure 3.13. PR curves 1 of the Temporal Objectness and the EdgeBox: the curves using the proposed temporal objectness and EdgeBox objectness under IoU = 0.5 and top n=1,...,50

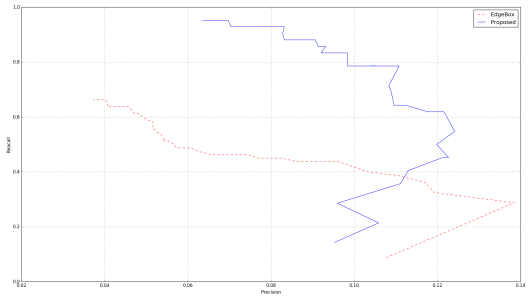

Figure 3.14. PR curves 2 of the Temporal Objectness and the EdgeBox: the curves using the proposed temporal objectness and EdgeBox objectness under IoU = 0.7 and top n=1,...,50

3.2.4 Summary

We propose a temporal objectness measure that takes both spatial feature and temporal changes into consideration and achieves higher performance in generating general object proposals than state-of-the-art methods. The advantage of the proposed method is that it is generally applicable to various object proposals and tracking methods. With tracking along the way, it can learn better object proposal representation using temporal information.

CHAPTER 4

OBJECT PROPOSAL CLASSIFICATION WITH DEEP NEURAL NETWORKS

After learning object proposal tracks, we have a large number of object proposal tracks resultant from one or multiple videos. Each object proposal track either contains non-object background or an object of a specific category. Suppose that there is a total of n object categories in the dataset domain, each object proposal track belongs to one of $n + 1$ categories (e.g., n object categories plus one background category). Based on the previous assumption that all object proposals in one track contain objects from the same category, we can simply select one object proposal (e.g., the first one) from each track as the representative object proposal and apply classification on it. The remaining object proposals in the same track have the same object category label. Hence, in this last module, the task is to classify each representative object proposal as one of $n + 1$ classes.

Traditional machine learning paradigms typically include two steps: engineering of feature representation and classification. Researchers have been working on developing new features with high discriminative power to represent instances from different classes in the past three decades. Different classifiers have been developed to be combined with the appropriate features for the classification task. Good feature representation is crucial for yielding high classification accuracy.

With the development of computing power such as faster CPUs and parallel processes with GPUs, along with the ever-growing big data, neural networks with deep architectures trained on large-scale data have achieved breakthrough performance on classification task since 2012. In the image processing and computer vision domain, the use of Convolution Neural Network (i.e., CNN) [85–87] achieved breakthrough performance for tasks such as image classification, object detection, and object recognition. The major difference between deep learning approaches and traditional machine learning approaches is that good feature representation is learned through deep neural networks

instead of engineered features developed by humans. By minimizing the loss function based on a certain type of classification error between predicted labels and true labels, weights and biases are gradually updated through backpropagation using the scholastic gradient descent for global optimization. Since features are learned by the global optimization function, they are better than traditional engineered features. That is the main reason that deep-learning-based methods have achieved striking results in recent five years.

In this chapter, we provide a detailed description of deep neural networks and how to employ deep learning to perform object proposal classification in the proposed framework. Section 4.1 provides an overview of the deep neural networks, together with the definition, some state-of-the-art methods, and the relevant concepts. Section 4.2 discusses some applications that are driven by deep neural networks. Section 4.3 introduces the basic structure of the neural networks and their basic components, named perceptrons. Section 4.4 discusses the backpropagation and the gradient descent algorithm, which are used to perform learning. Section 4.5 talks about different optimization techniques using objective functions, regularization functions, and hyper-parameter tuning methods. Section 4.6 introduces the CNN and its use in the proposed system to perform the classification of object proposals in the proposed system.

4.1 Overview

Unlike traditional machine learning, deep learning is a set of new algorithms inspired by the artificial neural network algorithm in machine learning but has been extended to deep architectures. The word "deep" here has two meanings. First, it refers to the height of the neural network. That is, the neural network has many more layers than the traditional neural networks. Second, it refers to the width of the neural network. That means the number of neurons in each layer is much larger than the number of neurons in each layer of the traditional neural networks. Even through the artificial neural network was introduced back in the 1980s [88], it has two limitations: First, the neural network typically requires a large amount of training data to have good performance on the testing data. Second, the computation power is limited, and the learning process is very computationally intensive. Various supervised and unsupervised machine learning and

data mining algorithms have been developed in the past 30 years. These algorithms include decision trees [89] , K-Nearest-Neighbors (KNN) [90], Support Vector Machine (SVM) [91], Naive Bayes [92], Regression [93], K-means [94] and many others. The performance and the choice of the classifiers depend on the specific problem and its context. Overall, SVM was one of the best classifiers for many problems before 2006.

In the past 20 years, as the growth of the electronic devices (e.g., computers, mobile devices, etc.) and Social Network Services, labeled data in different domains have been growing rapidly. Driven by Moore's law, the computational capacity of the machines has also been significantly increased with the advancement of hardware such as CPUs and GPUs. Driven by these two forces, artificial neural networks algorithm started achieving breakthrough performance and outperformed many state-of-the-art algorithms since 2006 [95], [96]. These breakthroughs occurred in the domains of image understanding, computer vision, speech recognition, and Natural Language Processing (NLP).

4.2 Applications

Driven by the significant progress of deep neural networks, many applications have been developed in multiple domains.

In the image processing and computer vision domain, some apps (e.g., Blippar [97]) are developed to automatically recognize the objects in real life through phone camera and provide assistive information such as wiki, knowledge graph, direct purchase, and video instruction of usage of each recognized object. For example, if a user scans a chair, he/she can then find the information such as the inventor of the chair, the history and the manufacturing process of the chair, and other related entities of the chair together with their information. If a user goes to a tourist place and sees a temple, he/she can scan the phone to recognize this temple. If a user sees a cool car, he/she can scan the app to recognize the model and made of the car and order for a test drive. If a user purchases a new cleaner, he/she can use the app to scan the cleaner and access the instructional video of how to use it once the cleaner is recognized. Users can scan the objects to check their relevant information or make purchases. Google photo app [98] can automatically assign the tags (e.g., objects and scenes) to everyone's photo collections on their mobile phones using the offline trained deep neural networks. Then the users

can easily search and locate the photos they are looking for by the query keywords such as "park", "mountain", "car" or "beach". Some other web services [99] provide API to automatically recognize the logos and the brands of the commercial products in images, so users can perform the reverse search by images to find the products of the particular brands. Enterprises can also use this to do marketing for their products. In terms of face recognition, DeepFace [100] reaches the accuracy comparable to humans. New face recognition technologies driven by deep neural networks can be employed in a wide range of applications in security domains. For example, users can log into the computers by their faces. This has already been used in many PCs today (e.g., Microsoft Surface). Another application is that people can scan their faces to make payments or enter the authorized places instead of using sensors or badges. Autonomous driving [101] is another big area in which computer vision and deep learning can be applied. Even though the self-driving cars use multiple sensors to detect the physical objects around them, it is challenging to distinguish multiple objects with similar size, and similar shapes at the same distances. Hence, perceiving and understanding of various objects such as traffic signs [102–104], lights, other cars, and many other objects using deep learning is one of the core components for implementing autonomous driving. Robotics [105] is a complex system with vision as one of the core parts. Accurately recognizing the objects and the environments is a pre-step for robots to take logic actions. Material classification [32] is a research topic, in which machines can automatically categorize different materials of objects and perform auto recycling. In this way, robots operate differently when walking on different materials of the surfaces.

NLP is another domain in which many applications have been driven by deep neural networks. It mainly uses Recurrent Neural Networks (RNN) [106], [107] to capture the temporal information as the memory to build the language model. Both speech recognition and machine translation have recently achieved significant improvement in accuracy. Microsoft Research has developed a real-time speech translation system using deep neural networks [108], [109]. Combining RNN in the language model and CNN in the computer vision can develop advanced AI applications. For example, once the objects in the images are recognized and the relationship between the entities is analyzed by the language model, description of images could be automatically generated. Some

examples of machine-generated descriptions based on the images are "two pizzas are sitting on top of the oven" and "a group of young kids are playing frisbee on some grass field".

4.3 Architecture

The basic unit of a neural network is called a perceptron. A perceptron can be considered as a "gate", which takes several input values and produces a single output value. Graphically, a perceptron is as shown in Fig. 4.1. The function of a perceptron is:

$$output = \begin{cases} 0 & if \sum_j (w_j x_j) < threshold \\ 1 & if \sum_j (w_j x_j) \geq threshold \end{cases} \tag{4.1}$$

where w_j represents the weight for the jth neuron and x_j represents the value of the jth input neuron.

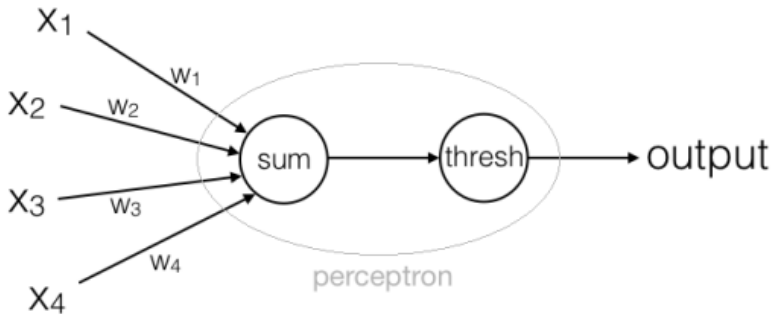

Figure 4.1. Example of a perceptron

A typical neural network consists of several layers with each layer containing multiple perceptrons. Fig. 4.2 shows a 4-layer neural network with 2 hidden layers, where the first layer contains 4 nodes and the second layer contains 3 nodes.

Each node inside the network is called a neuron. There are mainly three types of variables inside a neural network: the neuron values X, the weights W, and the biases b. Each neuron has a neural value associated with it. Each link between two nodes has a weight. Each neuron also has a bias except the neurons in the input layer. The activation function defined in Eq. (4.2) is used to compute a single output neuron value z

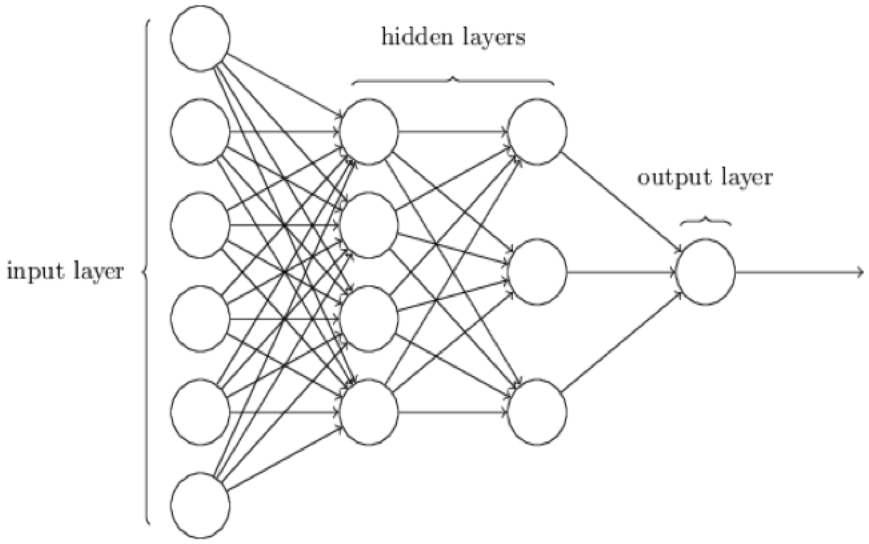

Figure 4.2. Example of a neural network with 2 hidden layers

(with bias b) based on the neuron values X from the input layer and their corresponding weights W.

$$Z = WX + b \tag{4.2}$$

Eq. (4.1) represents a step function, which maps from multiple inputs to a single output as shown in Fig. 4.3. However, the drawback is that the change is too abrupt. In other words, a nice property we want a perceptron to have in a neural network is that the small change in each input causes a small and gradual change in output.

Driven by this desired property, we choose to use the sigmoid function instead of the step function as the activation function to do the mapping. The sigmoid function is defined as Eq. (4.3)

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.3}$$

where z is the input parameter. The graph of the sigmoid function is shown in Fig. 4.4. As we can see, the shape of the sigmoid function is a smooth curve with input z ranging from negative infinity to positive infinity and output $\sigma(z)$ ranging between 0 and 1. The function is also monotonic. Hence, a small change in z causes a small change in σ.

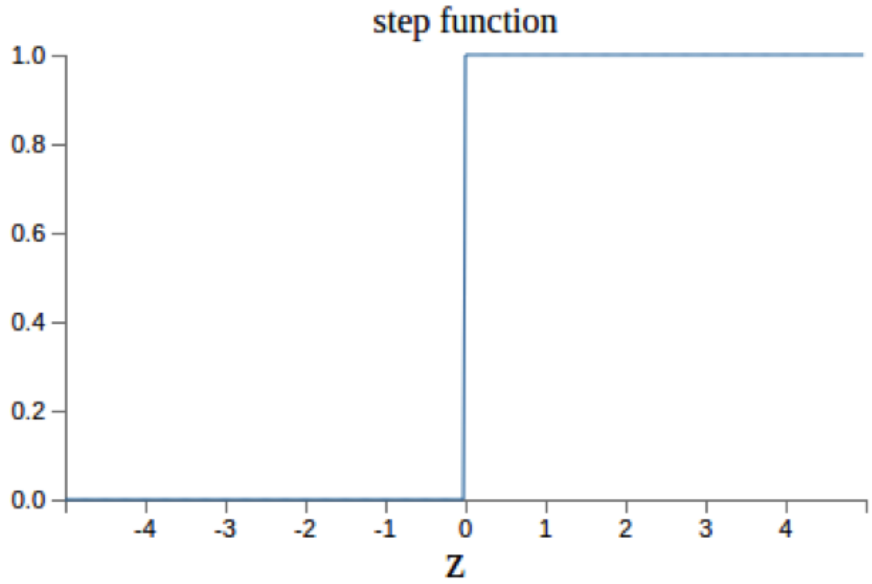

Figure 4.3. A step function

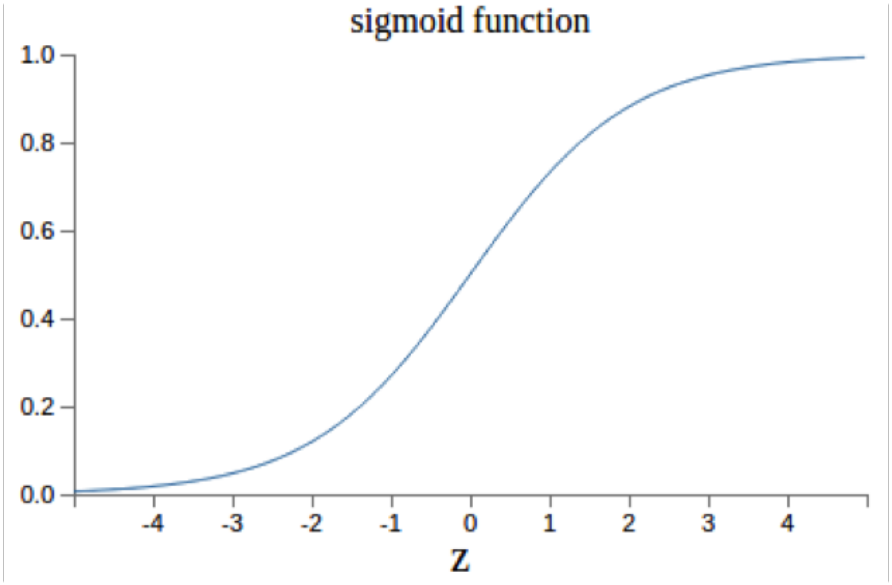

Figure 4.4. A sigmoid function

4.4 Feed-Forward and Back Propagation

The learning of a neural network is carried out through two types of update: feed-forward and back propagation. Feed-forward takes data from input layer and updates the neuron values layer by layer in the forward direction until it reaches the output layer. Suppose there is a total of n instances in a training set. The number of neurons in the

input layer is equal to the dimension of each data instance. At the beginning, all the weights and biases of the network are initialized in a random fashion. The neuron values in the input layers are fed in by the raw feature values of each training data instance. To compute each neuron value of the second layer (i.e., the first hidden layer), all the neuron values of the first layer (i.e., input layer) are multiplied with their corresponding weights to obtain a weighted sum by the activation function defined by Eq. (4.2). This weighted sum is fed into a non-linear transformation by using the activation function defined in Eq. (4.3). After each neuron value in the second layer is computed, we can use them as the input to similarly compute the neuron values in the third layer. In this way, we keep computing the neuron values in each layer in the forward direction until we obtain the neuron values in the output layer. This process is called feed-forward.

The process of updating the weights and the biases from the output layer to the input layer is called back propagation. To perform the back propagation, we first define a loss function (i.e., objective function) to measure the overall differences between the predicted labels from the network and the true labels of the training data. There are many different types of loss functions. For illustration purpose, we use a quadratic loss function as an example. The quadratic loss function is defined as follows:

$$C(w, b) = \frac{1}{2n} \sum_{x} \|y(x) - a^2\| \tag{4.4}$$

where C is the loss, w is the weight, b represents the bias, n is the total number of the training instances, $y(x)$ is the true label of the training instance and $a = \sigma(z)$ is computed by Eq. (4.3) and contain the predicted label from the network.

The goal of training the network is to minimize the cost function. We use back propagation to achieve this minimization. Theoretically, this minimization problem can be solved analytically. However, it becomes impractical when the feature dimension of the input data X is high. Therefore, an approximation technique is applied to solve it. The most common solver for this problem is gradient descent. Suppose that the parameters of the loss function are two dimensions, the loss function is in a 3-d space. Gradient descent treats the problem as finding the minimum point for a ball on a plane from some initial points by rolling the ball a little in a certain direction each time by

gravity. If the loss function is convex, each parameter is updated by subtracting the product of its partial derivative from a small weight using Eq. (4.5) and (4.6), where η is called the learning rate, and k and l represent the index for iteration. The learning rate decides how big the ball is moving in each iteration.

$$w_k \to w_k - \eta \frac{\partial C}{\partial w_k} \tag{4.5}$$

$$b_l \to b_l - \eta \frac{\partial C}{\partial b_l} \tag{4.6}$$

In practice of training a neural network, we need to compute the gradient vector for each of the large number of training instances, which leads to high computational cost. Therefore, a variant of gradient descent using sampling and approximation techniques is usually used to reduce the computational cost. It is called stochastic gradient descent. Its main idea is to take a sample from the training set each time to compute the gradient and use the average of the gradients in a small set of samples to approximate the update of the weight and the bias. Each time a sample called mini-batch is drawn from all the training instances without replacement. We keep taking samples until all the training instances are used up. Suppose the total number of mini-batches is m, the updates for stochastic gradient descent is computed by Eq. (4.7) and (4.8).

$$w_k \to w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k} \tag{4.7}$$

$$b_l \to b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l} \tag{4.8}$$

In summary, when training a neural network through feed-forward and back propagation, each input training instance X is directly passed into the input layer and goes through feed-forward steps. The output from the previous layer is the input to the next layer. During the transformation between two layers, the summation of the weighted products followed by a nonlinear transformation (e.g., sigmoid function) is applied. During the training, the entire dataset is split into three sets: a training set, a validation set, and a testing set. The training set is used to learn the weights and the biases by

iteratively feeding instances from the training data via mini-batches. After getting the output labels from the network by going forward, the predicted labels in the output layer are computed with the true labels of the training instances. Based on a pre-defined objective function, gradient descent is applied to update the weights and biases. After all m mini-batches are used once, one epoch is completed. For each iteration in the training process, the validation set is used to compute the classification error as the validation error. There are three options for the termination criterion in the training: First, a fixed number of epochs is pre-defined and the training is terminated once this fixed number of epochs is reached. Second, the training is stopped if the learning process converges, i.e., the change in loss between two consecutive epochs is less than a pre-defined small threshold. Third, the training is terminated if the change in validation errors between two consecutive epochs is less than a pre-defined small value.

Due to the non-linear transformation nature between every two layers, theoretically, this whole network can approximate any function if the network is large enough. This is the main reason deep network is able to achieve better performance than other classifiers in many problems.

4.5 Optimization

After defining the basic architectures of neural networks and their solvers, we discuss different cost functions and optimization strategies that can be used for training neural networks.

Recall the loss function defined in Eq. (4.4) is called the quadratic loss function. There are some drawbacks for the quadratic loss function. For simplicity, suppose that there is only one training instance, the Eq. (4.4) becomes

$$C = \frac{(y-a)^2}{2} \tag{4.9}$$

where $a = \sigma(z)$ where $z = wx + b$.

To compute the updates, we take partial derivative of cost C with respect to weight w and bias b, suppose $x = 1$ and the desired output $y = 0$, then we have:

$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x = a\sigma'(z) \tag{4.10}$$

$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z)x = a\sigma'(z) \tag{4.11}$$

It shows that the updates mainly depend on $\sigma'(z)$. However, based on the shape of the curve as shown in Fig. 4.4, the derivative of $\sigma(z)$ becomes smaller when $\sigma(z)$ approaches 0 or 1 (i.e., curve becomes smoother). Therefore, the learning is slow.

To speed up the learning, various techniques are proposed. One technique is to define different loss functions. There are two conditions that need to be met when defining a loss function. First, the cost needs to be greater than or equal to 0 (i.e., $C \geq 0$). Second, the cost is equal to 0 when the predicted output label from the neural network is equal to the true label for data points (i.e., $C = 0$ when $a = y$). Cross-entropy is one of the cost functions that can be used to speed up the learning, which is defined as in Eq. (4.12)

$$C = -\frac{1}{n}\sum_{x}[yln(a) + (1 - y)(1 - a)] \tag{4.12}$$

where $a = \sigma(z)$ and $z = \sum_j w_j x_j + b$

The cross-entropy loss function satisfies the two conditions. In addition, it has a nice property to speed up learning. To see that, by taking partial derivative of cost C with respect to weight w and bias b, we have

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_{x} x_j(\sigma(z) - y) \tag{4.13}$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_{x}(\sigma(z) - y) \tag{4.14}$$

It is clear that the learning speed depends on $\sigma(z) - y$, which is the output error. Consequently, it learns fast for big errors and slow for small errors. The cross-entropy is almost always better than the quadratic cost unless the relationship between input and output is linear since quadratic cost is able to model the linear relationship.

There are many other techniques that can improve the learning of neural networks. For example, softmax is a function that is commonly used to connect the output layer. It is defined in Eq. (4.15) and (4.16).

$$z_j^L = \sum_k w_{jk}^L a_k^{L-1} + b_j^L \tag{4.15}$$

where w_{jk}^L represents the weight connecting jth neuron in Lth layer and kth neuron in $L-1$th layer, a_k^{L-1} represents the neuron values in kth neuron of $L-1$th layer, and b_j^L stands for the bias on jth neuron of Lth layer, and

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \tag{4.16}$$

Since $\sum_j a_j^L$ equals to 1, the softmax can be interpreted as the probability distribution of each class in the output layer. The weighted sum in Eq. (4.15) for the softmax function is the same as the one in the sigmoid function. The difference is the activation function defined in Eq. (4.16). Hence, the sigmoid function cannot be interpreted as the probability distribution of each class whereas the softmax function can be interpreted as such. To avoid the slow learning issue for the softmax function, we use natural log to define the cost function so the learning speed depends on the error.

Overfitting is a general problem existing in machine learning, with no exception for neural networks. It makes the model capture many details from the noisy data and not have enough generalization power. It typically performs well on the training set but poorly on the testing set. Regularization can be used to reduce overfitting. For cross-entropy defined in Eq. (4.12), we can add an extra term $\frac{\lambda}{2n}\sum_w w^2$, where λ is a fixed ratio as the penalty to the original cross-entropy function. The regularized cross-entropy encourages small weights so noise in the input data does not change weights too much. In other words, it is less likely that the model is affected by the noise of the data. Similarly, the same extra term can be added to the quadratic function as well. This type of penalty is called L1 regularizer. Another type of regularizer defined by $\frac{\lambda}{2n}\sum_w |w|$ is called L2 regularizer. After taking the partial derivative of the regularized cross-entropy, L1 updates the weights by subtracting a constant and L2 updates the weights by subtracting a fixed ratio of the weight. Therefore, L1 regularization has

much less reduction than L2 regularization if the weight itself is large. In summary, regularized loss functions tend to learn a simpler model in general. However, it does not always perform better than the un-regularized loss. Large data is required for the regularized model to have good performance.

Unlike L1 and L2 regularization, which change the cost function, another technique called dropout reduces overfitting by changing the structure of the network. An observation is that overfitting could be reduced by training multiple neural networks followed by computing the average of the output. Driven by this observation, dropout reduces overfitting by randomly dropping a half of neurons at each layer and using the other half of the network to train for each iteration.

There are several other factors that can affect the learning of the neural networks. First, the way to initialize weights may affect the learning behaviors. If the weights are initialized to be too large, most z values defined by in Eq. (4.15) are either much greater than 1 or much less than -1. Based on the shape of the sigmoid function, the gradients of $\sigma(z)$ are small for these values. This can cause the learning to be slow. A good way of initializing weights is to generate weights from Gaussian distribution with mean being equal to 0 and standard deviation being equal to $1/\sqrt{n_{in}}$, where n_{in} represents the number of neurons in the input layer. Second, the choice of the activation function plays an important role in the learning process. Here we focus on two common activation functions, namely, the sigmoid function and the softmax function. The sigmoid function can cause the vanishing gradient problem. The vanishing gradient problem is defined as the unstable learning caused by the fact that updates of the gradient of the next layer is accumulated multiple of the previous layers. Rectified linear function defined by $y = max(0, w \times x + b)$ is used to handle the vanishing gradient problem. Since the sigmoid function has the range from 0 to 1 and the rectified linear unit function ranges from 0 to positive infinity, the sigmoid function can describe the probabilities and the rectified unit function can describe real numbers. In addition, the gradients decrease as input data X feed-forward through the layers for the sigmoid function. However, it is not the case for the rectified unit function. Hence, the rectified unit function does not have the vanishing point problem.

Other hyper-parameters that affect the learning of neural networks include the

number of epochs to run, choices of the mini-batch size, the learning rate, and the regularization parameter. How to systematically tune these parameters to optimize the learning of neural networks is still an ongoing research topic. There are some rules of thumb. First, we can fix a small number of epochs and mini-batch size and start tuning the learning rate from a fixed number (e.g., 0.1). If the cost starts increasing, a small adjustment on the learning rate to a smaller scale can be performed before continuing the training. The regularization parameter can be tuned in the same way as the learning rate. In term of the mini-batch size, if it is too small, we are not taking full advantage of the parallel computations of GPU, and large variations exist in different mini-batches. If it is too large, there will be infrequent updates of weights and biases. The good news is that the mini-batch size is independent of the choice of the other parameters, so we don't need to change it once it is fixed. Some variations of the gradient descent are Hessian and Momentum-based descent.

4.6 Convolution Neural Networks

All traditional networks are comprised of some layers with perceptrons, and each layer is one dimension. For some classification tasks in image domains, each gray-scale image is essentially a 2-dimensional matrix with each pixel represented by an element in the matrix. Furthermore, the spatial relationship between pixels has correlations in terms of the semantic meaning of the images. Therefore, flattening the image into a vector and feeding it into a 1-d input layer in neural networks lose the original spatial relationships in the image and cannot achieve great performance. A different type of network called Convolution Neural Network (CNN) is developed to do the learning for the problems in the image domain.

A CNN is similar to a traditional multi-layer perceptron network except that each layer is a 2-d array of neurons. Fig. 4.5 shows that convolution and pooling operations are used to connect each layer to the next layer. For convolution, a per-defined small template (e.g. 5x5) operates on each input layer by convolution in the scanning fashion to generate feature maps. Pooling means subsampling feature maps to a smaller dimension. For the weights between two layers, all the weights are shared by connecting template neurons in the input layer to different neurons in the next layer. It can signifi-

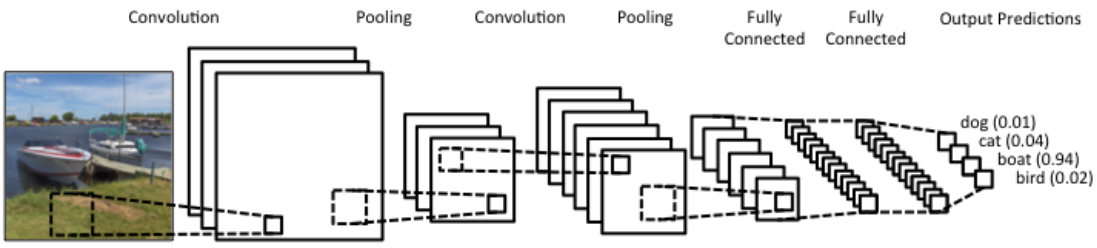

Figure 4.5. Convolution neural network example

cantly reduce the total number of parameters in the network to make training practical. The remaining steps for training are the same as the techniques that are used in the traditional multi-layer perceptron networks.

4.7 Object Proposal Classification

In our task of object proposal classification, we employ a two-stage deep convolution net approach: pre-train and fine-tune.

In the pre-train stage, large-scale out-of-domain labeled data could be used to train the network and learn the meaningful features, as long as the categories in the domain-specific dataset is a subset of the categories in the large-scale out-of-domain dataset. In the fine-tune stage, the pre-trained weights and bias are used to initialize training, replace the number of neurons in the output layer is replaced by the number of categories in the domain-specific dataset, and domain-specific data are used to train (i.e., fine tune) the network.

Specifically, we use ImageNet data [110] in the pre-train step since it is the largest image dataset with the labeled objects. We use the open-source deep learning framework Caffe [111] with AlexNet [85], which is an eight-layer convolution network, in the pre-training to learn meaningful feature representation from ImageNet data. We then use labeled objects in the domain-specific dataset from a video or videos to fine-tune the parameters (e.g., weights and biases). After the training, we will classify each of representative proposals into background or a specific object category. By assigning the same categorical label to the rest of object proposals in the same proposal track, we can label all object proposals in all individual frames throughout the video.

CHAPTER 5

EVALUATION OF THE PROPOSED SYSTEM

In this chapter, combining all the modules from the previous chapters, we illustrate the entire system in details by describing the input and output of each module in a workflow. Then we describe the experiments conducted on the entire system for object recognition in video, followed by presenting preliminary experimental results and refined experimental results. In preliminary experiments, we use the hard-coded threshold for shot boundary detection and only select a few representative frames from each shot to pass down to the remaining modules. In the refined experiments, we use an adaptive threshold for shot boundary detection and all labeled frames for experiments. For refined experiments, we have four settings which can separately evaluate the effect of adding each individual proposed module (i.e., hierarchical model and temporal objectness) to the whole system.

5.1 System Workflow

The architecture of the entire proposed system is shown in Fig. 1.1. Since a video includes multiple shots with relatively small visual content variations within each shot and large variations between the shots, we firstly apply the shot boundary detection algorithm to find the shot boundaries of the video. In other words, the video is divided into multiple shots with each shot containing a sequence of consecutive frames. The last frame of each shot is the predecessor of the first frame of the next shot.

Multiple shots are generated after applying the shot boundary detection algorithm on a video. For each shot, we apply the key frame extraction method to find the key frames. Key frames refer to a few frames in the shot that best represent the entire shot. In each shot, the number of resultant key frames depends on a parameter in the key frames extraction algorithm, which is defined as the ratio of the number of the key frames to the total number of frames in the shot. The key frames selected in each shot,

form a set of M key frames in the entire video. It should be noted that the first frame of the shot is always a key frame. We would have M frame sequences with the ith sequence starting with the ith key frame and ending with the frame before the $(i+1)$th key frame. In this way, each frame sequence between two selected key frames has little content variation.

Once the key frames in each shot are obtained, we apply the object proposal generation method on each of these key frames to locate the regions, which are likely to contain objects. We use bounding boxes on the images to represent the corresponding object proposals. Specifically, we apply the tree-based hierarchical model on a training set of images with labeled objects, which belong to one of the object categories of the video, to learn the characteristics of each object and generate top n object proposals on each key frame.

Since each of the M key frame sequences starts with a key frame containing up to n generated object proposals, we use the generated object proposals as the initial bounding boxes and apply the SPOT tracker together with the temporal objectness algorithm to simultaneously track the object proposals and re-rank the objectness of these object proposals during tracking for each key frame sequence. Finally, we have up to $M \times n$ object proposal tracks (i.e., each track refers to a sequence of tracked object proposals across consecutive frames). We remove object proposals with low temporal objectness for each of M sequences since low temporal objectness indicates low possibility to contain objects.

In the last step, the goal is to classify each object track into one of object categories. Due to the very small variations within each key frame sequence from the shot boundary detection and the key frame extraction, the object category usually stays the same within each object proposal track. Hence, after classifying object proposals of the first key frame in each key frame sequence, all object proposals in this object proposal track are assigned to the same classified label in the first key frame. The two-stage deep CNN is applied on a large labeled dataset, containing the categories of objects in testing videos for training.

Finally, we have the object locations, which are denoted by bounding boxes and category labels for all bounding boxes for every frame in the entire video.

5.2 Preliminary Experimental Results

We use the YouTube-Objects V2.2 dataset as described in chapter 3.1.3 to run the experiments.

In the shot boundary detection step, we detect shot boundaries by comparing the differences of neighboring frames with a predefined threshold value. We set the threshold to be 20 in the shot boundary detection method to locate all shots for each video. As a result, we divide these 155 videos into 6392 shots. For each shot, we always set the first frame as the key frame and apply dynamic programming to select the rest of the key frames. The ratio of the number of key frames to the total number of frames in each shot is empirically set to be 0.02. As a result, we generate a total of 15744 key frames from 6392 shots. We select 7380 frames out of 15744 key frames that contain annotations for bounding boxes as the testing set. We directly use images in the training set of the Youtube-Objects V2.2 database as training images. Since 101 images in the training set of Youtube-Objects V2.2 are also chosen as the testing key frames, we exclude these 101 images from the training set and use the remaining 4205 images in the training set to train the hierarchical model on the bounding boxes obtained by EdgeBox [58] to learn object proposals. We select up to 50 object proposals in each train image and generate top 20 object proposals in each key frame (if the number of detected object proposals is less than 20, we simply select all of the object proposals). Then we use top 20 proposals in each testing key frame (i.e., the first frame of each track) and apply the SPOT tracker [83] on up to 7380×20 object proposals to generate proposed track and compute the temporal objectness for all proposals on each track. Using the computed temporal objectness, we filter out the object proposals with temporal objectness less than two standard deviations from the mean of each track. In this way, we keep the "high quality" proposals based on the temporal objectness.

The last step is to classify each proposal into one of 10 object categories. We need the images that have the ground truth labels to evaluate the performance of the system. Hence, from the object proposals generated so far, we select those proposals that have ground truth object category labels from annotated images in Youtube-Objects V2.2. This includes a total of 9558 object proposals as the testing set. For training, we pre-train CNN on 1 million images from ImageNet with more than 1000 categories

(including 10 categories of the Youtube-Objects V2.2 dataset) to get the initial model (i.e., weights and biases with discriminative features). We then fine-tune the model by replacing the output layer with 11 neurons corresponding to 11 classes (i.e., 10 object categories and 1 background category) and replacing the training data with the data in domain-specific dataset. This domain-specific dataset is generated by the following two steps: 1) excluding the key frames that are from the annotated training set in the YouTube-objects V2.2 dataset; 2) applying object proposal generation on the remaining frames and compare the generated proposals with the ground truth annotations to select positive and negative proposals. Specifically, we label all proposals with IoU higher than 0.7 as positive proposals and assign them the same label as the ground truth label of one of possible 10 classes. For proposals with IoU lower than 0.7, we label them as negative proposals (i.e., the background proposals). Since there are significantly more negative proposals than positive proposals, we limit the number of the negative proposals to be 5 per image by sub-sampling. As a result, we have a total of 24824 labeled proposals (12662 background proposals + 12162 object proposals) in our training data for classification.

The entire system is implemented using Java, Python and Matlab. For module of learning object proposals, the experiment runs in a parallel fashion to process a large number of images on Amazon Web Service (AWS) EC2 m4.10xlarge linux instance with 40 virtual CPUs and 160G memory; For module of object proposal classification with deep neural networks, the experiment runs using AWS EC2 g2.8xlarge linux instance with 4 NVIDIA G520 GPUs (each with 1536 CUDA cores) and 32 virtual CPUs and 60G memory for about 24 hours.

We use two common criteria, interpolated average precision per category and mean Average Precision overall, to evaluate the proposed system. Table 5.1 compares the average precision per category and mean average precision overall of the proposed system and five state-of-the-art systems, namely, RCNN [49], Deformable Part Model (DPM) [55], fine-tune Selective Search [112], Fine-tune EdgeBox [58] and fine-tine Videos Through label Propagation (VOP) [113] on the test set of YouTube-Objects V2.2. For categories of bird and car, during the module of learning object proposal tracks using temporal objectness, we remove all object proposals of bird and car based on the temporal objectness. This is possibly due to the relatively skewed distribution of temporal objectness

Table 5.1. Comparison of the proposed method and the state-of-the-art: average precision for each category and mean average precision on the YouTube-Objects V2.2 testing set

	R-CNN	DPM	Fine-tune SS	Fine-tune EB	Fine-tune VOP	Proposed
plane	14.10	28.42	25.57	26.52	29.77	**49.99**
bird	24.20	**48.14**	27.27	27.27	28.82	N/A
boat	16.90	25.50	27.52	33.69	35.34	**37.18**
car	27.90	**48.99**	35.18	36.00	41.00	N/A
cat	17.90	1.69	25.02	27.05	33.7	**38.04**
cow	28.60	19.24	43.01	44.76	**57.56**	7.15
dog	12.20	15.84	24.05	27.07	34.42	**47.86**
horse	29.40	35.10	41.84	44.82	**54.52**	36.52
mbike	21.30	31.61	26.70	27.07	29.77	**46.13**
train	13.2 0	39.58	20.48	24.93	29.23	**45.23**
mAP	20.570	29.411	29.664	31.918	37.413	**38.5125**

for car and bird optical flows and the threshold for temporal objectness (i.e., mean minus two standard deviation). For the remaining eight categories, we can see that the proposed system achieves the highest precision in six categories and outperforms all state-of-the-art methods in terms of mAP. Since the object proposals of bird and car categories are all filtered out by the temporal objectness module due to the threshold used in temporal objectness, the AP for bird and car categories are not available. We improve the way to remove the object proposals with low temporal objectness in the refined experiments so the final AP for every object category is available.

5.3 Refined Experiments

In the proposed system, there are two major novel components: tree-based hierarchical model and temporal objectness. To evaluate the effectiveness of each proposed component, we design and conduct the experiments with different settings. In the following section, we describe each experiment in detail.

Overall, we design four settings of experiments to evaluate the impact of each proposed component.

The first experiment aims to test the performance of the system without the hierarchical model or temporal objectness. In other words, for stable sequence generation, we apply shot boundary detection followed by key frame extraction to obtain stable key frame sequences. In object proposal learning, we employ EdgeBox algorithm to

detect object proposals on the key frames that have ground truth labels. Then we apply SPOT tracker and CNN for objects classification. We call the system without these two components as the base system.

The second experiment aims to evaluate the impact of the proposed hierarchical model. We use the same setting for stable sequence generation to obtain stable key frame sequences. In object proposal learning, we use the hierarchical model to re-rank the learned object proposals generated by the EdgeBox algorithm. Last, we apply SPOT tracker and CNN for objects classification. We call this system as the base system plus the hierarchical model.

The third experiment aims to evaluate the effectiveness of the proposed temporal objectness. We generate the stable sequences in the same manner as the base system. In object proposal learning, we generate object proposals using EdgeBox, apply SPOT tracker to track object proposals, and use the proposed temporal objectness to re-rank the object proposals based on objectness. Last, we classify the object proposals by CNN. We call this system as the base system plus temporal objectness model.

The fourth experiment aims to evaluate the performance of the entire proposed system, which includes the two novel components. Specifically, after generating stable frame sequences, we apply EdgeBox followed by the hierarchical model to learn object proposals. During tracking using SPOT tracker, we also further re-rank the object proposals by computed temporal objectness. Finally, we employ CNN to classify the object proposals. We call this system as the proposed system.

5.3.1 Base System

For the base system, we first apply shot boundary detection for both hard cuts and gradual changes for each video. It outputs the divided shots, which are indicated by starting and ending frame numbers for each shot. The number of resultant shots for each category is summarized in Table 5.2. After shot boundary detection, we perform key frame extraction for each shot. One important parameter in key frame extraction is κ, which represents the proportion of the number of resultant key frames to the total number of frames. To test and tune this parameter, we repeat the key frame extraction experiment three times with κ values of 0.02, 0.04, and 0.06. As a result, we generate

18015, 31497, and 45157 key frames, respectively. This completes the module of stable sequence generation. Second, we apply EdgeBox object detector to learn top 50 object proposals on each key frame. To evaluate the detection rate of EdgeBox on these key frames, we select the key frames that have the ground truth labels available as described in section 3.1.2.3. For each of these images, we compute and plot the detection rate with the number of proposals increasing from 1 to 50. We use the standard IoU as the measure to detect whether an object proposal bounding box correctly detects the true object. We set the values of IoU as 0.5 0.7, and 0.9. For each value of IoUs, we compare the detection curves corresponding to three κ values.

Table 5.2. The number of videos and detected shots for each class

aeroplane: 13 videos, 460 shots	bird: 16 videos, 204 shots
boat: 17 videos, 588 shots	cat: 21 videos, 325 shots
car: 9 videos, 168 shots	cow: 11 videos, 228 shots
motorbike: 14 videos, 428 shots	dog: 24 videos, 395 shots
train: 15 videos, 689 shots	horse: 15 videos, 351 shots

Fig 5.1, 5.2 and 5.3 present the detection rates for the different κ values at IoU values of 0.5, 0.7, and 0.9, respectively. Setting κ value of 0.04 achieves the highest detection rate across all three IoU values. Hence, we select 0.04 as the optimal to build the base system and this value is used in the variant other systems, too. Using the object proposals learned on each key frames, we initialize the SPOT tracker and start tracking objects for each stable frame sequence. Out of all the tracked object patch tubes, we select the tubes with at least one frame containing the ground truth label. There are 1781 frames in tracked frames with at least one frame containing ground truth. Within each tube, we assign all frames the same label as the frames with the known object category based on the ground truth. This completes the module of object proposals learning.

Last, we apply CNN to classify the learned object proposals into 11 categories (10 object classes + 1 background class). For the training set, we simply employ EdgeBox object detector on all the frames that have the ground-truth labels, which are in the original training set of the YouTube-Objects V2.2 database. For the testing set, we

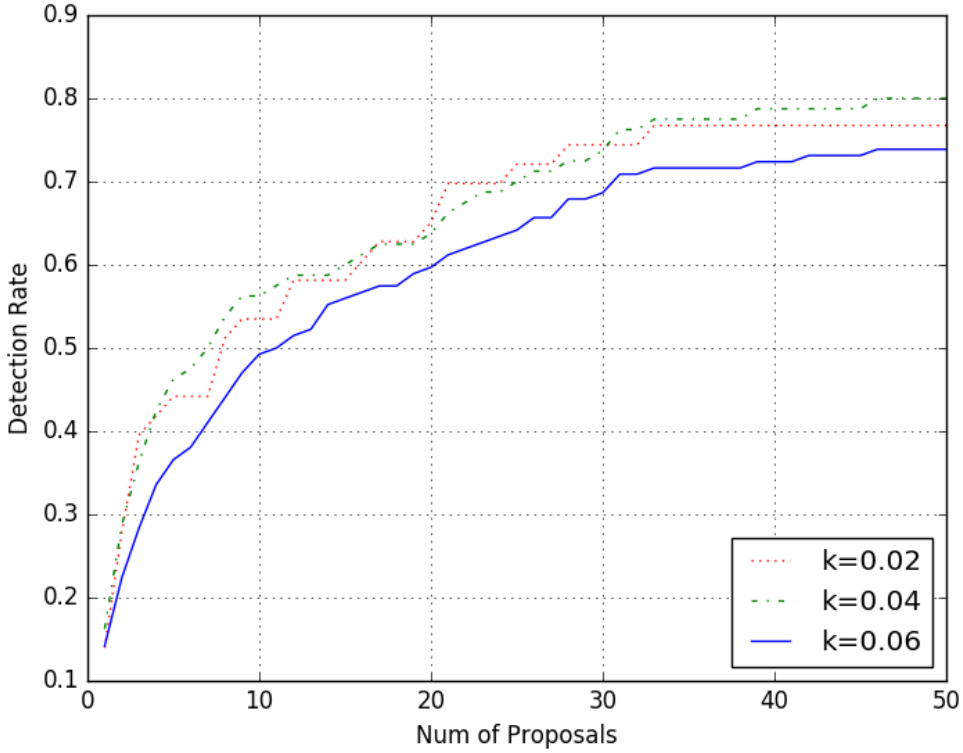

Figure 5.1. Detection rate for IoU = 0.5

exclude the key frames that are also in the training set. For the remaining key frames, we use the top 20 tracked object proposals that have ground-truth labels in the testing set of YouTube-Objects V2.2 dataset. As for CNN classification, we have 34848 images for training and 34895 images for testing. Surprisingly, all object proposals are classified as background. It turns out that most object proposals in the testing set for classification are background. It is clear that EdgeBox object detector is not able to generate good object proposals.

5.3.2 Base System Plus the Hierarchical Model

To build the first variant system, we incorporate the proposed hierarchical model into the base system. First, we directly use the resultant stable frame sequences generated from the base system. Since three κ values (i.e., 0.02, 0.04 and 0.06) for key frame extraction are tested in the base system and the κ value of 0.04 achieves the best performance, we use κ value of 0.04 throughout to test the remaining experiments. In

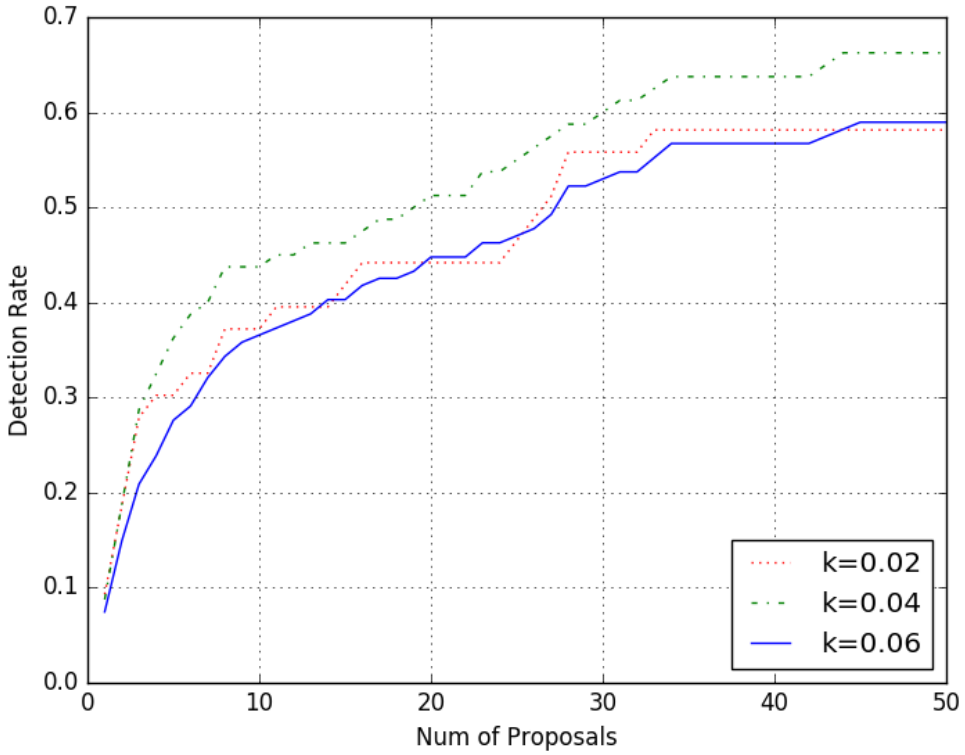

Figure 5.2. Detection rate for IoU = 0.7

other words, we use the resultant stable frame sequences from the base system (based on $\kappa = 0.04$ for key frame extraction) as the input for learning object proposals.

Second, we apply EdgeBox object detector to learn top 50 object proposals on each key frame. This step is also the same as the base system. So, we directly use the detected object proposals from the base system.

Third, we apply the hierarchical model to re-rank the object proposals from EdgeBox detector as described in section 3.1.3.2. We want to test the performance for IoU = 0.7. Since the weight w for balancing the visual distance and tree distance with the value 0.6 yields the best performance under IoU = 0.7, we use the object proposals generated from the hierarchical model with b of 0.6.

After applying the hierarchical model, we only choose the key frames that also have ground truth labels to start tracking. There are 65 frames of this type. Then we use each of these frames and apply the SPOT tracker with detected object proposals to start tracking. After tracking, up to 65×50 tracked patch tubes are generated. We

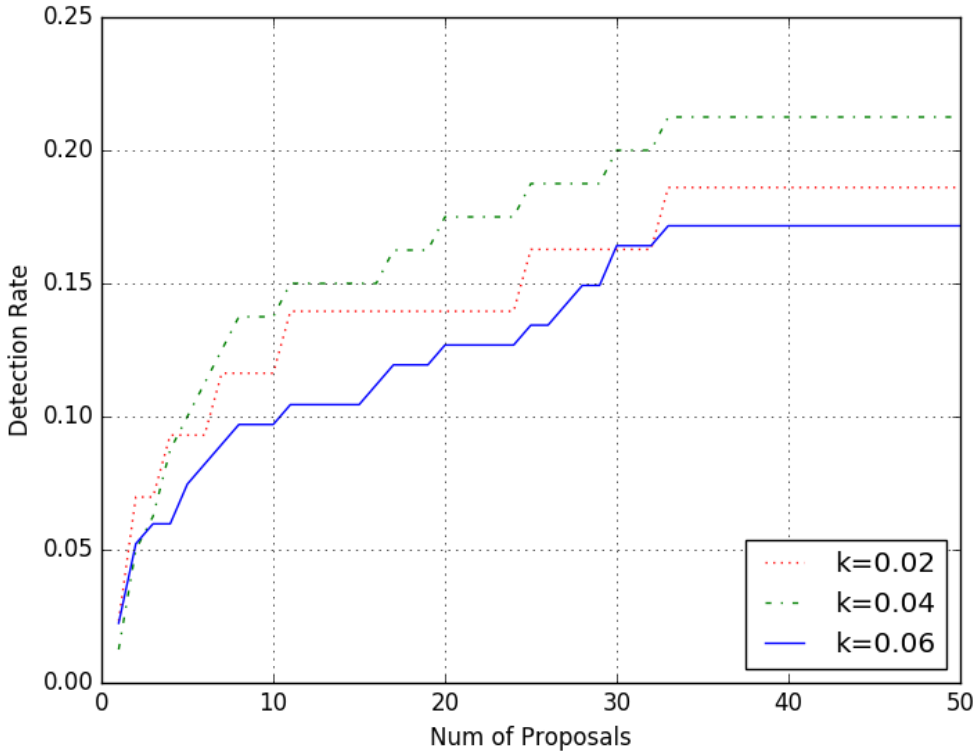

Figure 5.3. Detection rate for IoU = 0.9

use all frames with ground truth for each of these tubes as the testing set for CNN classification.

Last, we apply CNN to classify the object proposals. The training set is the same as the training set we used in the base system. The testing set includes 9720 object proposals.

The classification results are shown in Table 5.3. We can see that the average precision for most categories are still zeros. However, we get a little bit improvement comparing with the base system since we get some correct results for car, horse, and train categories. The final mAP is 3.92, which is still quite low compared with the state-of-the-art methods.

5.3.3 Base System Plus the Temporal Objectness Model

We incorporate the proposed temporal objectness model into the base system to build the second variant system. Since the stable frame sequence generation module

Table 5.3. Comparison of the BH system and the state-of-the-art: average precision of the state-of-the-art and base+hierarchical system for each category and mean average precision on the YouTube-Objects V2.2 testing set

	R-CNN	DPM	Fine-tune SS	Fine-tune EB	Fine-tune VOP	Base+Hierarchical
plane	14.10	28.42	25.57	26.52	**29.77**	0.00
bird	24.20	**48.14**	27.27	27.27	28.82	0.00
boat	16.90	25.50	27.52	33.69	**35.34**	0.00
car	27.90	**48.99**	35.18	36.00	41.00	15.66
cat	17.90	1.69	25.02	27.05	**33.7**	0.00
cow	28.60	19.24	43.01	44.76	**57.56**	0.00
dog	12.20	15.84	24.05	27.07	**34.42**	0.00
horse	29.40	35.10	41.84	44.82	**54.52**	1.33
mbike	21.30	**31.61**	26.70	27.07	29.77	0.00
train	13.2 0	**39.58**	20.48	24.93	29.23	22.25
mAP	20.570	29.411	29.664	31.918	**37.413**	3.92

is exactly the same with the base system, we use the resultant stable frame sequences from the experiment in the base system. This time, after applying EdgeBox detector to generate object proposals on the key frames, we directly apply the SPOT tracker to track the top 50 object proposals in each sequence. Then we compute the temporal objectness for each object patch tube and re-rank the object proposals based on their temporal objectness. After re-ranking, we use an empirically-studied threshold to remove the object proposals with low temporal objectness. For the remaining object proposals, we feed them into the CNN for classification and compute the AP for each category and mAP.

Under this setting, there are two parameters we would like to test. The first parameter is the value of IoU when we use detected object proposals based on the ground-truth labels to start tracking. We test 3 values of IoU, which are 0.5, 0.7 and 0.9. The other parameter we test is the threshold to use for removing the object proposals with low objectness. We test four values for this parameter and they are: mean minus one standard deviation, mean minus two standard deviations, median minus one standard deviation, and median minus two standard deviations.

When initializing trackers using the object proposals, we only the select first frame with at least two positive object proposals based on the ground truth to start tracking. This is due to the constraint imposed by the SPOT tracker (i.e., require the structures

from multiple objects). When using IoU value of 0.5, we have 51 tracked patch tubes satisfying the requirement; when using IoU value of 0.7, we have 34 qualified tracked patch tubes; when using IoU value of 0.9, no frames that contain two positive proposals, and we do not perform tracking. The CNN classification is finally applied to label each proposal.

In the Table 5.4, we present the AP for each category and the final mAP for IoU =0.7 under four different temporal thresholds in CNN classification. Overall, the results are much better than the results in the base system. Comparing the four thresholds, the APs for all 10 categories are close to each other. The four mAPs are also very close. The threshold with median minus one standard deviation has the highest mAP. Therefore, we will use this threshold in the final system.

Table 5.4. Performance for the BT system under four settings: average precision for each category and mean average precision for four settings of temporal objectness for base+temporal on the YouTube-Objects V2.2 testing set

	mean-1std	mean-2std	median-1std	median-2std
plane	0.00	0.00	0.00	0.00
bird	41.11	40.90	41.09	40.90
boat	98.69	98.69	98.69	98.69
car	27.74	24.70	24.90	24.70
cat	0.00	0.00	0.00	0.00
cow	0.00	0.00	0.00	0.00
dog	60.32	60.30	60.34	60.32
horse	78.70	74.75	78.70	74.75
mbike	32.14	50.63	50.00	41.76
train	0.00	0.00	0.00	0.00
mAP	33.90	37.84	**37.93**	36.66

5.3.4 Final System

We incorporate the proposed modules into the base system to test the performance of the entire system. First, we still apply stable frame sequence generation module. Second, we apply EdgeBox detector to learn the object proposals on the first frame of each stable sequence. Third, we apply the proposed hierarchical model to refine the object proposals. Then we apply SPOT tracker and compute the proposed temporal objectness to remove the object patch tubes with low objectness. Finally, we apply

CNN to classify all object proposals to one of 11 categories (10 object categories and 1 background category).

For the experiments, the setting of generating the stable frame sequences and learning object proposals from the hierarchical model, are exactly the same as the settings of the first variant system (i.e., Base Plus the Hierarchical model). We choose the optimal values for two parameters based on the experimental results obtained for the first variant system. We use IoU = 0.7 in testing the final system since it is the most commonly used value in all related work. For the weight b of balancing the visual distance tree distance in the hierarchical model, we use 0.6 since it achieves the best performance for IoU of 0.7. We have 1128 frames with ground truth labels and we use these detected object proposals to initialize the SPOT tracker to start tracking in each stable frame sequence and remove the object proposals with low temporal objectness. The threshold of removing the low temporal objectness is the median minus one standard deviation since it achieves the best performance in previous setting. Since we want to include as many frames as possible for the remaining experiments, we keep all the tracked frames, which have at least one frame with ground truth in each sequence. As a result, we have 558 tracked frames. After removing the frames that have object proposals with low temporal objectness, we end up with having 464 frames in the testing set for CNN classification. Table 5.5 shows the summary, which includes the number of detected shots, the number of the selected key frames, the number of frames with annotations testing set from YouTube-Objects V2.2, and the number of frames used in testing set of CNN classification.

The last module is to apply CNN classification on the proposals from these 464 frames. We specifically test the effect of the number of top proposals resulted from the temporal objectness. Suppose that k represents the number of top object proposals we employed for classification. We test three values of k, which are 30, 40, and 50, in classification accuracy.

Table 5.6 shows the average precision for each category, and mAP for two variant systems and the final system with different values of k's. We can see from the table that the worst performance is from the first variant system (i.e., the base plus the hierarchical model). The variant system with the base plus the temporal model achieves better

,

Table 5.5. Dataset statistics in the final system: the number of shots, number of key frames, number of frames in annotated testing set from YouTube-Objects V2.2, and number of frames in testing set of CNN classification using the threshold of one standard deviation from median in the final system

	shots	key frames	annotated testing set	testing set in CNN
plane	460	3095	180	29
bird	204	1352	162	51
boat	588	4683	234	31
car	168	1055	606	58
cat	325	2325	165	26
cow	228	1597	140	90
dog	395	3221	164	36
horse	351	2665	181	45
mbike	428	2623	165	57
train	689	5035	158	41

Table 5.6. Performance comparison of the BH, BT and the final system: average precision for each category and mean average precision for three settings of final system on the YouTube-Objects V2.2 testing set. BH: Base + Hierarchical. BT: Base + Temporal. F: Final

	BH	BT	F(k=30)	F(k=40)	F(k=50)
plane	0.00	0.00	13.49	12.48	12.48
bird	0.00	40.09	62.55	61.47	61.30
boat	0.00	98.69	53.78	52.88	52.73
car	15.66	24.90	55.12	54.90	54.81
cat	0.00	0.00	25.06	26.30	27.36
cow	0.00	0.00	6.64	6.62	6.62
dog	0.00	60.34	47.26	45.11	44.92
horse	1.33	78.70	58.31	58.27	58.31
mbike	0.00	50.00	53.90	53.06	54.35
train	22.25	0.00	53.60	53.80	53.61
mAP	3.92	37.93	42.97	42.49	42.65

result. The final system that includes both hierarchical and temporal achieves the best performance overall. Among three k values, the k value of 30 has the highest mAP. There are total of 23239 object proposals in the testing set for CNN classification.

Table 5.7 compares the proposed final system with k value of 30 with five state-of-the-art methods. We can see that the proposed final system achieves the highest AP for most categories. However, AP for the cow category achieves low AP since this particular Youtube-Objects V2.2 dataset has many crowded cows. The proposed

Table 5.7. Comparison of the final system and the state-of-the-art: average precision of the proposed final system (=30) with state-of-the-art systems for each category and mean average precision on the Youtube-Objects V2.2 testing set

	R-CNN	DPM	Fine-tune SS	Fine-tune EB	Fine-tune VOP	Proposed
plane	14.10	28.42	25.57	26.52	**29.77**	13.49
bird	24.20	48.14	27.27	27.27	28.82	**62.55**
boat	16.90	25.50	27.52	33.69	35.34	**53.78**
car	27.90	48.99	35.18	36.00	41.00	**55.12**
cat	17.90	1.69	25.02	27.05	**33.7**	25.06
cow	28.60	19.24	43.01	44.76	**57.56**	6.64
dog	12.20	15.84	24.05	27.07	34.42	**47.26**
horse	29.40	35.10	41.84	44.82	54.52	**58.31**
mbike	21.30	31.61	26.70	27.07	29.77	**53.90**
train	13.2 0	39.58	20.48	24.93	29.23	**53.60**
mAP	20.570	29.411	29.664	31.918	37.413	**42.97**

hierarchical model does not perform very well under this type of scenario due to many overlapping bounding boxes during the construction of the trees. However, the proposed system has the highest mAP across all the methods, which shows the effectiveness of the proposed system.

5.4 Contributions

In summary, we propose a system to recognize objects in videos with the following contributions: First, we propose a method to select key frames from a shot or video by formulating a novel cost function and applying dynamic programming to solve it. Second, we develop a tree-based hierarchical model for objects' structural representation and a compact feature for objects' visual appearance to learn better object proposals. Third, we propose a temporal objectness measure using visual consistency during the optical flows to further reduce false positive object proposals. Last, we integrate all modules into a single large framework and apply CNN to classify object proposals. To the best of our knowledge, this is the first framework that can recognize both static and moving objects across individual frames in unconstrained videos.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

We propose a system that can automatically recognize generic objects in unconstrained videos with high precision. There are three modules for the proposed system: stable frame sequence generation, object proposal learning, and CNN classification. The stable frame sequence generation module is able to divide the video/videos into multiple stable frame sequences with little content variation in each sequence. We take advantage of spatial and temporal features of objects in videos by developing the tree-based hierarchical model and temporal objectness model to improve the object detection rate. We finally employ CNN to classify the object proposals. Extensive experiments demonstrate the effectiveness of the proposed system in both object localization and classification. Our proposed system has the following contributions:

- Proposing a system that can recognize both static and moving objects in every frame in unconstrained videos with high precision.
- Developing a shot boundary detection method using an adaptive threshold to detect both hard cuts changes and gradual changes between the shots.
- Designing a tree-based hierarchical model to capture not only the visual features but also the internal structures of the objects. This model improves the detection rate over the state-of-the-art object detectors.
- Defining a temporal objectness measure to re-rank object proposals during tracking to further improve the detection rate.
- Integrating all the proposed modules into a system to bridge the gap between domain-specific objects for classification and generated generic object proposals and therefore lead to high object recognition accuracy in videos.

The hierarchical model has relatively high computational complexity when building trees. Effectively building the hierarchical trees with reduced computation cost will be the future direction. Currently, the proposed system is feasible to run on the desktop or server environments due to the required computational cost. The training on CNN classification module is also computational intensive. With the development of computing devices such as CPUs, GPUs and FPGAs on mobile devices, training deep neural networks on mobile devices becomes possible now. Exploring ways to perform the optimization on CNN classification to make it run on the mobile devices will be interesting future research work. With some compromise in accuracy, refining the whole system to make it become a real-time object recognition on mobile devices will have a lot of applications. For example, how to instantly recognize the objects in videos or real-time scenes on mobile can provide a user with assistive information about the environment or potential shopping opportunities. Comparing with desktops, mobile devices are much more battery hungry, especially for high intensive tasks such as training deep convolution neural networks. There are some existing works on offloading the battery consumption of mobile devices to other devices [114–121]. Using and improving this type of technology will further enhance the capability of computing on mobile devices, and will make running the proposed techniques on mobile devices become a reality.

REFERENCES

[1] G. Xu and Z. Zhang, *Epipolar geometry in stereo, motion and object recognition: a unified approach*, vol. 6. Springer Science & Business Media, 2013.

[2] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[3] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for rgb-d based object recognition," in *Experimental Robotics*, pp. 387–402, Springer, 2013.

[4] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Proceedings of the IEEE European Conf. on Comput. Vision*, pp. 329–344, Springer, 2014.

[5] C. F. Cadieu, H. Hong, D. L. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo, "Deep neural networks rival the representation of primate it cortex for core visual object recognition," *PLoS Comput Biol*, vol. 10, no. 12, p. e1003963, 2014.

[6] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: A survey," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 36, no. 11, pp. 2270–2287, 2014.

[7] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: A review," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 36, no. 7, pp. 1354–1369, 2014.

[8] A. Barsegyan, J. L. McGaugh, and B. Roozendaal, "Noradrenergic activation of the basolateral amygdala modulates the consolidation of object-in-context recognition memory," *Frontiers in Behavioral Neuroscience*, vol. 8, p. 160, 2014.

[9] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[10] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. Journal of Comput. Vision*, vol. 105, no. 3, pp. 222–245, 2013.

[11] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Good practice in large-scale learning for image classification," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 36, no. 3, pp. 507–520, 2014.

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

[13] J. Yu, Y. Rui, Y. Y. Tang, and D. Tao, "High-order distance-based multiview stochastic learning in image classification," *IEEE Trans. on Cybernetics*, vol. 44, no. 12, pp. 2431–2442, 2014.

[14] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?," *IEEE Trans. on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[15] Y. Huang, Z. Wu, L. Wang, and T. Tan, "Feature coding in image classification: A comprehensive study," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 36, no. 3, pp. 493–506, 2014.

[16] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Trans. on Geoscience and Remote sensing*, vol. 51, no. 1, pp. 217–231, 2013.

[17] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 35, no. 12, pp. 2916–2929, 2013.

[18] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Comput. Vision and Pattern Recognition (CVPR), 2010 IEEE Conf. on*, pp. 3384–3391, IEEE, 2010.

[19] H. Müller, P. Clough, T. Deselaers, B. Caputo, and I. CLEF, "Experimental evaluation in visual information retrieval," *The Information Retrieval Series*, vol. 32, 2010.

[20] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proceedings of the IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 3424–3431, IEEE, 2010.

[21] Y. Zhang, Z. Jia, and T. Chen, "Image retrieval with geometry-preserving visual phrases," in *IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 809–816, IEEE, 2011.

[22] B. Siddiquie, R. S. Feris, and L. S. Davis, "Image ranking and retrieval based on multi-attribute queries," in *IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 801–808, IEEE, 2011.

[23] C. B. Akgül, D. L. Rubin, S. Napel, C. F. Beaulieu, H. Greenspan, and B. Acar, "Content-based image retrieval in radiology: current status and future directions," *Journal of Digital Imaging*, vol. 24, no. 2, pp. 208–222.

[24] S. Murala, R. Maheshwari, and R. Balasubramanian, "Local tetra patterns: a new feature descriptor for content-based image retrieval," *IEEE Trans. on Image Processing*, vol. 21, no. 5, pp. 2874–2886, 2012.

[25] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Proceedings of the IEEE European Conf. on Comput. Vision*, pp. 584–599, Springer, 2014.

[26] Z.-C. Huang, P. P. Chan, W. W. Ng, and D. S. Yeung, "Content-based image retrieval using color moment and gabor texture feature," in *Proceedings of IEEE Int. Conf. on Machine Learning and Cybernetics*, vol. 2, pp. 719–724, IEEE, 2010.

[27] S. Russell, P. Norvig, and A. Intell., "A modern approach," *Artificial Intell.. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[28] W. Barfield, *Fundamentals of wearable Comput. and augmented reality.* CRC Press, 2015.

[29] M. S. Patel, D. A. Asch, and K. G. Volpp, "Wearable devices as facilitators, not drivers, of health behavior change," *Jama*, vol. 313, no. 5, pp. 459–460, 2015.

[30] V. Rosenzveig, S. Briot, P. Martinet, E. Özgür, and N. Bouton, "A method for simplifying the analysis of leg-based visual servoing of parallel robots," in *IEEE Int. Conf. on Robotics and Automation*, pp. 5720–5727, 2014.

[31] J. Aggarwal, *Multisensor fusion for Comput. vision*, vol. 99. Springer Science & Business Media, 2013.

[32] J. Yu, S. Skaff, L. Peng, and F. Imai, "Leveraging knowledge-based inference for material classification," in *ACM Int. Conf. on Multimedia*, pp. 1243–1246, ACM, 2015.

[33] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE Int. Conf. on Comput. Vision*, pp. 2722–2730, 2015.

[34] C. Berger and B. Rumpe, "Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system," *arXiv preprint arXiv:1409.0413*, 2014.

[35] A. Pugh, *Robot vision.* Springer Science & Business Media, 2013.

[36] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[37] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Proceedings of the IEEEE Comput. Vision and Pattern Recognition*, vol. 2, pp. II–II, IEEE, 2004.

[38] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Comput. Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[39] B. Zhang, Y. Gao, S. Zhao, and J. Liu, "Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor," *IEEE Trans. on Image Processing*, vol. 19, no. 2, pp. 533–544, 2010.

[40] L. Peng, Y. Yang, and H. Wang, "Automatic face annotation method and system," Nov. 3 2015. US Patent 9,176,987.

[41] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. on pattern analysis and machine Intell.*, vol. 20, no. 1, pp. 39–51, 1998.

[42] H. A. Rowley, S. Baluja, T. Kanade, *et al.*, *Human face detection in visual scenes*. Carnegie-Mellon University. Department of Comput. Science, 1995.

[43] S. Ben-Yacoub, "Fast object detection using mlp and fft," tech. rep., IDIAP, 1997.

[44] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *Comput. Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conf. on*, pp. 1–8, IEEE, 2008.

[45] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele, "Sliding-windows for rapid object class localization: A parallel technique," in *Pattern Recognition*, pp. 71–81, Springer, 2008.

[46] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA Int. symposium on Field Programmable Gate Arrays*, pp. 47–56, ACM, 2012.

[47] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Comput. Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Comput. Society Conf. on.*

[48] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation,"

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Comput. Vision and Pattern Recognition (CVPR), 2014 IEEE Conf. on*, pp. 580–587, IEEE, 2014.

[50] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *Proceedings of the European Conf. on Comput. Vision*, pp. 345–360, Springer, 2014.

[51] J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," in *Comput. Vision, 2005. ICCV 2005. Tenth IEEE Int. Conf. on*, vol. 1, pp. 503–510, IEEE, 2005.

[52] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Trans. on pattern analysis and machine Intell.*, vol. 30, no. 1, pp. 36–51, 2008.

[53] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Object detection by contour segment networks," in *European Conf. on Comput. vision*, pp. 14–28, Springer, 2006.

[54] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. Journal of Comput. Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[55] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Comput. Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conf. on*, pp. 1–8, IEEE, 2008.

[56] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Comput. Vision and Pattern Recognition (CVPR), 2014 IEEE Conf. on*, pp. 2155–2162, IEEE, 2014.

[57] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, pp. 2553–2561, 2013.

[58] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Comput. Vision–ECCV 2014*, pp. 391–405, Springer, 2014.

[59] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010.

[60] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual Int. Conf. on Machine Learning*, pp. 609–616, ACM, 2009.

[61] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. on pattern analysis and machine Intell.*, vol. 38, no. 1, pp. 142–158, 2016.

[62] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE Int. Conf. on Comput. Vision*, pp. 1440–1448, 2015.

[63] 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*, pp. 91–99, 2015.

[64] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *Comput. Vision and Pattern Recognition (CVPR), 2014 IEEE Conf. on*.

[65] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results." http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[66] G. Sharir and T. Tuytelaars, "Video object proposals," in *Proceedings of the IEEE Workshops on Comput. Vision and Pattern Recognition*, pp. 9–14, IEEE, 2012.

[67] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid, "Spatio-temporal object detection proposals," in *Proceedigns of the European Conf. on Comput. Vision*, Sept. 2014.

[68] Y. Hua, K. Alahari, and C. Schmid, "Online object tracking with proposal selection," in *Proceedings of the IEEE Int. Conf. on Comput. Vision*, pp. 3092–3100, 2015.

[69] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*, pp. 135–144, 2002.

[70] D. H. H. Santosh, P. Venkatesh, L. Rao, and N. Kumar, "Tracking multiple moving objects using gauusian mixture model," *Int. Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, no. 2, 2013.

[71] W. Abd-Almageed, "Online, simultaneous shot boundary detection and key frame extraction for sports videos using rank tracing," in *IEEE Int. Conf. on Image Processing*, pp. 3200–3203, IEEE, 2008.

[72] S. Utz, M. Tanis, and I. Vermeulen, "It is all about being popular: The effects of need for popularity on social network site use," 2012 Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA.

[73] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *Int. Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 2015.

[74] A. Hanjalic, "Shot-boundary detection: Unraveled and resolved?," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, pp. 90–105, Feb. 2002.

[75] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[76] L. Peng, Y. Yang, X. Qi, and H. Wang, "Highly accurate video object identification utilizing hint information," in *Int. Conf. on Computing, Networking and Communications (ICNC)*, pp. 317–321, IEEE, 2014.

[77] L. Peng and H. Wang, "Object identification system and method," Sept. 1 2015. US Patent 9,122,931.

[78] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 12, no. 9, pp. 855–867, 1990.

[79] L. Peng and X. Qi, "A hierarchical model to learn object proposals and its applications," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 5, pp. 2543–2551, 2016.

[80] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in *Comput. Vision and Pattern Recognition (CVPR), 2012 IEEE Conf. on*, pp. 3282–3289, IEEE, 2012.

[81] Z. Kalal, K. Mikolajczyk, and J. Matas, "Face-tld: Tracking-learning-detection applied to faces," in *IEEE Int. Conf. on Image Processing (ICIP)*, pp. 3789–3792, IEEE, 2010.

[82] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *Int. Joint Conf. on Artificial Intell.*, vol. 81, pp. 674–679, 1981.

[83] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *IEEE Internatioinal Conf. on Comput. Vision and Pattern Recognition*, pp. 1838–1845, IEEE, 2013.

[84] L. Peng and X. Qi, "Temporal objectness: Model-free learning of object proposals in video," in *Image Processing (ICIP), 2016 IEEE Int. Conf. on*, pp. 3663–3667, IEEE, 2016.

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[86] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[87] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[88] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. on Acoustics, Speech, and Signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[89] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[90] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?," in *Int. Conf. on database theory*, pp. 217–235, Springer, 1999.

[91] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[92] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI Workshop on Learning for Text Categorization*, vol. 752, pp. 41–48, Citeseer, 1998.

[93] D. R. Cox, "Regression models and life-tables," in *Breakthroughs in statistics*, pp. 527–541, Springer, 1992.

[94] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[95] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[96] 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 Conf. on Comput. Vision and Pattern Recognition*, pp. 1–9, 2015.

[97] M. Singh and M. P. Singh, "Augmented reality interfaces," *IEEE Internet Computing*, vol. 17, no. 6, pp. 66–70, 2013.

[98] S. B. Gokturk, D. Anguelov, V. O. Vanhoucke, K.-c. Lee, D. T. Vu, D. Yang, M. Shah, and A. Khan, "System and method for providing objectified image renderings using recognition information from images," Aug. 30 2016. US Patent 9,430,719.

[99] F. N. Iandola, A. Shen, P. Gao, and K. Keutzer, "Deeplogo: Hitting logo recognition with the deep neural network hammer," *arXiv preprint arXiv:1510.02131*, 2015.

[100] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[101] G. Hee Lee, F. Faundorfer, and M. Pollefeys, "Motion estimation for self-driving cars with a generalized camera," in *Proceedings of the IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 2746–2753, 2013.

[102] M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina, "Suspended traffic lights detection and distance estimation using color features," in *Proceedings of IEEE Int. Conf. on Intelligent Transportation Systems*, pp. 1315–1320, 2012.

[103] R. de Charette and F. Nashashibi, "Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates," in *IEEE Intelligent Vehicles Symposium*, pp. 358–363, 2009.

[104] C. Yu, C. Huang, and Y. Lang, "Traffic light detection during day and night conditions by a camera," in *IEEE 10th Int. Conf. on Signal Processing*, pp. 821–824, IEEE, 2010.

[105] G. Sommer, *Geometric computing with Clifford algebras: theoretical foundations and applications in Comput. vision and robotics*. Springer Science & Business Media, 2013.

[106] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling.," in *Interspeech*, pp. 338–342, 2014.

[107] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[108] J. Norberto Pires, "Robot-by-voice: Experiments on commanding an industrial robot using the human voice," *Industrial Robot: An Int. Journal*, vol. 32, no. 6, pp. 505–511, 2005.

[109] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conf. on Comput. Vision and Pattern Recognition*, pp. 3156–3164, 2015.

[110] 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," *Int. Journal of Comput. Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[111] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[112] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition," in *IEEE Int. Conf. on Comput. Vision*, pp. 1879–1886, IEEE, 2011.

[113] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen, "Detecting temporally consistent objects in videos through object class label propagation," *arXiv preprint arXiv:1601.05447*, 2016.

[114] H. Qian and D. Andresen, "Extending mobile devices battery life by offloading computation to cloud," in *2nd ACM Int. Conf. on Mobile Software Engineering and Systems*, 2015.

[115] H. Qian and D. Andresen, "Jade: An efficient energy-aware computation offloading system with heterogeneous network interface bonding for ad-hoc networked mobile devices," in *Proceedings of the 15th IEEE/ACIS Int. Conf. on Software Engineering, Artificial Intell., Networking and Parallel/Distributed Computing*, 2014.

[116] H. Qian and D. Andresen, "An energy-saving task scheduler for mobile devices," in *Proceedings of the 14th IEEE/ACIS Int. Conf. on Comput. and Information Science*, pp. 423–430, IEEE, 2015.

[117] H. Qian and D. Andresen, "Emerald: Enhance scientific workflow performance with computation offloading to the cloud," in *Proceedings of the 14th IEEE/ACIS Int. Conf. on Comput. and Information Science*, pp. 443–448, IEEE, 2015.

[118] H. Qian and D. Andresen, "Reducing mobile device energy consumption with computation offloading," in *IEEE/ACIS Int. Conf. on Software Engineering, Artificial Intell., Networking and Parallel/Distributed Computing (SNPD)*, pp. 1–8, IEEE, 2015.

[119] H. Qian and D. Andresen, "Jade: Reducing energy consumption of android app," *the Int. Journal of Networked and Distributed Computing (IJNDC), Atlantis press*, vol. 3, no. 3, pp. 150–158, 2015.

[120] L. Peng, "Enhanced camera capturing using object-detection-based autofocus on smartphones," in *Int. Conf. on Computational Science/Intell. and Applied Informatics (CSII)*, pp. 208–212, IEEE, 2016.

[121] L. Peng, "Gscheduler: Reducing mobile device energy consumption," in *4th Int. Conf. on Applied Computing and Information Technology (ACIT)*, pp. 1–6, IEEE, 2016.

CURRICULUM VITAE

Liang Peng

EDUCATION

Ph.D., Computer Science. Utah State University, Logan, UT. 2017.

M.S., Statistics. Kansas State University, Manhattan, KS. 2011.

B.S., Economics (Minor in Math). Emporia State University, Emporia, KS. 2008.

RESEARCH INTERESTS

Image processing computer vision, machine learning, deep learning, artificial intelligence

CONFERENCE PUBLICATIONS

Liang Peng, Yimin Yang, Xiaojun Qi, Haohong Wang. Highly accurate video object identification utilizing hint information. International Conference on Computing, Networking and Communications (ICNC 2014) in Honolulu, Hawaii, Feb. 2014

Jie Yu, Sandra Skaff, Liang Peng, Francisco Imai. Leveraging Knowledge-based Inference for Material Classification. In Proceedings of the 23rd Annual ACM Conference on Multimedia Conference (MM15). ACM, New York, NY, USA, 2015

Liang Peng, Xiaojun Qi. A Hierarchical Model to Learn Object Proposals and Its Application. Journal of Intelligent & Fuzzy Systems. Special Issue: Multimedia in Technology Enhanced Learning, Feb. 2016

Liang Peng, Xiaojun Qi. Temporal Objectness: Model-Free Learning of Object Proposals in Video. IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, Oct. 2016

U.S. PATENTS

Object Identification System and Method. Liang Peng, Haohong Wang (US20150117703 A1, Issued in 2015)

Automatic Face Annotation Method and System. Liang Peng, Yimin Yang, Haohong Wang (US9176987 B1, Issued in 2015)

INDUSTRY EXPERIENCE

Summer intern, Ads and Data Team, Yahoo! Inc., Sunnyvale, May-Aug. 2015

Research intern, Computational Imaging, Canon USA, San Jose, May-Dec. 2014

Research assistant, TCL Research America, San Jose, Dec. 2012 - May 2014

Summer intern, IM Flash Technologies, Lehi, UT, May-July 2012

TEACHING EXPERIENCE

STAT 325, 250: Intro. to Statistics, Kansas State University, 2008 to 2011.

CS 2410: Intro. to Java GUI, Utah State University, 2011 to 2012.

SKILLS

Proficient in software development using Java (6 years) Python (4 years), R (4 years), C# (4 years), Matlab (5 years), familiar with C++

Proficient in operating systems and toolkits of Linux, Mac OS X, Windows, AWS, Caffe, Tensorflow, Hive, Hadoop, PyLearn2, OpenCV

Deep understanding hands-on experiences of machine learning and computer vision algorithms

Excellent analytic and implementation skills in handling big data

Strong statistical analytic skills in regression, hypothesis testing and experimental design