ABSTRACT

Title of dissertation: IMPROVING EFFICIENCY FOR OBJECT DETECTION AND TEMPORAL MODELING FOR ACTION LOCALIZATION

Mingfei Gao
Doctor of Philosophy, 2019

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

Despite their great predictive capability, Convolutional Neural Networks (CNNs) are computational-expensive to deploy and usually require a tremendous amount of annotated data at training time. When analyzing videos, it is very important and challenging to model temporal dynamics due to large appearance variation and complex semantics. We propose methods to improve efficiency of model deployment for object detection in images and to capture temporal dependencies for online action detection in videos. To relieve the demand of human labor for data annotation, we introduce approaches to conduct object detection and natural language localization using weak supervisions.

First, we introduce a generic framework that reduces the computational cost of object detection while retaining accuracy for scenarios where objects with varied sizes appear in high resolution images. Detection progresses in a coarse-to-fine manner, first on a down-sampled version of the image and then on a sequence

of higher resolution regions identified as likely to improve the detection accuracy. Built upon reinforcement learning, our approach consists of a model (R-net) that uses coarse detection results to predict the potential accuracy gain for analyzing a region at a higher resolution and another model (Q-net) that sequentially selects regions to zoom in.

Second, we propose a novel framework, Temporal Recurrent Network (TRN), to model greater temporal context of a video frame by simultaneously performing online action detection and anticipation of the immediate future. At each moment in time, our approach makes use of both accumulated historical evidence and predicted future information to better recognize the action that is currently occurring, and integrates both of these into a unified end-to-end architecture. We evaluate our approach on two popular online action detection datasets, HDD and TVSeries, as well as another widely used dataset, THUMOS'14.

Third, we propose StartNet to address Online Detection of Action Start (ODAS) where action starts and their associated categories are detected in untrimmed, streaming videos. Our method decomposes ODAS into two stages: action classification (using ClsNet) and start point localization (using LocNet). ClsNet focuses on per-frame labeling and predicts action score distributions online. Based on the predicted action scores of the past and current frames, LocNet conducts class-agnostic start detection by optimizing long-term localization rewards using policy gradient methods. The proposed framework is validated on two large-scale datasets, THUMOS'14 and ActivityNet.

Fourth, we introduce Count-guided Weakly Supervised Localization (C-WSL),

an approach that uses per-class object count as a new form of supervision to improve Weakly Supervised Localization (WSL). C-WSL uses a simple count-based region selection algorithm to select high-quality regions, each of which covers a single object instance during training, and improves existing WSL methods by training with the selected regions. To demonstrate the effectiveness of C-WSL, we integrate it into two WSL architectures and conduct extensive experiments on VOC2007 and VOC2012.

In the last, we propose Weakly Supervised Language Localization Networks (WSLLN) to detect events in long, untrimmed videos given language queries. WSLLN relieves the annotation burden by training with only video-sentence pairs without accessing to temporal locations of events. With a simple end-to-end structure, WSLLN measures segment-text consistency and conducts segment selection (conditioned on the text) simultaneously. Results from both are merged and optimized as a video-sentence matching problem. Experiments are conducted on ActivityNet Captions and DiDeMo.

IMPROVING EFFICIENCY FOR OBJECT DETECTION AND TEMPORAL MODELING FOR ACTION LOCALIZATION

by

Mingfei Gao

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Professor David Jacobs
Professor Ramani Duraiswami
Professor Tom Goldstein

Acknowledgments

First of all, I want to express my gratefulness to my advisor, Professor Larry Davis. As one of the greatest researchers in the field, Larry is very knowledgeable in diverse areas of research topics. He always encourages me to be creative and provides me great opportunities to explore interesting topics. Larry is great at advising. He inspires me by introducing promising directions and gives me enough freedom to go deep into the research. This is very important for me to be an independent researcher. I want to especially thank Larry for his kindness and wisdom. Thank him for admitting me to his group when I was very junior with limited related experience and for being very patient when I constantly proposed immature ideas in the early stage of my PhD. It is really my honor to be his student.

Meanwhile, I want to acknowledge the help of people who work closely with me. When I was a beginner, I obtained great help from Vlad Morariu. As my research mentor, he discussed ideas with me, worked through my research plans and helped polish papers. I am really grateful for his support. I want to thank my colleagues, Yaming Wang, Ruichi Yu, Ang Li and Mingze Xu for their great collaborations. I also want to express thankfulness to the mentors of my internships including Ashish Tawari at Honda Research Institute, Caiming Xiong at Salesforce Research and Zizhao Zhang, Tomas Pfister at Google. I would also like to thank my lab mates, Zuxuan Wu, Hengduo Li, Peng Zhou, Xitong Yang, Shiyi Lan, Luyu Yang and Jun Wang who make my PhD life joyful.

My research was partially supported by the Intelligence Advanced Research

Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00345. I really appreciate the funding that supports my research work.

Finally, I own my deepest gratitude to my family and friends for their unlimited love. Thank my mother, Liping Liu who always stands by my side. Thank my aunty Aixiang Liu and her family who took care of me since I was a little child. Thank my dearest Tong Guan who unconditionally understands, supports and trusts me as always. I would also like to thank my friends, Chaoyue Liu, Fanxi Qi and Ran Tao who make my life colorful.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

IoU	Intersection over Union
AP	Average Precision
mAP	mean Average Precision
cAP	calibrated Average Precision
FPS	Frame Per Second
GT	Ground Truth
WS	Weakly Supervised
WSL	Weakly Supervised Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Units
RCNN	Region-based Convolution Neural Network
MIL	Multiple Instance Learning
NLL	Natural Language Localization
OAD	Online Action Detection
ODAS	Online Detection of Action Start
DZN	Dynamic Zoom-in Network
C-WSL	Count-guided Weakly Supervised Localization
CRS	Count-based Region Selection
TRN	Temporal Recurrent Network
WSLLN	Weakly Supervised Language Localization Network

Chapter 1: Introduction

1.1 Proposed Approach

Despite their great predictive capability, Convolutional Neural Networks (CNNs) are computational-expensive to deploy and usually require a tremendous amount of annotated data at training time. When analyzing videos, it is very important and challenging to model temporal dynamics due to large appearance variation and complex semantics. We propose methods to improve efficiency of model deployment for object detection in images and to capture temporal dependencies for online action detection in videos. To relieve the demand of human labor for data annotation, we introduce approaches to conduct object detection and natural language localization using weak supervisions.

In the first part, we introduce, Dynamic Zoom-in Network (DZN), a generic framework that reduces the computational cost of object detection while retaining accuracy for scenarios where objects with varied sizes appear in high resolution images. Detection progresses in a coarse-to-fine manner, first on a down-sampled version of the image and then on a sequence of higher resolution regions identified as likely to improve the detection accuracy. Built upon reinforcement learning, our approach consists of a model (R-net) that uses coarse detection results to predict

the potential accuracy gain for analyzing a region at a higher resolution and another model (Q-net) that sequentially selects regions to zoom in. Final detection results are obtained by combining the coarse and fine predictions. Experiments on the Caltech Pedestrians dataset show that our approach reduces the number of processed pixels significantly without a drop in detection accuracy. The merits of our approach become more significant on a high resolution test set collected from YFCC100M dataset.

In the second part, we propose, Temporal Recurrent Network (TRN), to address online action detection in videos. Most work on temporal action detection is formulated as an offline problem, in which the start and end times of actions are determined after the entire video is fully observed. However, important real-time applications including surveillance and driver assistance systems require identifying actions as soon as each video frame arrives, based only on current and historical observations. Our novel framework models greater temporal context of a video frame by simultaneously performing online action detection and anticipation of the immediate future. At each moment in time, our approach makes use of both accumulated historical evidence and predicted future information to better recognize the action that is currently occurring, and integrates both of these into a unified end-to-end architecture. We evaluate our approach on two popular online action detection datasets, HDD and TVSeries, as well as another widely used dataset, THUMOS'14.

In the third part, we propose StartNet to address Online Detection of Action Start (ODAS) where action starts and their associated categories are detected in untrimmed, streaming videos. Previous methods aim to localize action starts by

learning feature representations that can directly separate the start point from its preceding background. It is challenging due to the subtle appearance difference near the action starts and the lack of training data. Instead, our method decomposes ODAS into two stages: action classification (using ClsNet) and start point localization (using LocNet). ClsNet focuses on per-frame labeling and predicts action score distributions online. Based on the predicted action scores of the past and current frames, LocNet conducts class-agnostic start detection by optimizing long-term localization rewards using policy gradient methods. The proposed framework is validated on two large-scale datasets, THUMOS'14 and ActivityNet.

In the fourth part, we introduce Count-guided Weakly Supervised Localization (C-WSL), an approach that uses per-class object count as a new form of supervision to improve weakly supervised localization (WSL). Previous WSL methods utilizes image-class to supervise the training process. However, their detectors tend to group multiple objects of the same class as a single instance at test time, since the image-class label has no information of object counts. C-WSL uses a simple count-based region selection algorithm to select high-quality regions, each of which covers a single object instance during training, and improves existing WSL methods by training with the selected regions. To demonstrate the effectiveness of C-WSL, we integrate it into two WSL architectures and conduct extensive experiments on VOC2007 and VOC2012.

In the last part, we propose Weakly Supervised Language Localization Networks (WSLLN) to detect events in long, untrimmed videos given language queries. To learn the correspondence between visual segments and texts, most previous meth-

ods require temporal coordinates (start and end times) of events for training, which leads to high costs of annotation. WSLLN relieves the annotation burden by training with only video-sentence pairs without accessing to temporal locations of events. With a simple end-to-end structure, WSLLN measures segment-text consistency and conducts segment selection (conditioned on the text) simultaneously. Results from both are merged and optimized as a video-sentence matching problem. Experiments are conducted on ActivityNet Captions and DiDeMo.

1.2 Related Work

1.2.1 CNN based Object Detectors

One way to deploy an object detector efficiently on high resolution images is to improve the underlying structure of the detector. Girshick *et al.* [8] speed up the region proposal based CNN [9] by sharing convolutional features between proposals. Ren *et al.* propose Faster R-CNN [2], a fully end-to-end pipeline that shares features between proposal generation and object detection, improving both accuracy and computational efficiency. He *et al.* propose Mask R-CNN [10] which extends Faster R-CNN by adding object mask prediction as an auxiliary task. Single-shot detectors [11, 12, 13] have received much attention for real-time performance. These methods remove the proposal generation stage and formulate detection as a regression problem. Although these detectors performed well on PASCAL VOC [14, 15] and MS COCO [16] datasets, which generally contain large objects in images with relatively low resolution, they do not generalize as well on large images with objects

of variable sizes. Also, their processing cost increases dramatically with image size due to the large number of convolution operations.

To ease the burden of human annotation, Weakly Supervised Localization (WSL) methods are proposed to train a detector using weak supervision. Most WSL detectors utilize image-class label for training. Bilen *et al.* [5] propose a two-stream CNN architecture to classify and localize simultaneously and train the network in an end-to-end manner. Following [5], Kantorov *et al.* [17] add *additive* and *contrastive* models to improve localization on object boundaries instead of local parts. Singh *et al.* [18] propose the 'Hide-and-Seek' framework which hides informative patches to encourage WSL to detect complete object instances. In [19], Li *et al.* conduct progressive domain adaption and significantly improved the localization ability of the baseline detector. Diba *et al.* [20] perform WSL in two/three cascaded stages to find the best candidate location based on a generated class activation map. Jie *et al.* propose a self-taught learning approach in [21] which alternates between classifier training and online supportive sample harvesting. Similarly, in [6], Tang *et al.* design an online classifier refinement pipeline to progressively locate the most discriminative region of an image.

Alternatively, other types of weak supervisions are proposed to reduce annotation cost. [22] propose a novel framework where an annotator verifies predicted results instead of manually drawing boxes. Kolesnikov *et al.* [23] assign object or distractor labels to co-occuring objects in images to improve WSL. Papadopoulos *et al.* [24] propose click supervision and integrate it into existing MIL-based methods to improve localization performance. However, these methods either highly

depend on the produced results and require frequent interactions with annotators or require annotators to search for and click on each instance in an image.

1.2.2 Temporal Action Localization

Most existing methods [25, 26, 27, 28, 29, 30] on temporal action detection formulate the problem in an offline manner. These methods segment actions from long, untrimmed videos and require observing the entire video before making a decision. S-CNN [25] localizes actions with three stages: action proposal generation, proposal classification, and proposal regression. Dai *et al.* [27] propose TCN which incorporates local context of each proposal for proposal ranking. By sharing features between proposal generation and classification, R-C3D [31] reduces computational cost significantly. Buch *et al.* [28] propose an efficient proposal generation model that avoids working on overlapping regions. Instead of treating temporal action detection as segment-level classification, Shou *et al.* [32] propose CDC network to produce per-frame predictions using 3D convolutional networks.

Online action detection is usually solved as a per-frame labeling task [33] on live, streaming videos. As soon as a video frame arrives, it is classified to an action class or background without accessing future frames. De Geest *et al.* [33] first introduced the problem and proposed several models as baselines. Gao *et al.* [34] propose a Reinforced Encoder-Decoder network for action anticipation and treat online action detection as a special case of their framework.

The goal of the above mentioned temporal action detectors is to localize actions

in pre-defined categories. However, activities in the wild is very complicated and it is challenging to cover all the activities of interest by using a finite set of categories.

Natural Language Localization (NLL) in untrimmed videos was first introduced in [4, 7], where given an arbitrary text query, the methods attempt to localize the text (predict its start and end times) in a video. Hendricks *et al.* propose MCN [4] which embeds the features of visual proposals and sentence representations in the same space and ranks proposals according their similarity with the sentence. Gao *et al.* propose CTRL [7], where alignment and regression are conducted for clip candidates. Liu *et al.* introduce TMN [35] which measures the clip-sentence alignment guided by the semantic structure of the text query. Later, Hendricks *et al.* propose MLLC [36] that explicitly reasons about temporal clips of a video. Zhang *et al.* propose MAN [37] which utilizes Graph Convolutional Networks [38] to model temporal relations among visual clips.

Annotating actions in videos is very expensive. Instead of using temporally labeled segments, weakly supervised action detectors use weaker supervisions, *e.g.*, movie script [39, 40], the order of the occurring action classes in videos [41, 42] and video-level class labels [43, 44]. Duan *et al.* proposed WSDEC to handle weakly supervised dense event captioning in [3] by alternating between language localization and caption generation iteratively. This approach can be used as a weakly supervised natural language detector.

1.3 Outline of Thesis

The thesis is organized as follows. Chapter 2 introduces, DZN, a framework to improve efficiency of object detection. This chapter is based on our work in [45]. Chapter 3 is based on the work in [46], where TRN are proposed to address online action detection. Chapter 4 based on the work in [47] introduces StartNet, an effective framework for online detection of action starts. In Chapter 5, we propose C-WSL to handle weakly supervised object detection using object counts as supervision. This chapter is based on our work in [48]. Chapter 6 based on [49] introduces WSLLN, a weakly supervised natural language detector. The thesis is concluded in Chapter 7.

Chapter 2: Dynamic Zoom-in Network for Fast Object Detection in Large Images

2.1 Introduction

Most recent convolutional neural network (CNN) detectors are applied to images with relatively low resolution, *e.g.*, VOC2007/2012 (about 500×400) [14, 15] and MS COCO (about 600×400) [16]. At such low resolutions, the computational cost of convolution is low. However, the resolution of everyday devices has quickly outpaced standard computer vision datasets. The camera of a 4K smartphone, for instance, has a resolution of 2,160×3,840 pixels and a DSLR camera can reach 6,000×4,000 pixels. Applying state-of-the-art CNN detectors directly to those high resolution images requires a large amount of processing time. Additionally, the convolution output maps are too large for the memory of current GPUs.

Prior works address some of these issues by simplifying the network architecture [50, 51, 52, 53, 54] to speed up detection and reduce GPU memory consumption. However, these models are tailored to particular network structures and may not generalize well to new architectures. A more general direction is treating the detector as a black box that is judiciously applied to optimize accuracy and

efficiency. For example, one could partition an image into sub-images that satisfy memory constraints and apply the CNN to each sub-image. However, this solution is still computationally burdensome. One could also speed up detection process and reduce memory requirements by running existing detectors on down-sampled images. However, the smallest objects may become too small to detect in the down-sampled images. Object proposal methods are the basis for most CNN detectors, restricting expensive analysis to regions that are likely to contain objects of interest [55, 56, 57, 58]. However, the number of object proposals needed to achieve good recall for small objects in large images is prohibitively high which leads to huge computational cost.

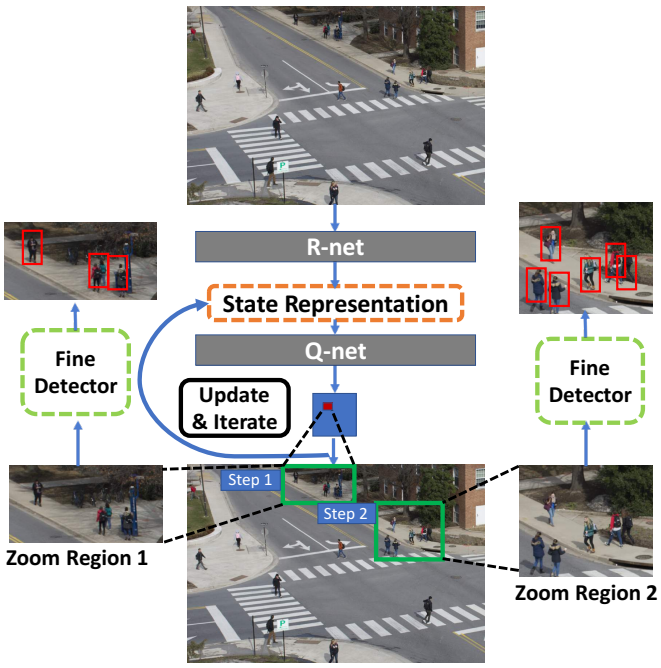


Figure 2.1: Illustration of our approach. The input is a down-sampled version of the image to which a coarse detector is applied. The R-net uses the initial coarse detection results to predict the utility of zooming in on a region to perform detection at higher resolution. The Q-net, then uses the computed accuracy gain map and a history of previous zooms to determine the next zoom that is most likely to improve detection with limited computational cost.

Our approach is illustrated in Fig. 2.1. We speed up object detection by first performing coarse detection on a down-sampled version of the image and then sequentially selecting promising regions to be analyzed at a higher resolution. We employ reinforcement learning to model long-term reward in terms of detection accuracy and computational cost and dynamically select a sequence of regions to analyze at higher resolution. Our approach consists of two networks: a zoom-in accuracy gain regression network (R-net) learns correlations between coarse and fine detections and predicts the accuracy gain for zooming in on a region; a zoom-in Q function network (Q-net) learns to sequentially select the optimal zoom locations and scales by analyzing the output of the R-net and the history of previously analyzed regions.

Experiments demonstrate that, with a negligible drop in detection accuracy, our method reduces processed pixels by over 50% and average detection time by 25% on the Caltech Pedestrian Detection dataset [59], and reduces processed pixels by about 70% and average detection time by over 50% on a high resolution dataset collected from YFCC100M [60] that has pedestrians of varied sizes. We also compare our method to recent single-shot detectors [11, 13] to show our advantage when handling large images.

2.2 Dynamic zoom-in network

Our work employs a coarse-to-fine strategy, applying a coarse detector at low resolution and using the outputs of this detector to guide an in-depth search for

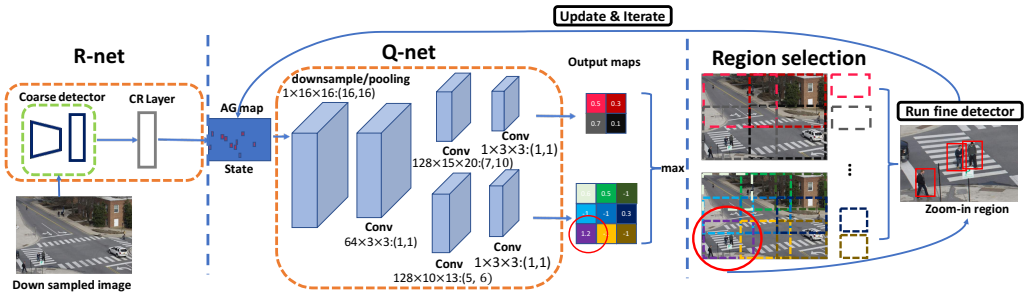


Figure 2.2: Given a down-sampled image as input, the R-net generates an initial accuracy gain (AG) map indicating the potential zoom-in accuracy gain of different regions (initial state). The Q-net is applied iteratively on the AG map to select regions. Once a region is selected, the AG map will be updated to reflect the history of actions. For the Q-net, two parallel pipelines are used, each of which outputs an action-reward map that corresponds to selecting zoom-in regions with a specific size. The value of the map indicates the likelihood that the action will increase accuracy at low cost. Action rewards from all maps are considered to select the optimal zoom-in region at each iteration. The notation $128\times15\times20$:(7,10) means 128 convolution kernels with size 15×20, and stride of 7/10 in height/width. Each grid cell in the output maps is given a unique color, and a bounding box of the same color is drawn on the image to denote the corresponding zoom region size and location.

objects at high resolution. The intuition is that, while the coarse detector will not be as accurate as the fine detector, it will identify image regions that need to be further analyzed, incurring the cost of high resolution detection only in promising regions. We make use of two major components: 1) a mechanism for learning the statistical relationship between the coarse and fine detectors, so that we can predict which regions need to be zoomed in given the coarse detector output; and 2) a mechanism for selecting a sequence of regions to analyze at high resolution, given the coarse detector output and the regions that have already been analyzed by the fine detector. Our pipeline is illustrated in Fig. 2.2. We learn a strategy that models the long-term goal of maximizing the overall detection accuracy with limited cost.

2.2.1 Problem formulation

Our work is formulated as a Markov Decision Process (MDP) [61]. At each step, the system observes the current state, estimates potential cost-aware rewards of taking different actions and selects the action that has the maximum long-term cost-aware reward.

Action. Our algorithm sequentially analyzes regions with high zoom-in reward at high resolution. In this context, an *action* corresponds to selecting a region to analyze at high resolution. Each action a can be represented by a tuple (x, y, w, h) where (x, y) indicates the location, and (w, h) specifies the size of the region. At each step, the algorithm scores a set of potential actions—a list of rectangular regions—in terms of the potential long-term reward of taking those actions.

State. The representation encodes two types of information: 1) the predicted accuracy gain of regions yet to be analyzed; and 2) the history of regions that have already been analyzed at high resolution (the same region should not be zoomed in multiple times). We design a zoom-in accuracy gain regression network (R-net) to learn an informative accuracy gain map (AG map) as the state representation from which the zoom-in Q function can be successfully learned. The AG map has the same width and height as the input image. The value of each pixel in the AG map is an estimate of how much the detection accuracy might be improved if that pixel in the input image were included by the zoom-in region. As a result, the AG map provides detection accuracy gain for selecting different actions. After an action is taken, values corresponding to the selected region in the AG map decrease

accordingly, so the AG map can dynamically record action history.

Cost-aware reward function. The state representation encodes the predicted accuracy gain of zooming in on each image subregion. To maintain a high accuracy with limited computation, we define a cost-aware reward function for actions. Given state s and action a, the cost-aware reward function scores each action (zoom region) by considering both cost increment and accuracy improvement as

$$R(s,a) = \sum_{k \ in \ a} |g_k - p_k^l| - |g_k - p_k^h| - \lambda \frac{b}{B} \tag{2.1}$$

where $k \ in \ a$ means that proposal k is included in the region selected by action a. p_k^l and p_k^h indicates coarse and fine detection scores, and g_k is the corresponding ground-truth label. The variable b represents the total number of pixels included in the selected region, and B indicates the total number of pixels of the input image. The first term measures the detection accuracy improvement. The second term indicates the zoom-in cost. The trade-off between accuracy and computation is controlled by the parameter λ. During training, the Q-net uses this reward function to calculate the immediate rewards of taking actions and learns a long-term reward function by Q learning [62].

2.2.2 Zoom-in accuracy gain regression network

The zoom-in accuracy gain regression network (R-net) predicts the accuracy gain of zooming in on a particular region based on the coarse detection results. The R-net is trained on pairs of coarse and fine detections so that it can observe how

they correlate with each other to learn a suitable accuracy gain.

Toward this end, we apply two pre-trained detectors to a set of training images and obtain two sets of image detection results: low-resolution detections $\{(\mathbf{d}_i^l, p_i^l, \mathbf{f}_i^l)\}$ in the down-sampled image and high-resolution detections $\{(\mathbf{d}_j^h, p_j^h)\}$ in the high resolution version of each image, where $\mathbf{d}$ is the detection bounding box, p is the probability of being the target object and $\mathbf{f}$ indicates a feature vector of the corresponding detection. We use the superscripts h and l to indicate the high resolution and low resolution (down-sampled) images. For the model to learn whether or not a high resolution detection improves the overall results, given a set of coarse detections at training time, we introduce a *match layer* which associates detections produced by the two detectors. In this layer, we pair the coarse and fine detection proposals and generate a set of correspondences between them. The object proposals i in the down-sampled image and j in the high-resolution image are defined as corresponding to each other if we find a j with sufficiently large intersection over union $IoU(d_i^l, d_j^h)$ with i ($IoU > 0.5$).

Given a set of correspondences, $\{(\mathbf{d}_k^l, p_k^l, p_k^h, \mathbf{f}_k^l)\}$, we estimate the zoom-in accuracy gain of a coarse detection. A detector can handle only objects within a range of sizes, so applying the detector to the high-resolution image does not always produce the best accuracy. For example, larger objects might be detected with higher accuracy at lower resolution if the detector was trained on mostly smaller objects. So, we measure which detection (coarse or fine) is closer to groundtruth using the metric $|g_k - p_k^l| - |g_k - p_k^h|$ where $g_k \in \{0, 1\}$ indicates the groundtruth label. When the high resolution score p_k^h is closer to the groundtruth than the low

resolution score p_k^l, the function indicates that this proposal is worth zooming in on. Otherwise, applying a detector on the down-sampled image is likely to yield a higher accuracy, so we should avoid zooming in on this proposal. We use a Correlation Regression (CR) layer to estimate the zoom-in accuracy gain of proposal k such that

$$\min_{\mathbf{W}}(|g_k - p_k^l| - |g_k - p_k^h| - \Phi(\mathbf{W}, f_k^l))^2 \ , \tag{2.2}$$

where Φ represents the regression function and $\mathbf{W}$ indicates the parameters. The output of this layer is the estimated accuracy gain. The CR layer contains two fully connected layers where the first layer has 4,096 units and the second one has only one output unit.

The AG map can be generated given the learned accuracy gain of each proposal. We assume that each pixel inside a proposal bounding box has equal contribution to its accuracy gain. Consequently, the AG map is generated as

$$AG(x,y) = \begin{cases} \alpha \frac{\Phi(\widehat{\mathbf{W}}, f_k^l)}{b_k} & if (x,y)\ in\ \mathbf{d}_k^l \\ 0 & otherwise \end{cases} \tag{2.3}$$

where $(x,y)\ in\ \mathbf{d}_k^l$ means point (x,y) is inside the bounding box $\mathbf{d}_k^l$ and b_k denotes the number of pixels included in $\mathbf{d}_k^l$. α is a constant number. $\widehat{\mathbf{W}}$ denotes the estimated parameters of the CR layer. The AG map is used as the state representation and it naturally contains the information of coarse detections' qualities. After zooming in and performing detection on a region, all the values inside the region are set 0 to prevent future zooming on the same region.

2.2.3 Zoom-in Q function learning network

The R-net provides information about which image region is likely to be the most informative if it is inspected next. Since the R-net is embedded within a sequential process, we use reinforcement learning to train a second network, the Q-net, to learn a long-term zoom-in reward function. At each step, the system takes an action by considering both immediate (Eq. 2.1) and future rewards. We formulate our problem in a Q learning framework, which approximates the long-term reward function for actions by learning a Q function. Based on the Bellman equation [63], the optimal Q function, $Q^*(s, a)$, obeys an important identity: given the current state, the optimal reward of taking an action equals the combination of its immediate reward and a discounted optimal reward at the next state triggered by this action Eq. 2.4

$$Q^*(s, a) = E_{s'}[R(s, a) + \gamma \max_{a'} Q^*(s', a')|s, a] \tag{2.4}$$

where s is the state and a is an action. Following [64], we learn the Q function for candidate actions by minimizing the loss function at the i-th iteration, *i.e.*,

$$L_i = (R(s, a) + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2 \tag{2.5}$$

where θ_i and θ_i^- are parameters of the Q network and those needed to calculate future reward at iteration i, respectively.

Eq. 2.5 implies that the optimal long-term reward can be learned iteratively if the immediate reward $R(s, a)$ is provided for a state-action pair. Since $R(s, a)$ is a cost-aware reward, the Q-net learns a long-term cost-aware reward function for the action set.

In practice, $\theta_i^- = \theta_{i-C}$ where C is a constant parameter. γ is future reward discount factor. We choose $C = 10$ and $\gamma = 0.5$ empirically in our experiments. We also adopt the ϵ-greedy policy [65] at training to balance between exploration and exploitation. The ϵ setting is the same as in [66].

The structure of our Q-net is shown in Fig. 2.2. The input is the AG map and each pixel in the map measures the predicted accuracy gain if the pixel at that location in the input image is included in the zoom region. The output is a set of maps and each value of a map measures the long-term reward of taking the corresponding action (selecting a zoom region at a location with a specified size). To allow the Q-net to choose zoomed-in regions with different sizes, we use multiple pipelines, each of which outputs a map corresponding to zoomed-in regions of a specific size. These pipelines share the same features extracted from the state representation. In the training phase, actions from all maps are concatenated to produce a unified action set and trained end-to-end together by minimizing the loss function in Eq. 2.5 so that all actions values compete with each other.

After zooming in on a selected region, we get both coarse and fine detections on the region. We just replace the coarse detections with fine ones in each zoom-in region.

Window selection refinement. The output of the Q-net can be directly

used as a zoom-in window. However, because candidate zoom windows are sparsely sampled, the window can be adjusted slightly to increase the expected reward. The Refine module takes the Q-net output as a coarse selection and locally moves the window towards a better location, as measured by the accuracy gain map by

$$\hat{a} = \arg\max_{a \in A} \sum_{(x,y)\ in\ a} AG(x, y) \tag{2.6}$$

where $\hat{a}$ selects the refined window and $A = (x_q \pm \mu_x, y_q \pm \mu_y, w, h)$ corresponds to the local refinement area controlled by parameter μ, where (x_q, y_q, w, h) indicates the output window of Q-net. We show a qualitative example of refinement in Fig. 2.3.

2.3 Experiments

We perform experiments on the Caltech Pedestrian Detection dataset (CPD) [59] and a Web Pedestrian dataset (WP) collected from YFCC100M [60]. Datasets like Pascal VOC [14] and MS COCO [16] are not chosen to validate our method, because they are not close to our scenario. In [14] and [16], there are generally very few objects per image and most objects are large, which leads to 1) close-to-zero rewards for regions, since large objects are likely to maintain high detection accuracy after reasonable down sampling; and, 2) large zoom-in windows in order to enclose large objects. Low region rewards discourage the window selection process and large zoom-in windows produce high cost, which make our method invalid.

Caltech Pedestrian Detection (CPD). There are different settings according to different annotation types, *i.e.*, *Overall*, *Near scale*, *Medium scale*, *No*

occlusion, *Partial occlusion* and *Reasonable* [59]. Similar to the *Reasonable* setting, we only train and test on pedestrians at least 50 pixels tall. We sparsely sample images (every 30 frames) from the training set. There are 4,321 images in the training set and 4,088 images in the test set. We rescale the images to 600 pixels on the shorter side to form the high resolution version of image during both training and testing. All of our model components are trained on this training set.

Web Pedestrian (WP) dataset. The image resolution in the CPD dataset is low (640×480). To better demonstrate our approach, we collect 100 test images with much higher resolution from the YFCC100M [60] dataset. The images are collected by searching for keywords "Pedestrian", "Campus" and "Plaza". An example is shown in Fig. 2.4 where pedestrians have varied sizes and are densely distributed in the images. For this dataset, we annotate all the pedestrians with at least 16-pixel width and less than 50% occlusion. Images are rescaled to 2,000 pixels on the longer side to fit for our GPU memory.

2.3.1 Baseline methods

We compare to the following baseline algorithms:

Fine-detection-all. This baseline directly applies the fine detector to the high resolution version of image. This method leads to high detection accuracy with high computational cost. All of the other approaches seek to maintain this detection accuracy with less computation.

Coarse-detection-all. This baseline applies the coarse detector on down-

sampled images with no zooming.

GS+Rnet. Given the initial state representation generated by the R-net, we use a greedy search strategy (GS) to densely search for the best window every time based on the current state without considering the long-term reward.

ER+Qnet. The entropy of the detector output (object vs no object) is another way to measure the quality of a coarse detection. [67] used entropy to measure the quality of a region for a classification task. Higher entropy implies lower quality of a coarse detection. So, if we ignore the correlation between fine and coarse detections, the accuracy gain of a region can also be computed as

$$-p_i^l log(p_i^l) - (1 - p_i^l) log(1 - p_i^l) \tag{2.7}$$

where p^l indicates the score of the coarse detection. For fair comparison, we fix all parameters of the pipeline except replacing the R-net output of a proposal with its entropy. **SSD and YOLOv2.** We also compare our method with off-the-shelf SSD [11] and YOLOv2 [13] trained on CPD, to show the advantage of our method on large images.

2.3.2 Variants of our framework

We use *Qnet-CNN* to represent the Q-net developed using a fully convolutional network (see Fig. 2.2). To analyze the contributions of different components to the performance gain, we evaluate three variants of our framework: Qnet*, Qnet-FC and Rnet*.

Figure 2.3: Effect of region refinement. Red boxes indicate zoom regions and the step number denotes the order that the zoom windows were selected. Before refinement, windows are likely to cut people in half due to the sampling grid, leading to a bad detection performance. Refinement locally adjusts the location of a window and produces better results.

Qnet*. This method uses a Q-net with refinement to locally adjust the zoom-in window selected by Q-net.

Qnet-FC. Following [66], we develop this variant with two fully connected (FC) layers for Q-net. For *Qnet-FC*, the state representation is resized to a vector of length $1,200$ as the input. The first layer has 128 units and the second layer has 34 units (9+25). Each output unit represents a sampled window on an image. We uniformly sample 25 windows of size 320×240 and 9 windows of size 214×160 on the CPD dataset. Since the output number of *Qnet-FC* can not be changed, windows sizes are proportionally increased when *Qnet-FC* is applied to WP dataset.

Rnet*. This is an R-net learned using a reward function that does not explicitly encode cost ($\lambda = 0$ in Eq. 2.1).

2.3.3 Evaluation metric

We use three metrics when comparing to the *Fine-detection-all* strategy: AP percentage (A_{perc}), processed pixel numbers percentage (P_{perc}), and average detection time percentage (T_{perc}). A_{perc} quantifies the percentage of AP we obtain compared to the *Fine-detection-all* strategy. P_{perc} and T_{perc} indicate the computational cost as a percentage of the *Fine-detection-all baseline* strategy.

2.3.4 Implementation details

We downsample the high resolution image by a factor of 2 to form a downsampled image for all of our experiments and only handle zoom-in regions at the high resolution.

For the Q-net, we spatially sample zoom-in candidate regions with two different window sizes (320×240 and 214×160) in a sliding window manner. For windows of size $W \times H$, we uniformly sample windows with horizontal stride $S_x = W/2$ and vertical stride $S_y = H/2$ pixels. For the refinement, we set $(\mu_x, \mu_y) = 0.5(S_x, S_y)$. The Q-net stops taking actions when the sum over all the values of the AG map is smaller than 0.1.

We use Faster R-CNN as our detector due to the success of R-CNN in many computer vision applications [68, 69, 70, 71, 72, 73]. Two Faster R-CNNs are trained on the CPD training set at the fine and coarse resolutions and used as black-box coarse and fine detectors afterwards. YOLOv2 and SSD are trained on the same training set with default parameter settings in the official codes released by the

authors. All experiments are conducted using a K-80 GPU.

2.3.5 Qualitative results

The qualitative comparisons, which show the effect of refinement on the selected zoom-in regions, are shown in Fig. 2.3. We observe that refinement significantly reduces the cases in which pedestrians only partly occur in the selected windows. Due to the sparse window sampling of Q-net, optimal regions might not be covered by any window candidate, especially when the window size is relatively small compared to the image size.

We show a comparison between our method (*Q-net*-CNN+Rnet*) and the greedy strategy (*GS+Rnet*) in Fig. 2.4. *GS* tends to select duplicate zooms on the same portion of the image. While the Q-net might select a sub-optimal window in the near term, it leads to better overall performance in the long term. As shown in the first example of Fig. 2.4, this helps Q-net terminate with fewer zooms.

Fig. 2.5 shows a qualitative comparison of R-net and *ER*. The examples in the first row are detections that do not need to be zoomed in on, since the coarse detections are good enough. R-net produces much lower accuracy gains for these regions. On the other hand, R-net outputs much higher gains in the second row which includes regions needing analysis at higher resolution. The third row contains examples which get worse results at higher resolution. As we mentioned before, entropy cannot determine if zooming in will help, while R-net produces negative gains for these cases and avoids zooming in on these regions.

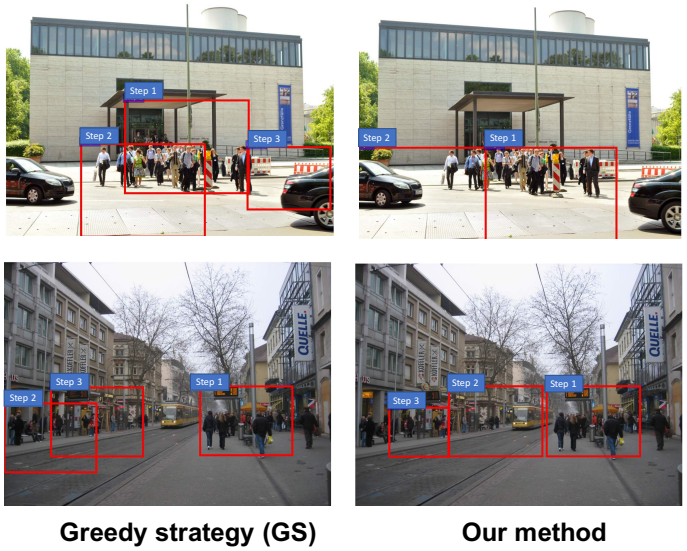


Figure 2.4: Qualitative comparison between using the Q-net* and a greedy strategy (GS) that selects the region with highest predicted accuracy gain at each step. Red bounding boxes indicate zoom-in windows and step number denotes the order of windows selection. The Q-net selects regions that appear sub-optimal in the near term but better zoom sequences in the long term, which leads to fewer steps as shown in the first row.

2.3.6 Quantitative evaluation

Table 2.1 shows the average precision (AP) and average detection time per image for *Fine-detection-all* and *Coarse-detection-all* strategies on CPD and WP datasets. The coarse baseline maintains only about 65% and 71% AP on CPD and WP, respectively, suggesting that the naive downsamping method significantly decreases detection accuracy.

Dataset	AP_f	AP_c	DT_f(ms)	DT_c(ms)
CPD	0.493	0.322	304	123
WP	0.407	0.289	1375	427

Table 2.1: *Coarse-detection-all*(with subscript c) v.s. *Fine-detection-all* (with subscript f) on CPD and WP datasets. DT indicates average detection time per image.

Comparative results on the CPD and WP dataset are shown in Table 2.2.

Figure 2.5: Qualitative comparison of R-net and ER on the Caltech Pedestrians test set. The first row of numbers indicate probability of the red box being a pedestrian. C denotes coarse detection and F indicates fine detection. Red font denotes the accuracy gain of R-net and blue is for ER. Positive and negative values are normalized to [0, 1] and [-1, 0). Compared to ER, R-net gives lower positive scores (row #1)/ negative scores (row #3) for regions that coarse detections are good enough/ better than fine detections and it produces higher scores for regions (row #2) where fine detections are much better than coarse ones.

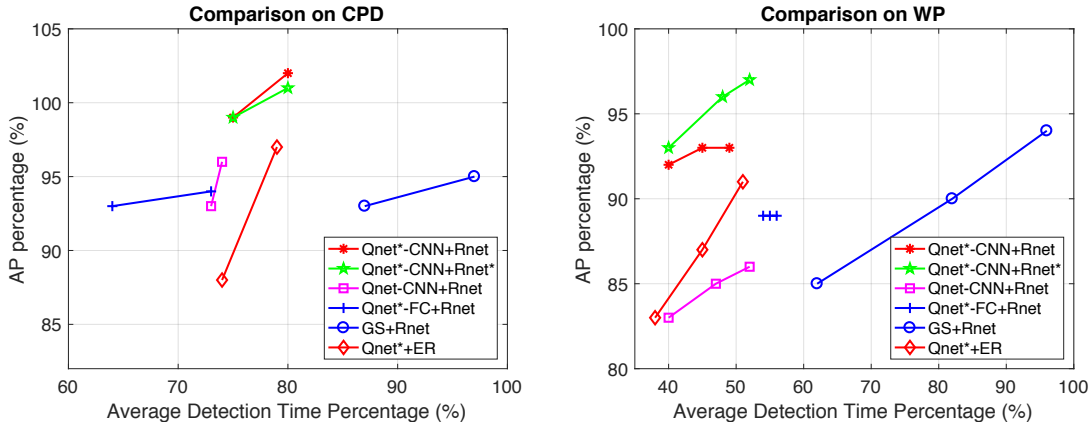


Figure 2.6: Detection time and accuracy comparison on the CPD/WP dataset after zooming in on two/three regions.

		Baselines		Variants under our framework			
	P_{perc}	GS+Rnet	Qnet*-CNN+ER	Qnet*-CNN+Rnet	Qnet*-CNN+Rnet*	Qnet*-FC+Rnet	Qnet-CNN+Rnet
CPD	≤40%	65%(40%)	88%(74%)	**99%**(75%)	65%(40%)	93%(64%)	65%(40%)
	≤45%	93%(87%)	97%(79%)	**102%**(80%)	101%(80%)	94%(73%)	96%(73%)
	≤50%	95%(97%)	97%(79%)	**102%**(80%)	101%(80%)	94%(73%)	96%(73%)
WP	≤30%	85%(62%)	83%(38%)	92%(40%)	**93%**(40%)	71%(31%)	83%(40%)
	≤35%	90%(82%)	91%(51%)	93%(45%)	**96%**(48%)	71%(31%)	85%(47%)
	≤40%	94%(96%)	91%(51%)	93%(49%)	**97%**(52%)	89%(54%)	86%(52%)

Table 2.2: Detection accuracy comparisons in terms of A_{perc} on the CPD and WP datasets under a fixed range of processed pixel percentage (P_{perc}). Bold font indicates the best result. Numbers are display as $A_{perc}(T_{perc})$- T_{perc} is included in the parentheses for the reference of running time. Note that 25% P_{perc} overhead is incurred simply by analyzing the down-sampled image (this overhead is included in the table) and percentages are relative to *Fine-detection-all* baseline (an A_{perc} of 80% means that an approach reached 80% of the AP reached by the baseline).

Q-net-CNN + R-net* reduces processed pixels by over 50% with comparable (or even better) detection accuracy than the *Fine-detection-all* strategy and improves detection accuracy of *Coarse-detection-all* by about 35% on the CPD dataset. On the WP dataset, the best variant (*Q-net*-CNN + R-net**)) reduces processed pixels by over 60% while maintaining 97% detection accuracy of *Fine-detection-all*. Table 2.2 shows that variants of our framework outperform *GS+Rnet* and *Qnet+ER* in most cases which suggests that *Qnet* and *Rnet* are better than *GS* and *ER*. Q-net is better than *GS* since the greedy strategy considers individual actions separately, while Q-net utilizes a RL framework to maximize the long term reward.

	CPD		**WP**	
	AP	DT(ms)	AP	DT(ms)
SSD500 [11]	0.405	128	0.255	570
SSD300 [11]	0.400	74	0.264	530
YOLOv2 [13]	0.398	70	0.261	790
Our method	**0.503**	243	**0.379**	619

Table 2.3: Comparison between Qnet*-CNN+Rnet and single-shot detectors trained on CPD. DT indicates average detection time per image. Bold font indicates the best result.

Qnet-CNN+Rnet* always produces better detection accuracy than *Qnet**-

CNN+ER under the same cost budget, which demonstrates that learning the accuracy gain using an R-net is preferable to using entropy, a hand-crafted measure. This could be due to two reasons: 1) entropy measures only the confidence of the coarse detector, while our R-net estimates the correlation with the high-resolution detector based on confidence and appearance; 2) according to the regression target function in Eq. 2.2, our R-net also measures whether the zoom-in process will improve detection accuracy. This avoids wasting resources on regions that cannot be improved (or might even be degraded) by fine detections.

We observe from Fig. 2.6 that our approach (*Qnet*-CNN+Rnet* and *Qnet*-CNN+Rnet**) reduces detection time by 50% while maintaining a high accuracy on the WP dataset. On the CPD dataset, they can reduce detection time by 25% without a significant drop of accuracy. Detection time cannot be reduced as much as on the WP dataset, since CPD images are relatively small; however, it is notable that our approach helps even in this case.

Table 2.3 shows accuracy/cost comparisons between YOLO/SSD and our method. Experiments suggest the following conclusions: 1) although fast, these single-shot detectors achieve much lower AP on images with objects occurring over a large range of scales; 2) as image size increases, YOLO/SSD processing time increases dramatically, while, our method achieves much higher accuracy with comparable detection time; 3) SSD consumes much more GPU memory than other detectors on large images due to the heavy convolution operations. We have to resize images of WP to 800×800 to fit within GPU memory. Note that it is possible to improve the results of YOLO/SSD by pruning the networks or training with more

data, but that is not within the scope of this paper.

2.3.7 Ablation analysis

Improvement by refinement (*Qnet*-CNN+Rnet* vs. *Qnet-CNN+Rnet*). In Table 2.2, we find that region refinement significantly improves detection accuracy under fixed cost ranges, especially on the WP. Refinement is more useful when zoom-in window size is relatively small compared with image size due to the sparse window sampling of Q-net. Fig. 2.3 qualitatively shows the effect of refinement.

Improvement by CNN (*Qnet*-CNN+Rnet* vs. *Qnet*-FC+Rnet*). FC has two obvious drawbacks in our setting. First, it has a fixed number of inputs and outputs which makes it hard to handle images with different sizes. Second, it is spatially dependent. Images from the CPD dataset consist of driving views which have strong spatial priors, *i.e.*, most pedestrians are on the sides of the street and the horizon is roughly in the same place. *Qnet-FC* takes advantage of these spatial priors, so it works better on this dataset. However, when it is applied to the WP dataset, its performance drops significantly compared to other methods, since the learned spatial priors now distract the detector.

Improvement by the cost term (*Qnet*-CNN+Rnet* vs. *Qnet*-CNN+Rnet**). *Qnet*-CNN+Rnet* outperforms *Qnet*-CNN+R-net** on CPD, especially when P_{perc} is low (40%). Without explicit cost penalization, the algorithm often selects the largest zoom regions, a poor strategy when there is a low pixel budget. However, since the window sizes are relatively small compared to the image size of the WP

dataset, *Qnet*-CNN+Rnet** does not suffer much from this limitation. On the contrary, it benefits from zooming in on relatively bigger regions. Consequently, it outperforms other variants. Nevertheless, *Qnet*-CNN+Rnet* has comparable detection accuracy and can generalize better on scenarios where window sizes are comparable with image size.

2.4 Conclusion

We propose a dynamic zoom-in network to speed up object detection in large images without manipulating the underlying detector's structure. Images are first downsampled and processed by the R-net to predict the accuracy gain of zooming in on a region. Then, the Q-net sequentially selects regions with high zoom-in reward to conduct fine detection. The experiments show that our method is effective on both Caltech Pedestrian Detection dataset and a high resolution pedestrian dataset.

Chapter 3: Temporal Recurrent Networks for Online Action Detection

3.1 Introduction

As we go about our lives, we continuously monitor the social environment around us, making inferences about the actions of others that might affect us. Is that child running into the road or just walking towards the sidewalk? Is that passerby outstretching his hand for a punch or a handshake? Is that oncoming car turning left or doing a U-turn? These and many other actions can occur at any time, without warning. In order to be able to react to the world around us, we must make and update our inferences in real-time, updating and refining our hypotheses moment-to-moment as we collect additional evidence over time.

In contrast, action recognition in computer vision is often studied as an offline classification problem, in which the goal is to identify a single action occurring in a short video clip given *all of its frames* [25, 26, 27, 28, 29, 30]. This offline formulation simplifies the problem considerably: a left turn can be trivially distinguished from a U-turn if the end of the action can be observed. But emerging real-world applications of computer vision like self-driving cars, interactive home virtual assistants, and

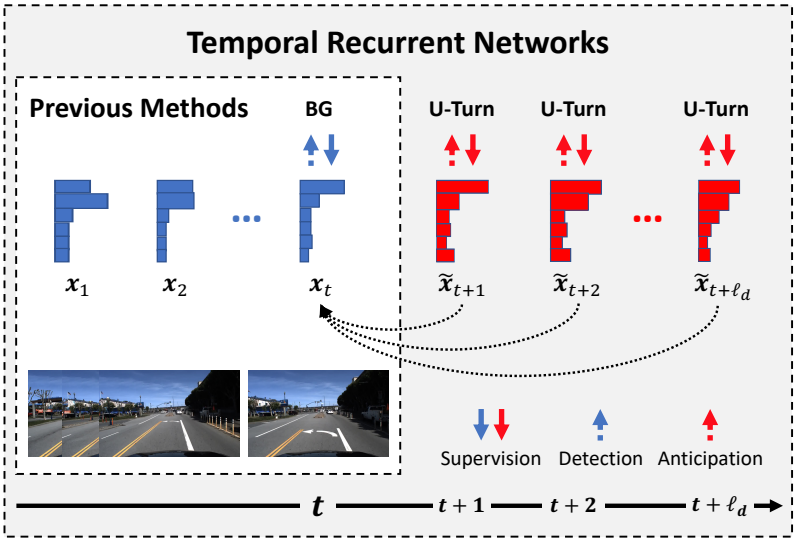


Figure 3.1: *Comparison between our proposed Temporal Recurrent Network (TRN) and previous methods.* Previous methods use only historical observations and learn representations for actions by optimizing current action estimation. Our approach learns a more discriminative representation by jointly optimizing current and future action recognition, and incorporates the *predicted* future information to improve the performance of action detection in the present.

collaborative robots require detecting actions online, in real-time. Several recent papers have considered this online action detection problem [1, 33, 34, 47, 74, 75], but accuracies are generally lower than the offline case because using only current and past information makes the problem much more challenging.

Here we introduce the novel hypothesis that although future information is not available in an online setting, *explicitly predicting the future can help to better classify actions in the present.* We propose a new model to estimate and use this future information, and we present experimental results showing that predicted future information indeed improves the performance of online action recognition. This may seem like a surprising result because at test time, a model that predicts the future to infer an action in the present observes exactly the same evidence as a

model that simply infers the action directly. However, results in cognitive science and neuroscience suggest that the human brain uses prediction of the future as an important mechanism for learning to make estimates of the present [76, 77, 78, 79]. Our findings seem to confirm that the same applies to automatic online action recognition, suggesting that jointly modeling current action detection and future action anticipation during training forces the network to learn a more discriminative representation.

In more detail, in this paper we propose a general framework called Temporal Recurrent Network (TRN), in which future information is predicted as an anticipation task and used together with historical evidence to recognize action in the current frame (as shown in Fig. 3.1). To demonstrate the effectiveness of our method, we validate TRN on two recent online action detection datasets (Honda Research Institute Driving Dataset (HDD) [80] and TVSeries [33]) and a widely used action recognition dataset, THUMOS'14 [81]. Our model is general enough to use both visual and non-visual sensor data, as we demonstrate for the HDD driving dataset. Experimental results show that our approach significantly outperforms baseline methods, especially when only a fraction of an action is observed. We also evaluate action anticipation (predicting the next action), showing that our method performs better than state-of-the-art methods even though anticipation is not the focus of this work.

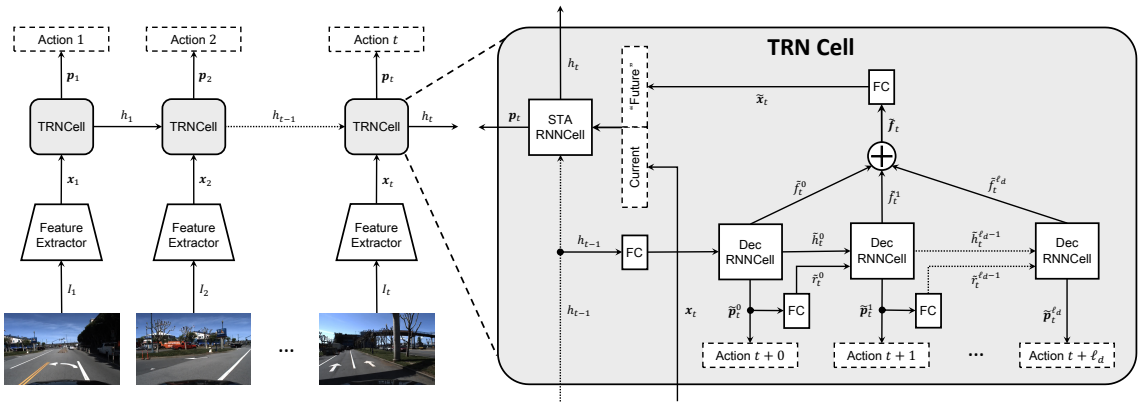


Figure 3.2: *Our proposed Temporal Recurrent Network (TRN)*, which sequentially processes input video frames and outputs frame-level action class probabilities, like any RNN. But while RNNs only model historical temporal dependencies, TRN anticipates the future via a temporal decoder, and incorporates that predicted information to improve online action detection.

3.2 Online Action Detection

Given a live video stream that contains one or more actions, our goal is to recognize actions of interest occurring in each video frame. Unlike most prior work that assumes the entire video is available at once, this *online action detection* problem requires us to process each frame as soon as it arrives, without accessing *any* future information. More formally, our goal is to estimate, for each frame I_t of an image sequence, a probability distribution $\mathbf{p}_t = [p_t^0, p_t^1, p_t^2, \cdots, p_t^K]$ over K possible actions, given only the past and current frames, $\{I_1, I_2, \cdots, I_t\}$ (where p_t^0 denotes the "background" probability that no action is occurring).

3.2.1 Temporal Recurrent Network (TRN)

To solve this problem, we introduce a novel framework called a Temporal Recurrent Network (TRN). The main idea is to train a network that predicts actions

several frames into the future, and then uses that prediction to classify an action in the present. Fig. 3.2 shows the architecture of TRN. The core of the network is a powerful recurrent unit, the *TRN cell*. Like a general RNN cell, at each time t a TRN cell receives a feature vector $\mathbf{x}_t$ corresponding to the observation at time t, which could include some combination of evidence from the appearance or motion in frame I_t or even other sensor modalities collected at time t, and the hidden state h_{t-1} from the previous time step. The cell then outputs $\mathbf{p}_t$, a probability distribution estimating which action is happening in I_t. The hidden state h_t is then updated and used for estimating the next time step. But while a traditional RNN cell only models *prior* temporal dependencies by accumulating historical evidence of the input sequence, a TRN cell also takes advantage of the temporal correlations between current and future actions by anticipating upcoming actions and explicitly using these estimates to help recognize the present action.

3.2.2 TRN Cell

The TRN cell controls the flow of internal information by using a temporal decoder, a future gate, and a spatiotemporal accumulator (STA). We use LSTMs [82] as the backbone for both the temporal decoder and the STA in our implementation, although other temporal models such as gated recurrent units (GRUs) [83] and temporal convolutional networks (TCNs) [84] could be used. The temporal decoder learns a feature representation and predicts actions for the future sequence. The future gate receives a vector of hidden states from the decoder and embeds

these features as the future context. The STA captures the spatiotemporal features from historical, current, and predicted future information, and estimates the action occurring in the current frame.

The temporal decoder works sequentially to output the estimates of future actions and their corresponding hidden states $\{\widetilde{h}_t^0, \widetilde{h}_t^1, \cdots, \widetilde{h}_t^{\ell_d}\}$ for the next ℓ_d time steps, where h_t^i for $i \in [0, \ell_d]$ indicates the hidden state at the i-th time step after t. The input to the decoder at the first time step is all zeros. At other time steps t, we feed in the predicted action scores $\widetilde{r}_t^{i-1}$, embedded by a linear transformer.

The future gate takes hidden states from the decoder and models the feature representation of future context. For simplicity, our default future gate is an average pooling operator followed by an fully-connected (FC) layer, but other fusion operations such as non-local (NL) blocks [85] could be used. More formally, the future context feature $\widetilde{\mathbf{x}}_t$ is obtained by averaging and embedding the hidden state vector, $\widetilde{\mathbf{h}}_t$, gathered from all decoder steps,

$$\widetilde{\mathbf{x}}_t = \mathrm{ReLU}(\mathbf{W}_f^T \mathrm{AvgPool}(\widetilde{\mathbf{h}}_t) + \mathbf{b}_f). \tag{3.1}$$

The spatiotemporal accumulator (STA) takes the previous hidden state h_{t-1} as well as the concatenation of the image feature $\mathbf{x}_t$ extracted from I_t and the predicted future feature $\widetilde{\mathbf{x}}_t$ from the future gate, and updates its hidden states h_t. It then calculates a distribution over candidate actions,

$$\mathbf{p}_t = \mathrm{softmax}(\mathbf{W}_c^T h_t + \mathbf{b}_c), \tag{3.2}$$

where $\mathbf{W}_c$ and $\mathbf{b}_c$ are the parameters of the FC layer used for action classification.

As we can see, in addition to the estimated action of the current frame t, TRN outputs predicted actions for the next ℓ_d time steps. In order to ensure a good future representation and jointly optimize online action detection and prediction, we combine the accumulator and decoder losses during training, *i.e.*, the loss of one input sequence is

$$\sum_t (loss(\mathbf{p}_t, l_t) + \alpha \sum_{i=0}^{\ell_d} loss(\widetilde{\mathbf{p}}_t^i, l_{t+i})), \tag{3.3}$$

where $\widetilde{\mathbf{p}}_t^i$ indicates the action probabilities predicted by the decoder for step i after time t, l_t represents the ground truth, *loss* denotes cross-entropy loss, and α is a scale factor. We optimize the network using offline training in which labels of both current and future frames are used. At test time, our model uses the *predicted* future information *without accessing actual future frames*, and thus is an online model.

3.3 Experiments

We evaluated our online action detector against multiple state-of-the-art and baseline methods on three publicly-available datasets: HDD [80], TVSeries [33], and THUMOS'14 [81]. We chose these datasets because they include long, untrimmed videos from diverse perspectives and applications: HDD consists of on-road driving from a first-person (egocentric) view recorded by a front-facing dashboard camera, TVSeries was recorded from television and contains a variety of everyday activities, and THUMOS'14 is a popular dataset of sports-related actions.

3.3.1 Datasets

HDD[80] includes nearly 104 hours of 137 driving sessions in the San Francisco Bay Area. The dataset was collected from a vehicle with a front-facing camera, and includes frame-level annotations of 11 goal-oriented actions (*e.g.*, intersection passing, left turn, right turn, *etc.*). The dataset also includes readings from a variety of non-visual sensors collected by the instrumented vehicle's Controller Area Network (CAN) bus. We followed prior work [80] and used 100 sessions for training and 37 sessions for testing.

TVSeries [33] contains 27 episodes of 6 popular TV series, totaling 16 hours of video. The dataset is temporally annotated at the frame level with 30 realistic, everyday actions (*e.g.*, pick up, open door, drink, *etc.*). The dataset is challenging with diverse actions, multiple actors, unconstrained viewpoints, heavy occlusions, and a large proportion of non-action frames.

THUMOS'14 [81] includes over 20 hours of sports video annotated with 20 actions. The training set contains only trimmed videos that cannot be used to train temporal action detection models, so we followed prior work [34] and train on the validation set (200 untrimmed videos) and evaluate on the test set (213 untrimmed videos).

3.3.2 Implementation Details

We implemented our proposed Temporal Recurrent Network (TRN) in PyTorch [86], and performed all experiments on a system with Nvidia Quadro P6000

graphics cards. To learn the network weights, we used the Adam [87] optimizer with default parameters, learning rate 0.0005, and weight decay 0.0005. For data augmentation, we randomly chopped off $\Delta \in [1, \ell_e]$ frames from the beginning for *each* epoch, and discretized the video of length L into $(L - \Delta)/\ell_e$ non-overlapping training samples, each with ℓ_e consecutive frames. Our models were optimized in an end-to-end manner using a batch size of 32, each with ℓ_e input sequence length. The constant α in Eq. (3.3) was set to 1.0.

3.3.3 Settings

To permit fair comparisons with the state-of-the-art [33, 34, 80], we follow their experimental settings, including input features and hyperparameters.

HDD. We use the same setting as in [80]. Video frames and values from CAN bus sensors are first sampled at 3 frames per second (fps). The outputs of the `Conv2d_7b_1x1` layer in InceptionResnet-V2 [88] pretrained on ImageNet [89] are extracted as the visual feature for each frame. To preserve spatial information, we apply an additional 1×1 convolution to reduce the extracted frame features from $8 \times 8 \times 1536$ to $8 \times 8 \times 20$, and flatten them into 1200-dimensional vectors. Raw sensor values are passed into a fully-connected layer with 20-dimensional outputs. These visual and sensor features are then concatenated as a *multimodal* representation for each video frame. We follow [80] and set the input sequence length ℓ_e to 90. The number of decoder steps ℓ_d is treated as a hyperparameter that we cross-validate in experiments. The hidden units of both the temporal decoder and the STA are set

to 2000 dimensions.

TVSeries and THUMOS'14. We use the same setting as in [34]. We extract video frames at 24 fps and set the video chunk size to 6. Decisions are made at the chunk level, and thus performance is evaluated every 0.25 seconds. We use two different feature extractors, VGG-16 [90] and two-stream (TS) CNN [91]. VGG-16 features are extracted at the `fc6` layer from the central frame of each chunk. For the two-stream features, the appearance features are extracted at the `Flatten_673` layer of ResNet-200 [92] from the central frame of each chunk, and the motion features are extracted at the `global_pool` layer of BN-Inception [93] from precomputed optical flow fields between 6 consecutive frames. The appearance and motion features are then concatenated to construct the two-stream features. The input sequence length ℓ_e is set to 64 due to GPU memory limitations. Following the state-of-the-art [34], the number of decoder steps ℓ_d is set to 8, corresponding to 2 seconds. As with HDD, our experiments report results with different decoder steps. The hidden units of both the temporal decoder and the STA are set to 4096 dimensions.

3.3.4 Evaluation Protocols

We follow most existing work and use per-frame **mean average precision (mAP)** to evaluate the performance of online action detection. We also use per-frame **calibrated average precision (cAP)**, which was proposed in [33] to better

evaluate online action detection on TVSeries,

$$cAP = \frac{\sum_k cPrec(k) * I(k)}{P}, \tag{3.4}$$

where calibrated precision $cPrec = \frac{TP}{TP+FP/w}$, $I(k)$ is 1 if frame k is a true positive, P denotes the total number of true positives, and w is the ratio between negative and positive frames. The advantage of cAP is that it corrects for class imbalance between positive and negative samples.

Another important goal of online action detection is to recognize actions as early as possible; *i.e.*, an approach should be rewarded if it produces high scores for target actions at their early stages (the earlier the better). To investigate our performance at different time stages, we follow [33] and compute mAP or cAP for each decile (ten-percent interval) of the video frames separately.

3.3.5 Baselines

CNN baseline models [90, 94] consider online action detection as a general image classification problem. These baselines identify the action in each individual video frame without modeling temporal information. For TVSeries and THUMOS'14, we reprint the results of CNN-based methods from De Geest *et al.* [33] and Shou *et al.* [32]. For HDD, we follow Ramanishka *et al.* [80] and use InceptionResnet-V2 [88] pretrained on ImageNet as the backbone and finetune the last fully-connected layer with softmax to estimate class probabilities.

LSTM and variants have been widely used in action detection [80, 95]. LSTM

networks model the dependencies between consecutive frames and jointly capture spatial and temporal information of the video sequence. For each frame, the LSTM receives the image features and the previous hidden state as inputs, and outputs a probability distribution over candidate actions.

Encoder-Decoder (ED) architectures [96] also model temporal dependencies. The encoder is similar to a general LSTM and summarizes historical visual information into a feature vector. The decoder is also an LSTM that produces predicted representations for the future sequence based only on these encoded features. Since there are no published results of ED-based methods on HDD, we implemented a baseline with the same experimental settings as TRN, including input features, hyperparameters, loss function, *etc.*.

Stronger Baselines. In addition to the above basic baselines, we tested three types of stronger baselines that were designed for online action detection on TVSeries and THUMOS'14. **Convolutional-De-Convolutional (CDC)** [32] places CDC filters on top of a 3D CNN and integrates two reverse operations, spatial downsampling and temporal upsampling, to precisely predict actions at a frame-level. Note that CDC is an offline method, and comparing with CDC confirms the effectiveness of our model. **Two-Stream Feedback Network (2S-FN)** [74] is built on an LSTM with two recurrent units, where one stream focuses on the input interpretation and the other models temporal dependencies between actions. **Reinforced Encoder-Decoder (RED)** [34] with a dedicated reinforcement loss is an advanced version of ED, and currently performs the best among all the baselines for online action

detection.

3.3.6 Results

Method	Inputs	Individual actions											Overall mAP
		intersection passing	L turn	R turn	L lane change	R lane change	L lane branch	R lane branch	crosswalk passing	railroad passing	merge	u-turn	
CNN	Sensors	34.2	72.0	74.9	16.0	8.5	7.6	1.2	0.4	0.1	2.5	32.5	22.7
LSTM [80]		36.4	66.2	74.2	26.1	13.3	8.0	0.2	0.3	0.0	3.5	33.5	23.8
ED		43.9	73.9	75.7	31.8	15.2	15.1	2.1	0.5	0.1	4.1	39.1	27.4
TRN		46.5	75.2	77.7	35.9	19.7	18.5	3.8	0.7	0.1	2.5	40.3	**29.2**
CNN	InceptionResNet-V2	53.4	47.3	39.4	23.8	17.9	25.2	2.9	4.8	1.6	4.3	7.2	20.7
LSTM [80]		65.7	57.7	54.4	27.8	26.1	25.7	1.7	16.0	2.5	4.8	13.6	26.9
ED		63.1	54.2	55.1	28.3	35.9	27.6	8.5	7.1	0.3	4.2	14.6	27.2
TRN		63.5	57.0	57.3	28.4	37.8	31.8	10.5	11.0	0.5	3.5	25.4	**29.7**
CNN	Multimodal	73.7	73.2	73.3	25.7	24.0	27.6	4.2	4.0	2.8	4.7	30.6	31.3
LSTM [80]		76.6	76.1	77.4	41.9	23.0	25.4	1.0	11.8	3.3	4.9	17.6	32.7
ED		77.2	74.0	77.1	44.6	41.4	36.6	4.1	11.4	2.2	5.1	43.1	37.8
TRN		79.0	77.0	76.6	45.9	43.6	46.9	7.5	13.4	4.5	5.8	49.6	**40.8**

Table 3.1: *Results of online action detection on HDD,* comparing TRN and baselines using mAP (%).

Method	Inputs	mcAP
CNN [33]	VGG	60.8
LSTM [33]		64.1
RED [34]		71.2
Stacked LSTM [74]		71.4
2S-FN [74]		72.4
TRN		**75.4**
SVM [33]	FV	74.3
RED [34]	TS	79.2
TRN		**83.7**

Table 3.2: *Results of online action detection on TVSeries,* comparing TRN and the state-of-the-art using cAP (%).

3.3.6.1 Evaluation of Online Action Detection

Table 3.1 presents evaluation results on HDD. TRN significantly outperforms the state-of-the-art, Ramanishka *et al.* [80], by 5.4%, 2.8%, and 8.1% in terms of

Method	mAP
Single-frame CNN [90]	34.7
Two-stream CNN [94]	36.2
C3D + LinearInterp [32]	37.0
Predictive-corrective [97]	38.9
LSTM [98]	39.3
MultiLSTM [95]	41.3
Conv & De-conv [32]	41.7
CDC [32]	44.4
RED [34]	45.3
TRN	**47.2**

Table 3.3: *Results of online action detection on THUMOS'14,* comparing TRN and the state-of-the-art using mAP (%).

mAP with sensor data, InceptionResnet-v2, and multimodal features as inputs, respectively. Interestingly, the performance gaps between TRN and [80] are much larger when the input contains sensor data. Driving behaviors are highly related to CAN bus signals, such as steering angle, yaw rate, velocity, *etc.*, and this result suggests that TRN can better take advantage of these useful input cues. Table 3.2 presents comparisons between TRN and baselines on TVSeries. TRN significantly outperforms the state-of-the-art using VGG (mcAP of 3.0% over 2S-FN [74]) and two-stream input features (mcAP of 4.5% over RED [34]). We also evaluated TRN on THUMOS'14 in Table 3.3. The results show that TRN outperforms all the baseline models (mAP of 1.9% over RED [34] and 2.8% over CDC [32]).

3.3.6.2 Ablation Studies

Importance of Temporal Context. By directly comparing evaluation results of TRN with CNN and LSTM baselines, we demonstrate the importance of explicitly modeling temporal context for online action detection. LSTMs capture long-

and short-term temporal patterns in the video by receiving accumulated historical observations as input. Comparing TRN and LSTM measures the benefit of incorporating predicted action features as future context. CNN-based methods conduct online action detection by only considering the image features at each time step. Simonyan *et al.* [94] build a two-stream network and incorporate motion features between adjacent video frames by using optical flow as input. Table 3.3 shows that this motion information yields a 1.5% improvement. *TRN-TS* also takes optical flow as input and we can clearly see a significant improvement (83.7% vs. 75.4%) on TVSeries.

Future Context: An "Oracle" Study. To demonstrate the importance of using predictions of future context, we implemented an oracle baseline, *RNN-offline*. RNN-offline shares the same architecture as RNN but uses the features extracted from both the current and future frames as inputs. Note that RNN-offline uses future information and thus is not an online model; our goal is to quantify (1) the effectiveness of incorporating future information in action detection, given access to actual (instead of predicted) future information, and (2) the performance gap between estimated future information of TRN and "real" future information of RNN-offline. To permit fair comparison, the input to RNN-offline is a concatenation of the feature from the current frame and the average-pooled features of the next ℓ_d frames (where ℓ_d is the same as the number of decoder steps of TRN).

The results of RNN-offline are 41.6%, 85.3%, and 47.3% on HDD, TVSeries, and THUMOS'14 datasets, respectively. Comparing RNN-offline with the RNN

Method	Inputs	Portion of video									
		0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%	80%-90%	90%-100%
CNN [33]	VGG	61.0	61.0	61.2	61.1	61.2	61.2	61.3	61.5	61.4	61.5
LSTM [33]	VGG	63.3	64.5	64.5	64.3	65.0	64.7	64.4	64.3	64.4	64.3
TRN	VGG	73.9	74.3	74.7	74.7	75.1	75.1	75.3	75.2	75.2	75.3
SVM [33]	FV	67.0	68.4	69.9	71.3	73.0	74.0	75.0	76.4	76.5	76.8
TRN	TS	78.8	79.6	80.4	81.0	81.6	81.9	82.3	82.7	82.9	83.3

Table 3.4: *Online action detection results when only portions of videos are considered* in terms of cAP (%) on TVSeries.

Datasets	Models	0.25s	0.5s	0.75s	1.0s	1.25s	1.5s	1.75s	2.0s	Mean
TVSeries	ED [34]	78.5	78.0	76.3	74.6	73.7	72.7	71.7	71.0	74.5
	RED [34]	79.2	78.7	77.1	75.5	74.2	73.0	72.0	71.2	75.1
	TRN	79.9	78.4	77.1	75.9	74.9	73.9	73.0	72.3	**75.7**
THUMOS'14	ED [34]	43.8	40.9	38.7	36.8	34.6	33.9	32.5	31.6	36.6
	RED [34]	45.3	42.1	39.6	37.5	35.8	34.4	33.2	32.1	37.5
	TRN	45.1	42.4	40.7	39.1	37.7	36.4	35.3	34.3	**38.9**

Table 3.5: Action anticipation results of TRN compared to published state-of-the-art methods in terms of per-frame *cAP* (%) and *mAP* (%) on TVSeries and THUMOS'14 datasets. Two-stream input features are used in all the models.

baseline, we see that the "ground-truth" future information significantly improves detection performance. We also observe that the performance of TRN and RNN-offline are comparable, *even though TRN uses predicted rather than actual future information*. This may be because TRN improves its representation during learning by jointly optimizing current and future action recognition, while RNN-offline does not. We also evaluated TRN against ED-based networks, by observing that ED can also improve its representation by jointly conducting action detection and anticipation. Thus, comparisons between TRN with ED and its advanced version [34] measure how much benefit comes purely from explicitly incorporating anticipated future information.

Effect of Decoder Step Count. Finally, we evaluated the effectiveness of dif-

Dataset	Task	Decoder steps (ℓ_d)			
		4	6	8	10
HDD	Online Action Detection	39.9	40.8	40.1	39.6
	Action Anticipation	34.3	32.2	28.8	25.4
TVSeries	Online Action Detection	83.5	83.4	83.7	83.5
	Action Anticipation	77.7	76.4	75.7	74.1
THUMOS'14	Online Action Detection	46.0	45.4	47.2	46.4
	Action Anticipation	42.6	39.4	38.9	35.0

Table 3.6: *Online action detection and action anticipation results* of TRN with decoder steps $\ell_d = 4, 6, 8, 10$.

ferent decoder step counts, $\ell_d = \{4, 6, 8, 10\}$. Table 3.6 shows the results, with the performance of action anticipation averaged over the decoder steps. The results show that a larger number of decoder steps does not guarantee better performance. This is because anticipation accuracy usually decreases for longer future sequences, and thus creates more noise in the input features of STA. To be clear, we follow the state-of-the-art [34] and set ℓ_d to 2 video seconds (6 frames in HDD, 8 frames in TVSeries and THUMOS'14) when comparing with baseline methods of online action detection in Tables 3.1, 3.2, and 3.3.

3.3.6.3 Evaluation of Different Stages of Action

We evaluated TRN when only a fraction of each action is considered, and compared with published results [33] on TVSeries. For example, 20%-30% means only frames in the 20%-30% time range of action sequences were evaluated. Table 3.4 shows that TRN significantly outperforms existing methods at every time stage. Specifically, when we compare *TRN-TS* with the best baseline *SVM-FV*, the performance gaps between these two methods are roughly in ascending order as less

and less of the actions are observed (the gaps are 6.5%, 6.4%, 6.3%, 7.3%, 7.9%, 8.6%, 9.7%, 10.5%, 11.2% and 11.8% from actions at 100% observed to those are 10% observed). This indicates the advantage of our approach at earlier stages of actions.

3.3.6.4 Evaluation of Action Anticipation

We also evaluated TRN on predicting actions for up to 2 seconds into the future, and compare our approach with the state-of-the-art [34] in Table 3.5. The results show that TRN performs better than RED and ED baselines (mcAP of 75.7% vs. 75.1% vs. 74.5% on TVSeries and mAP of 38.9% vs. 37.5% vs. 36.6% on THUMOS'14). The average of anticipation results over the next 2 seconds on HDD is 32.2% in terms of per-frame mAP.

3.4 Conclusion

We propose Temporal Recurrent Networks (TRNs) to model greater temporal context, and we evaluate them on the online action detection problem. Unlike previous methods that consider only historical temporal consistencies, TRN jointly models the historical and future temporal context under the constraint of an online setting. Experimental results on three popular datasets demonstrate that incorporating predicted future information improves learned representation of actions and significantly outperforms the state-of-the-art. Moreover, TRN shows greater advantage at earlier stages of actions and in predicting future actions. More generally,

we believe that our approach of incorporating estimated future information could benefit many other online tasks, such as video object localization and tracking, and plan to pursue this in future work.

Chapter 4: StartNet: Online Detection of Action Start in Untrimmed Videos

4.1 Introduction

Temporal action localization (TAL) in untrimmed videos has been widely studied in offline settings, where start and end times of an action are recognized after the action is fully observed [25, 26, 27, 28, 29, 30]. With the emerging applications that require identifying actions in real time, *e.g.*, autonomous driving, surveillance system, and collaborative robots, online action detection (OAD) methods [1, 33, 34, 46] have been proposed. They typically pose the TAL problem as a per-frame class labeling task.

However, in some time-sensitive scenarios, detecting accurate action starts in a timely manner is more important than successfully detecting every frame containing actions. For example, an autonomous driving car needs to detect the start of "pedestrian crossing" as soon as it happens to avoid collision; a surveillance system should generate alert as soon as a dangerous event is initiated. Online Detection of Action Start (ODAS) was proposed to address this problem specifically [1]. Instead of classifying every frame, ODAS detects the occurrence and category of an action

start as soon as possible. Thus, it addresses two sub-tasks: (i) if an action starts at time t and (ii) its associated action class.

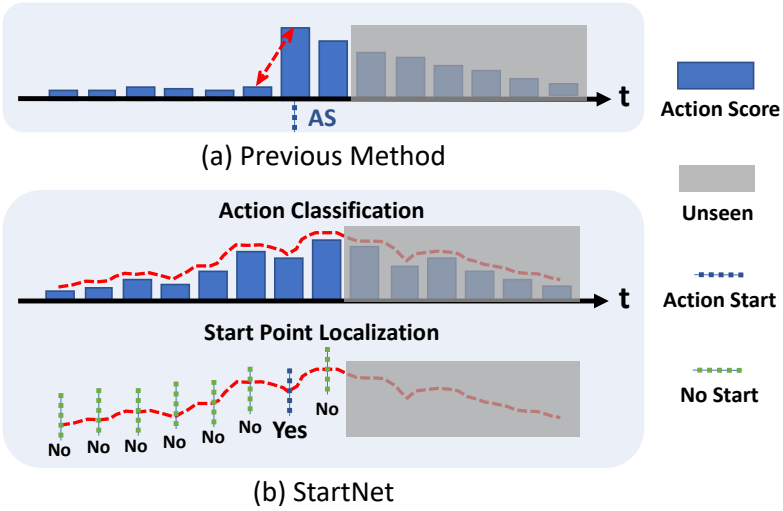


Figure 4.1: Comparison between (a) the previous method [1] and (b) the proposed framework. [1] aims to generate an action score sequence which produces low score for background and high score for the correct action immediately when the action starts. We propose a two-stage framework: the first stage only focuses on per-frame action classification and the second stage learns to localize the start points given the historical trend of the action scores generated by the first stage.

The existing method [1] handles the two sub-tasks jointly by training a classification network that is capable of localizing the starts of different action classes. The network attempts to make the representation of a start point close to that of its associated action class and far from its preceding background. As shown in Fig. 4.1 (a), the network is encouraged to react immediately when an action starts. However, it is hard to achieve this goal due to the subtle appearance difference near start points and the lack of labeled training data (one action only contains one start point).

Our method is inspired by three key insights. First, decomposing a complex

task properly allows sub-modules to focus on their own sub-tasks and makes the learning process easier. A good example is the success of the two-stage object detection framework [2, 8, 99]. Second, as mentioned in [99], when training data is scarce, learning from a representation that is pre-trained on an auxiliary task may lead to a significant performance boost. Third, OAD (per-frame labeling) is very related to ODAS. Comparing to the scarce labeled data of action starts, the amount of per-frame action labels is much larger. Thus, there may be potential benefits if we take advantage of the per-frame labeling task.

Instead of focusing on learning subtle difference near start points, we propose an alternative framework, *i.e.* startNet, and address ODAS in two stages: classification (using ClsNet) and localization (using LocNet). ClsNet conducts per-frame labeling as an auxiliary task based on the spatial-temporal feature aggregation from input videos, and generates score distributions of action classes as a high-level representation. Based on the historical trend of score distributions, LocNet predicts class-agnostic start probability at each time (see Fig 4.1 (b)). At the end, late fusion is applied on the outputs of both modules to generate the final result. When designing LocNet, we consider the implicit temporal constraint between action starts – two start point are unlikely to be close by. To impose the temporal constraint into the framework under the online setting, historical decisions are taken into account for later predictions. To optimize the long-term reward for start detection, LocNet is trained using reinforcement learning techniques. The proposed framework and its variants are validated on THUMOS'14 [81] and ActivityNet [100]. Experimental results show that our approach significantly outperforms the state-of-the-art

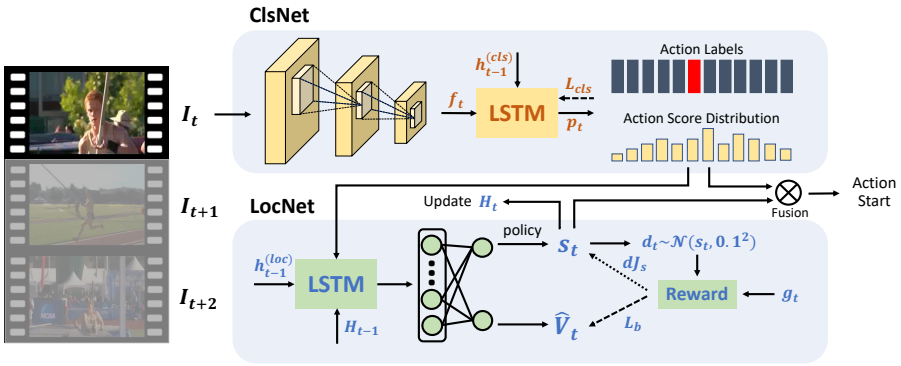


Figure 4.2: Our method works in two stages with ClsNet and LocNet. ClsNet: at time t, features, $\mathbf{f}_t$, are extracted by deep convolutional networks and input to an one-layer LSTM; The LSTM generates action score distributions at each time step and ClsNet is optimized with cross-entropy loss between action labels and the generated action scores. LocNet: after action score generation, it inputs together with a historical decision vector, $\mathbf{H}$, to a second one-layer LSTM which works as an agent to generate two-dimensional start probability sequentially; $\mathbf{H}$ is updated and the state is changed accordingly; The agent is trained using policy gradient mechanism to optimize long-term reward of start localization. At the end, results from ClsNet and LocNet are fused to obtain the final action start detection results at each time step. Here, ClsNet is implemented with LSTM. CNN and C3D can also be used to construct ClsNet (see Sec. 4.2.1 for details).

by 10%-30% p-mAP under offsets of 1-10 seconds on THUMOS'14, and achieves comparable p-mAP with 10 times smaller time offset on ActivityNet.

4.2 Action Start Detection Network (StartNet)

The input of an ODAS system is untrimmed, streaming video frames $\{I_1, I_2, ..., I_t\}$. The system processes each video frame sequentially and detects the start of each action instance. At time step t, it outputs a probability distribution, $\mathbf{as}_t^k$, which indicates the start probability of the action class k, without accessing any future information.

The overview of the proposed framework is illustrated in Fig. 4.2. The frame-

work contains two sub-networks, *i.e.*, a classification network (ClsNet) and a localization network (LocNet). ClsNet focuses on per-frame class labeling. It takes the raw video frames as input and outputs action class probabilities at every time step in an online manner. ClsNet serves two purposes. First, it learns simpler but useful representation for localizing action starts. Second, the classification results can be combined later with the localization results to produce the action starts for each class. LocNet takes the output of ClsNet together with the historical decision vector as inputs. At each time step, it outputs a two-dimensional probability distribution indicating the probability that this frame contains an action start. The historical decision vector records its predictions in the previous n steps in order to model the effect of historical decisions on later ones. Finally, the results of the two networks are fused to construct the final output.

4.2.1 Classification Network (ClsNet)

Inspired by recent online action detection methods [33, 34, 46], we utilize recurrent networks, specifically, LSTM [82], to construct ClsNet. At each time t, it uses the previous hidden state $\mathbf{h}_{t-1}^{(cls)}$, the cell $\mathbf{c}_{t-1}^{(cls)}$, and the feature, $\mathbf{f}_t$, extracted from the current video frame, I_t, as inputs, to update its hidden state $\mathbf{h}_t^{(cls)}$ and cell $\mathbf{c}_t^{(cls)}$. Then, the likelihood distribution over all the action classes can be obtained in Eq. 4.1,

$$\mathbf{p}_t = softmax(\mathbf{W}_{cls}^T \mathbf{h}_t^{(cls)} + \mathbf{b}), \tag{4.1}$$

where $\mathbf{p}_t$ is a K dimensional vector and K indicates the number of action classes including background.

To learn ClsNet, action class label for each frame is needed. The cross-entropy loss, $L_{cls}(\mathbf{W}_c)$, is used for optimization during training, where $\mathbf{W}_c$ represents the parameter set of ClsNet.

We observe that ClsNet can be implemented with different architectures. Thus, we validate our framework using two additional structures as the backbone of ClsNet, *i.e.*, CNN and C3D [101]. CNN conducts action classification based only on the arriving frame, I_t. It focuses on the spatial information of the current frame without considering temporal patterns of actions. C3D labels I_t based on each temporal segment consisting of 16 consecutive video frames, from I_{t-15} to I_t. It captures spatial and temporal information jointly using 3D convolutional operations. Comparisons and explanations are discussed in Sec. 4.3.

4.2.2 Localization Network (LocNet)

As discussed in Sec. 4.1, historical action scores can provide useful cues for identifying action starts. At time t, LocNet observes the action score distribution over classes of each frame, $\mathbf{p}_t$, obtained from ClsNet and outputs a two-dimensional vector, $\mathbf{s}_t$, indicating the start and non-start probability distribution.

The start probability is generated sequentially. In general, if an action starts at time step t, there is a low probability that another action also starts at time $t+1$, given reasonable frames per second (FPS). Thus, there are implicit temporal con-

straints between nearby start points. To enable the model to consider constraints between decisions, we record the historical decisions made by LocNet and use the history to influence later decisions. To enable long-term decision planning, we formulate the problem as a Markov Decision Process (MDP) and use reinforcement learning to optimize our model. When making a decision[1], the model not only considers the effect of the decision at the current step, but also how it will influence the later ones by maximizing the expected long-term reward. In the following, we first discuss the inference phase of LocNet and then the training phase in detail.

Inference Phase. LocNet is built upon a LSTM structure. It acts as an agent which interacts with historical action scores recurrently. During testing, at each state, the agent makes a decision (predicts start probability) that produces the maximum expected long-term reward and updates the state according to the decision. To model the dependency between decisions, we incorporate the record of historical decisions (the decisions made by the agent at previous steps) as a part of the state. The state update procedure is described in Eq. 4.2 and 4.3, where $\mathbf{H}_{t-1} = \mathbf{s}_{t-n:t-1}$ indicates historical decisions from step $t-n$ to $t-1$ and $[\mathbf{p}_t, \mathbf{H}_{t-1}]$ indicates the concatenation of the vectors. At the beginning, $\mathbf{H}$ is initialized with zeros.

$$\mathbf{h}_t^{(loc)}, \mathbf{c}_t^{(loc)} = LSTM(\mathbf{h}_{t-1}^{(loc)}, \mathbf{c}_{t-1}^{(loc)}, [\mathbf{p}_t, \mathbf{H}_{t-1}]). \tag{4.2}$$

$$\mathbf{s}_t = softmax(\mathbf{W}_{loc}^T \mathbf{h}_t^{(loc)} + \mathbf{b}). \tag{4.3}$$

[1]The term "action" is generally used in reinforcement learning, we use "decision" instead to remove the confusion with action class.

Training Phase. We train an agent that acts optimally based on the state of the environment. The goal is to maximize the reward by changing the predicted start probability distribution: at a given state, the start probability should be increased when the decision introduces bigger reward and be decreased otherwise. The start prediction procedure is formulated as a decision making policy defined using Gaussian distribution. Following [102, 103], the policy is trained by optimizing with d_t, where d_t, is sampled from $\pi(.|\mathbf{h}_t^{(loc)}, \mathbf{p}_t, \mathbf{H}_{t-1}) = \mathcal{N}(s_t, 0.1^2)$ and s_t indicates the output start probability.

Reward function. Each decision at a given state is associated with an immediate reward to measure the decision made by the agent at the current time. With the goal of localizing start points, we define the immediate reward function in Eq. 4.4, where $g_t \in \{0, 1\}$ indicates the ground-truth label of action start and d_t is the sampled start probability. The reward function encourages a high probability when there is an actual start and a low probability when there is not by giving a negative reward. Considering the sample imbalance between start points and background, weighted rewards are used by setting a parameter α. In particular, we set α to be the ratio between the number of negative samples to positive samples for each dataset.

$$r_t = \alpha g_t d_t - (1 - g_t) d_t. \tag{4.4}$$

The long-term reward is the summation of discounted future rewards. In order to maximize the expected long-term reward, the policy is trained by maximizing the objective in Eq. 4.5, where $\mathbf{W}_s$ represents the parameters of the network and γ is a

constant scalar for calculating the discounted rewards over time.

$$J_s(\mathbf{W}_s) = \mathop{\mathbb{E}}_{d_t \sim \pi(.|\mathbf{W}_s)} [\sum_{i=0} \gamma^i r_{t+i}]. \tag{4.5}$$

Optimization. When optimizing Eq. 4.5, it is not possible to train the network using error back propagation directly, since the objective is not differentiable. Following [104], we use policy gradient to calculate the expected gradient of J_s as in Eq. 4.6, where $R_t = \sum_{i=0} \gamma^i r_{t+i}$ indicates the long-term reward at time step t and $\hat{V}_t$ is a baseline value (generated by an fully-connected (FC) layer as shown in Fig. 4.2) which is widely used in policy gradient frameworks to reduce the variance of the gradient. The principle of policy gradient is to maximize the probability of an action with high reward given a state.

$$\nabla_{\mathbf{W}_s} J_s = \mathbb{E}[\sum_{t=0}^{\infty} (R_t - \hat{V}_t) \nabla_{\mathbf{W}_s} log\pi(.|\mathbf{W}_s)]. \tag{4.6}$$

Following [103], we use the expected long-term reward at the current state as the baseline value and approximate it by minimizing the l_2 loss: $L_b(\mathbf{W}_b) = \frac{1}{2}||R_t - \hat{V}_t||_2$. The training procedure of LocNet is summarized in Alg. 1.

The full objective including the loss term in ClsNet is shown in Eq. 4.7, where λ_1 and λ_2 are constant scalars.

$$\min L_{cls}(\mathbf{W}_c) + \lambda_1 L_b(\mathbf{W}_b) - \lambda_2 J_s(\mathbf{W}_s). \tag{4.7}$$

Algorithm 1: Training Process of LocNet

Initialize parameters, $\mathbf{W}_s$, of LocNet
for iteration = 1:N **do**
  Obtain training sequence samples of length T_{loc}
  for t = 1:T_{loc} **do**
    Obtain s_t based on current policy
    Sample decisions: $d_t \sim \mathcal{N}(s_t, 0.1^2)$
    Obtain r_t and $\hat{V}_t$ for each sample
  Compute $R_{1:T_{loc}}$, $\nabla_{\mathbf{W}_s} J_s$ and $L_b(\mathbf{W}_b)$
  Update parameters, $\mathbf{W}_s$, of LocNet

Late Fusion. ClsNet outputs an action score distribution and LocNet produces class-agnostic start probabilities at each time step. Then, late fusion is applied to obtain the start probability for each action class, $\mathbf{as}_t^k$, using Eq. 4.8, where superscript 1:K-1 indicates positive classes and 0 indicates background.

$$\mathbf{as}_t^k = \begin{cases} s_t \mathbf{p}_t^{1:K-1} & k = 1 : K-1 \\ (1 - s_t)\mathbf{p}_t^0 & k = 0 \end{cases} . \tag{4.8}$$

Action start generation. Follow [1], final action starts are generated online if all of the three conditions are satisfied: (i) $c_t = \text{argmax}_k(\mathbf{as}_t^k)$ is an action; (ii) $c_t \neq c_{t-1}$ and (iii) $\mathbf{as}_t^{c_t}$ exceeds a threshold. We set this threshold to 0 by default. An action score sequence generated by ClsNet can also generate action start points online following this procedure. LocNet can locally adjust the start point by boosting time points with higher start probabilities and suppressing those with lower start probabilities.

4.3 Experiments

To validate the proposed framework, we conduct extensive experiments on two large-scale action recognition datasets, *i.e.*, THUMOS'14 [81] and ActivityNet v1.3 [100].

Evaluation protocol. To permit fair comparisons, we use the point-level average precision (p-AP) proposed in [1] to evaluate our framework. Under this protocol, each action start prediction is associated with a time point. For each action class, predictions of all frames are first sorted in descending order based on their confidence scores and then measured accordingly. An action start prediction is counted as correct only if it matches the correct action class and its temporal distance from a ground-truth point is smaller than an offset threshold (offset tolerance). Similar to segment-level average precision, no duplicate detections are allowed for the same ground-truth point. p-mAP is then calculated by averaging p-AP over all the action classes.

Following [1], we use two metrics based on p-AP to evaluate our framework on THUMOS'14. First, we use p-AP under different offset tolerances, varying from 1 to 10 seconds. Also, we adopt the metric *AP depth at recall (Rec) X*% which averages p-AP on the Precision-Recall curve with the recall rate from 0% to X%. p-mAPs under different offset thresholds are then averaged to obtain the final average p-mAP at each depth. This metric is particularly used to evaluate top ranked predictions and to measure what precision a system can achieve if low recall is allowed. For

ActivityNet, we evaluate our methods using p-mAP under offset thresholds of 1-10 seconds at depth Rec=1.0.

Baselines. We compare our framework with the state-of-the-art method, *i.e.*, *Shou et al.* [1] and two baselines that were presented in [1], *i.e.*, *SceneDetect* and *ShotDetect*. The numbers were obtained from the authors [1]. Comparison results with *Shou et al.* [1] demonstrate the superior performance of StartNet. *SceneDetect* and *ShotDetect* are also two-stage methods. Similar to two-stage frameworks of object detection, they first conduct localization by getting action start proposals, which are generated by soft boundary detectors, and then classify them to different classes. Comparison with *SceneDetect* and *ShotDetect* shows the effectiveness of our decomposition design. Our framework trained by policy gradient is indicated by *StartNet-PG*.

Implementation details. Following [1, 34, 46], decisions are made on short temporal chunks, $\mathcal{C}_t$, where I_t is its central frame. The appearance feature (RGB) of $\mathcal{C}_t$ is extracted from I_t and the motion feature (optical flow) is computed using the whole chunk as input. Following [34, 46], chunk size is fixed to 6 and image frames are obtained at 24 FPS. Two adjacent chunks are not overlapping, thus, there are exactly 4 chunks per second. Following [46], for ClsNet, we set the size of LSTM's hidden state to 4096 and the length of each training sequence to 64. When using CNN, we finetune an FC layer with different CNN features as input (see feature descriptions for each dataset). C3D is pretrained on Sports-1M [105] and finetuned for the per-frame labeling task on each dataset. Hidden state of LocNet is set to

128 and the length of each training sequence, T_{loc}, is fixed to 16. Following [103], γ in Eq. 4.5 is fixed to 0.9. The length of the historical decision vector, n, is set to 8. λ_1 and λ_2 in Eq. 4.7 are fixed to 1. We adopt an alternating strategy for classification and localization training: ClsNet is first trained and fixed afterwards, and then LocNet is trained upon the pre-trained ClsNet. We implement the models in PyTorch [86], and set batch size to 32 for THUMOS'14 and 64 for ActivityNet. For parameter optimization, we used the Adam [87] optimizer with learning rate $5e^{-4}$ and weight decay $5e^{-4}$.

4.3.1 Experiments on THUMOS'14

Dataset. THUMOS'14 [81] is a popular benchmark for temporal action detection. It contains 20 action classes related to sports. There are only trimmed videos in the training set which makes it not appropriate for training ODAS methods. Following [1], we use the validation set (including 200 untrimmed videos, 3K action instances) for training and the test set (including 213 untrimmed videos, 3.3K action instances) for testing.

Feature description. Two types of features are adopted on THUMOS'14 dataset, *RGB* and *Two-Stream (TS)* features. Following [34, 46], we extract appearance (RGB) feature at the *Flatten 673* layer of ResNet-200 [92] and motion feature at the *global pool* layer of BN-Inception [93] with optical flows of 6 consecutive frames as inputs. The TS feature is the concatenation of appearance and motion features,

which are extracted with models[2] pre-trained on ActivityNet.

	Offsets (second)	1	2	3	4	5	6	7	8	9	10
Baselines	SceneDetect [106]	1.0	2.0	2.3	3.1	3.6	4.1	4.7	5.0	5.1	5.2
	ShotDetect [107]	1.1	1.9	2.3	3.0	3.4	3.9	4.3	4.5	4.6	4.9
	Shou et al. [1]	3.1	4.3	4.7	5.4	5.8	6.1	6.5	7.2	7.6	8.2
StartNet-PG	C3D [101] + LocNet	6.8	8.0	9.4	10.1	10.6	10.9	10.9	11.1	11.2	11.2
	CNN [108] + LocNet	17.0	23.6	27.6	29.9	31.3	32.1	33.2	33.5	33.9	34.5
	LSTM [82] + **LocNet**	**19.5**	**27.2**	**30.8**	**33.9**	**36.5**	**37.5**	**38.3**	**38.8**	**39.5**	**39.8**

Table 4.1: Comparisons using p-mAP at depth *Rec*=1.0 on THUMOS'14. Results are under different offset thresholds. ClsNet is implemented with different structures, *i.e.*, C3D, CNN and LSTM. CNN and LSTM are using TS features.

	Depth Rec.	@0.1	@0.2	@0.3	@0.4	@0.5	@0.6	@0.7	@0.8	@0.9	@1.0
Baselines	SceneDetect [106]	30.0	18.3	12.2	9.1	7.2	6.1	5.2	4.6	4.0	3.6
	ShotDetect [107]	26.3	15.9	11.3	8.6	6.8	5.8	4.9	4.3	3.8	3.4
	Shou et al. [1]	42.7	27.3	19.8	14.9	11.8	10.0	8.5	7.4	6.6	5.9
StartNet-PG	C3D [101] + LocNet	34.8	27.7	22.6	19.0	16.3	14.4	12.9	11.8	10.8	10.0
	CNN [108] + LocNet	71.8	64.7	58.0	52.4	47.2	43.3	39.5	35.9	32.5	29.6
	LSTM [82] + **LocNet**	**77.4**	**70.2**	**64.5**	**59.1**	**54.2**	**49.3**	**45.1**	**41.2**	**37.6**	**34.2**

Table 4.2: Comparisons using average p-mAP at different depths on THUMOS'14. Average p-mAP means averaging p-mAP over offsets from 1 to 10 seconds. ClsNet is implemented with different structures, *i.e.*, C3D, CNN and LSTM. CNN and LSTM are using TS features.

4.3.1.1 Evaluation Results

Comparisons with previous methods are shown in Table 4.1 and Table 4.2. Table 4.1 shows comparisons based on p-mAP at depth *Rec*=1.0 under different offset thresholds. All previous methods are under 4% p-mAP at 1 second offset, while StartNet with LSTM achieves 19.5% p-mAP, outperforming the state-of-the-arts largely by over 15%. At 10 seconds offset, previous methods obtain less than 9% p-mAP and StartNet (LSTM) improves over *Shou et al.* [1] by 30% p-mAP. Table 4.2 shows comparisons based on average p-mAP (averaging over offsets from 1 to 10 seconds) at different depths. The results demonstrate that StartNet with

[2]https://github.com/yjxiong/anet2016-cuhk.

LSTM outperforms previous methods significantly (by around 30%-20% average p-mAP) at depth from *Rec*=0.1 to *Rec*=1.0. Obviously, under both metrics, StartNet outperforms previous methods by a very large margin.

To measure the performance gap between online and offline methods. We obtain scores of two recent offline methods [26] and [109] from the authors and evaluate start detection using p-mAP. The p-mAP are 32.7 and 35.7 (*Rec*=1.0, offset is 1 second). As expected, they outperform StartNet, since they observe the entire action before prediction.

4.3.1.2 Ablation Experiments

ClsNet implemented with different structures. Comparisons among StartNet with different ClsNet's backbones are shown in Table 4.1 and Table 4.2. *LSTM+LocNet* achieves the best performance among the three structures and *C3D* performs worse than *CNN* and *LSTM*. *Shou et al.* [1] chose *C3D* as its backbone and proposed sophisticated training strategies for optimization. With *C3D*, StartNet still significantly outperforms *Shou et al.*, which demonstrates the effectiveness of our framework. Since *LSTM+LocNet* achieves the best performance, the following ablation studies are conducted using ClsNet implemented with LSTM.

Effectiveness of LocNet. The results from ClsNet alone can be used to generate action starts by following the *action start generation procedure* in late fusion. To evaluate the contribution of LocNet, we construct *ClsNet-only* by removing LocNet from our framework. Results of *ClsNet-only* can also demonstrate the performance

Features	Offsets (second)	1	2	3	4	5	6	7	8	9	10
RGB	ClsNet-only	11.8	17.2	21.3	24.9	27.9	28.7	29.5	30.0	30.4	30.7
	StartNet-CE	13.7	20.7	23.8	27.2	29.4	30.7	31.9	32.5	33.2	33.6
	StartNet-PG	**15.9**	**21.0**	**24.8**	**28.4**	**30.7**	**31.8**	**33.0**	**33.5**	**34.0**	**34.4**
Two Stream	ClsNet-only	13.9	21.6	25.8	28.9	31.1	32.5	33.5	34.3	34.8	35.2
	StartNet-CE	17.4	25.4	29.8	33.0	34.6	36.3	37.2	37.7	38.6	38.8
	StartNet-PG	**19.5**	**27.2**	**30.8**	**33.9**	**36.5**	**37.5**	**38.3**	**38.8**	**39.5**	**39.8**

Table 4.3: Ablation study of our framework using p-mAP at depth *Rec*=1.0 on THUMOS'14. LSTM is used to implement ClsNet. Different offset thresholds are used to evaluate our framework with different features. Best performance is marked in bold.

Features	Depth Rec.	@0.1	@0.2	@0.3	@0.4	@0.5	@0.6	@0.7	@0.8	@0.9	@1.0
RGB	ClsNet-only	71.2	61.1	52.8	47.0	42.0	37.7	34.0	30.6	27.5	25.3
	StartNet-CE	73.2	64.5	56.8	50.2	45.1	40.5	36.6	33.5	30.5	27.7
	StartNet-PG	**73.6**	**65.0**	**58.0**	**51.2**	**45.9**	**41.5**	**37.8**	**34.3**	**31.5**	**28.8**
Two Stream	ClsNet-only	71.3	63.0	56.9	52.0	46.9	42.3	38.7	35.0	31.8	29.2
	StartNet-CE	72.7	65.6	60.2	55.3	51.0	46.8	43.0	39.2	36.0	32.9
	StartNet-PG	**77.4**	**70.2**	**64.5**	**59.1**	**54.2**	**49.3**	**45.1**	**41.2**	**37.6**	**34.2**

Table 4.4: Ablation study of our framework using average p-mAP at different depths on THUMOS'14. At each depth, we average p-mAP over offset thresholds from 1 to 10 seconds. LSTM is used to implement ClsNet. Best performance is marked in bold.

of OAD methods if applied on the ODAS task directly. As shown in Table 4.3, *ClsNet-only* has already achieved good results, outperforming *C3D* based methods. When adding *LocNet*, *StartNet-PG* improves *ClsNet-only* by 5%-6% p-mAP with TS feature and by 4%-5% p-mAP with RGB features under varying offsets. We can also observe a trend that the gaps between *StartNet-PG* and *ClsNet-only* are larger when the offset is smaller. As shown in Table 4.4, *StartNet-PG* outperforms *ClsNet-only* by 5%-6% p-mAP with TS features and about 3%-5% p-mAP with RGB features at different depths. The qualitative comparison in Fig. 4.3 shows an example that *ClsNet-only* generates a false positive at the last frame. It may be because that the frame contains a classic appearance of the action, *i.e.*, *Basketball Dunk*. With the help of LocNet, the false positive is corrected by *StartNet-PG*.

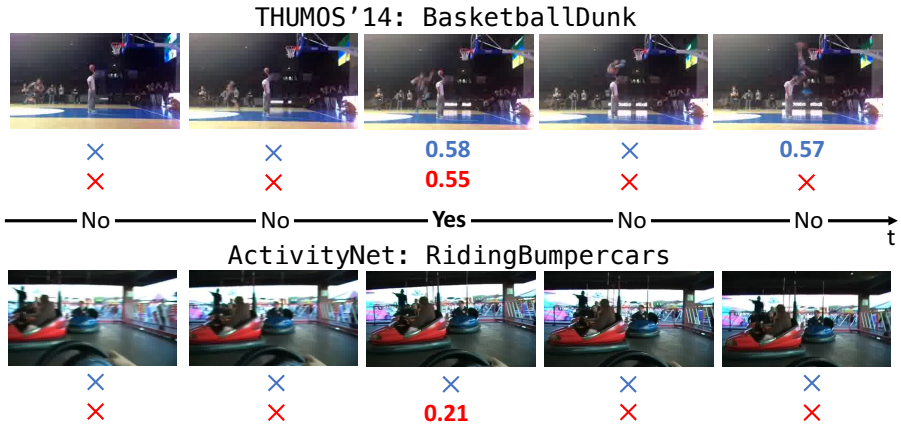


Figure 4.3: Qualitative results on THUMOS'14 and ActivityNet after action start generation in late fusion. × means no starts are detected at those times. Numbers indicate the scores of detected action starts. Results of ClsNet and StartNet are marked in blue and red, respectively. Yes/No (ground-truth) indicates if an action of the associated class starts at the time. Best viewed in color.

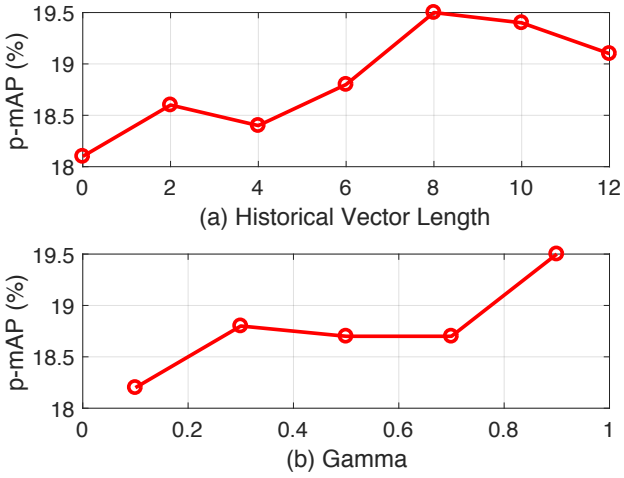


Figure 4.4: Ablation study of LocNet: (a) effect of length of historical decision vector (b) effect of different gamma values in Eq. 4.5. Generally, the model performs better with bigger gamma and longer historical decision vector.

Effectiveness of long-term planning. In order to investigate the effect of long-term planning, we replace the policy gradient training strategy with simple cross-entropy loss – $-\beta g_t log(s_t) - (1-g_t)log(1-s_t)$ – such that every frame is considered independently. This baseline is referred as *StartNet-CE*. Similar to *StartNet-PG*, weight factor, β, is used to handle sample imbalance. Same as α in Eq. 4.4, we set β equal to the ratio between the number of negative samples and positive ones. As shown in Table 4.3 and 4.4, *StartNet-PG* significantly outperforms *StartNet-CE* under each offset threshold and at different depths, which proves the usefulness of the long-term planning.

In order to further investigate effects of parameter settings for LocNet, we conduct an ablation study on different values of the length of historical decision vector, n, and gamma in Eq. 4.5 when offset threshold is set to 1 second and depth *Rec*=1.0. Results are shown in Fig. 4.4. Increasing the length of the historical decision vector means increasing the dependency of later decisions on previous ones. As is shown, the model performs much better when incorporating historical decisions and it reaches its highest performance when 8 historical decisions are considered. Increasing gamma indicates increasing the effect of future rewards to the total long-term reward. It shows that when increasing values of gamma, the model performs better.

Results with different features. To investigated the performance of our framework when using different features, we add experiments with *ClsNet-only*, *StartNet-CE* and *StartNet-PG* using appearance features (RGB) only. Results are displayed

in Table 4.3 and Table 4.4. We see that when using only RGB features, performance of the three models drops. However, even with RGB features, our method still outperforms *Shou et al.* [1] largely.

Effectiveness of two-stage design. We validate our two-stage design by comparing with *one-stage network* which has similar structure as ClsNet (LSTM) except that we modify it to directly predict action starts for all classes and optimize it with cross-entropy loss. We get 6.5% and 10.2% p-mAP at 1 second offset (depth *Rec*=1.0) using RGB and TS features, respectively. The results are much worse than *StartNet-CE* and *StartNet-PG* (drops about 7% and 9%), demonstrating that simply learning classification and localization of action starts jointly is not a good strategy.

Learning from low-level features. Our framework uses action score distributions pretrained on an auxiliary task as inputs of LocNet. We believe that learning from this high-level representation is better than learning from low-level noisy features for our task due to the lack of training data. To prove this point, we construct *StartNet-img* where LocNet learns directly from the low-level image features. The p-mAP using RGB and TS features under offsets of 1 second (depth is 1.0) is 10.2% and 14.0%, respectively, which much under perform our framework (drops about 5%).

Efficiency analysis. We test our method with a single Quadro P6000 GPU. It takes 8ms and 0.3ms on average to forward pass ClsNet(C3D) and LocNet. When using ClsNet (LSTM-TS), LSTM takes 0.3ms. The bottleneck is RGB and motion

feature extraction including flow computation with FlowNet-V2 (97ms). Even so, our method can process each frame within 0.1s in total. One can reduce time largely by using real-time flow extractors, e.g. PWC-Net [110].

4.3.2 Experiments on ActivityNet

Dataset. ActivityNet v1.3 [100] is one of the largest datasets for action recognition. It contains annotations of 200 action classes. There are around 10K untrimmed videos (15K action instances) in the training set and 5K (7.6K action instances) untrimmed videos in the validation set. Averagely, there are around 1.6 action instances in each video. Following [1], we train our models on the train set and test them on the validation set.

Feature description. TS feature is constructed by concatenating appearance and motion features that are extracted from TSN model (with BN-Inception) [108] pretrained on Kinetics [111]. Besides, we validate our method using appearance features extracted from *fc6* layer of VGG-16 [90]. The VGG-16 model is pretrained on ImageNet [89]. VGG-16 features are not as good as ResNet and InceptionNet features for action recognition tasks. We use VGG-16 features to show that our framework can produce reasonable results even when using simple features pretrained only on images.

Training sample strategy of LocNet. Unlike THUMOS'14 which contains around 16 action instances per video in average, ActivityNet has only one action instance in most of the videos. Thus, ActivityNet has much severer imbalance prob-

lem between start and non-start classes. To balance the samples, we randomly select equal numbers of positive and negative sequences for each training batch. Positive sequence is defined as containing at least one action start and negative one contains no action start. Then, α is set to the ratio between the number of negative samples over the number of positive ones after the sample balance.

	Offsets (second)	1	2	3	4	5	6	7	8	9	10
Baselines	SceneDetect [106]	–	–	–	–	–	–	–	–	–	4.7
	ShotDetect [107]	–	–	–	–	–	–	–	–	–	6.1
	Shou et al. [1]	–	–	–	–	–	–	–	–	–	8.3
StartNet	ClsNet-only-VGG	2.7	4.1	5.1	5.9	6.7	7.5	8.1	8.7	9.2	9.8
	StartNet-CE-VGG	4.2	6.1	7.4	8.7	9.7	10.5	11.4	12.0	12.6	13.1
	StartNet-PG-VGG	**6.0**	**7.6**	**8.8**	**9.8**	**10.7**	**11.5**	**12.2**	**12.6**	**13.1**	**13.5**
	ClsNet-only-TS	4.2	6.1	7.7	8.8	9.8	10.7	11.3	12.2	13.0	13.6
	StartNet-CE-TS	6.0	8.3	10.1	11.7	12.9	13.9	15.0	15.8	16.7	17.5
	StartNet-PG-TS	**8.1**	**10.2**	**11.8**	**13.3**	**14.4**	**15.3**	**16.1**	**16.7**	**17.4**	**18.0**

Table 4.5: Comparisons using p-mAP under various offset thresholds at depth *Rec*=1.0 on ActivityNet. ClsNet is implemented with LSTM. Numbers of baseline methods are cited from [1]. – indicates that numbers are not provided in [1].

Evaluation results. Comparisons of StartNet with previous methods on ActivityNet are shown in Table 4.5. StartNet significantly outperforms previous methods. Specifically, StartNet with TS feature achieves similar performance under 1 second offset tolerance compared to *Shou et al.* [1] under 10 seconds offset. At offset of 10 seconds, our method improves *Shou et al.* [1] by around 10%. It also outperforms *SceneDetect* and *ShotDetect* largely by 13.3% and 11.9%, respectively. Even with VGG features pretrained on only images, our method significantly outperforms the state-of-the-arts. Besides, we demonstrate the contribution of each module by comparing with *ClsNet-only* and *StartNet-CE*. Results show that by adding LocNet, *StartNet-PG* improves *ClsNet-only* by over 3% (using VGG) and around 4% (using TS) p-mAP. With long-term planning, *StartNet-PG* significantly

outperforms *StartNet-CE* under both features, especially when the offset tolerance is small. Qualitative results in Fig. 4.3 shows a hard case where *ClsNet-only* misses an action start due to the subtle appearance difference near the start point. With LocNet, *StartNet-PG* successfully captures the start point although the score is low.

4.4 Conclusion

We proposed StartNet to handle Online Detection of Action Starts. StartNet consists of two networks, *i.e.*, ClsNet and LocNet. ClsNet processes the input streaming video and generates action scores for each video frame. LocNet localizes start points by optimizing long-term planning rewards using policy gradient methods. At the end, results from the two sub-networks are fused to produce the final action start predictions. Experimental results on THUMOS'14 and ActivityNet demonstrate that our framework significantly outperforms the state-of-the-arts. Extensive ablation studies were conducted to show the effectiveness of each module.

Chapter 5: C-WSL: Count-guided Weakly Supervised Localization

5.1 Introduction

Convolutional neural networks (CNN) have achieved state-of-the-art performance on the object detection task [2, 11, 12, 13, 45, 69, 112, 113, 114, 115, 116, 117]. However, these detectors are trained in a strongly supervised setting, requiring a large number of bounding box annotations and huge amounts of human labor.

To ease the burden of human annotation, weakly supervised localization (WSL) methods train a detector using weak supervision, *e.g.*, image-level supervision, instead of tight object bounding boxes. The presence of an object category in an image can be obtained on the Internet nearly for free, so most existing WSL architectures require only object categories as supervision.

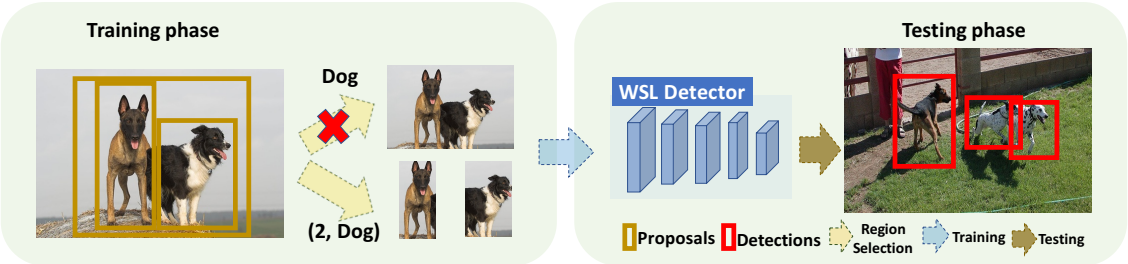


Figure 5.1: Given a set of object proposals and the per-class object count label, we select high-quality positive regions (that tightly cover a single object) to train a WSL detector. Count information significantly reduces detected bounding boxes that are loose and contain two or more object instances, one of the most common errors produced by weakly supervised detectors

Existing methods [5, 6, 17, 18, 19, 20, 21, 118, 119, 120, 121, 122, 123] have proposed different architectures to address the WSL problem. However, there is still a large performance gap between weakly and strongly supervised detectors [2, 11, 13] on standard object detection benchmarks [14, 15, 16]. Often, this is due to the limited information provided by object-category supervision. One major unsolved problem of WSL is that high confidence detections tend to include multiple objects instead of one. As shown in Fig. 5.1 (red cross branch), since training images containing multiple dogs are labeled just as "Dog", detectors tend to learn the composite appearance of multiple dogs as if they were one dog and group multiple dogs as a single instance at test time. To resolve this ambiguity, we use per-class object count information to supervise detector training.

Object count is a type of image-level supervision which is much weaker and cheaper than instance-level supervisions, such as center clicks [24] and bounding boxes. Unlike center click and bounding box annotations, which require several well-trained annotators to specify the center and tight box of each object, object count contains no location information and can be obtained without actually clicking on an object. Moreover, a widely studied phenomenon in psychology, called subitizing [124] suggests that humans are able to determine the number of objects without pointing to or fixating on each object sequentially if the total number of objects in the image is small (typically 1-4) [125]. Thus, people may be able to specify the object count with just a glance. To demonstrate the inexpensiveness of count annotation, we conduct annotation experiments on Pascal VOC2007. Experimental results show that only a small amount of extra time is needed to obtain

per-class object counts compared to labeling just object categories in an image and the response time of the count annotation is much less than that of object center and bounding box.

Our proposed method, Count-guided WSL (C-WSL), is illustrated in Fig. 5.1. During the training process, C-WSL makes use of per-class object count supervision to identify the correct high-scoring object bounding boxes from a set of object proposals. Then, a weakly supervised detector is refined with these high-quality regions as pseudo ground-truth (GT) bounding boxes. This strategy is similar to existing WSL methods that refine detectors using automatically identified bounding boxes [6, 19, 21]. However, since these methods do not make use of object count supervision, they treat only the top-scoring region as the pseudo GT box, regardless of the number of object instances present in the image. This sometimes leads to multiple object instances being grouped into a single pseudo GT box, which hurts the detector's ability to localize individual objects. With the guidance of the object count label, C-WSL selects tight box regions that cover individual objects as shown in Fig. 5.1 (the "(2, Dog)" branch).

The main contribution of C-WSL is that it uses per-class object count, a cheap and effective form of image-level supervision, to address a common failure case in WSL where one detected bounding box contains multiple object instances. To implement C-WSL, we develop a simple Count-based Region Selection (CRS) algorithm and integrate it into two existing architectures—alternating detector refinement (ADR) and online detector refinement (ODR)—to significantly improve WSL. Experimental results on Pascal VOC2007 [14] and VOC2012 [15] show that C-WSL

significantly improves WSL detection and outperforms state-of-the-art methods.

5.2 Proposed Approach

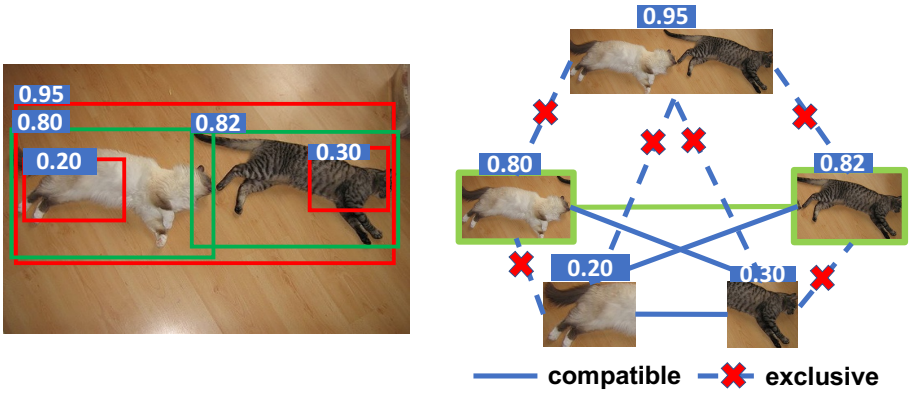


Figure 5.2: A common failure case of WSL methods (left) and graph representation of our region selection formulation (right). Our goal is to select the two green boxes, each of which tightly covers one object, as the positive training samples for WSL detectors. We achieve this by analyzing the confidence scores and spatial constraints among regions

C-WSL selects regions covering a single object with the help of per-class object count supervision and then refines the WSL detector using these regions as the pesudo GT bounding boxes. We first introduce a simple Count-based Region Selection (CRS) algorithm that C-WSL relies on to select high-quality regions from object proposals on training images. Then, we integrate CRS into two detector refinement structures to improve weakly supervised detectors.

5.2.1 Count-based Region Selection (CRS)

As shown in Fig. 5.2 (left), without object count information, previous methods often select the top-scoring box in training images as the positive training sample

to refine the WSL detector [6, 19, 21]. Their detection performance is degraded because in many cases the top-scoring box contains multiple objects from the same category, *e.g.*, two cats. Our goal is to select distinct regions, each covering a single object as positive training samples with the help of object count constraints so that the detector will learn the appearance of a single cat.

We formulate the problem as a region selection problem. Given a set of boxes $\mathbf{B} = \{b_1, ..., b_N\}$ and the corresponding confidence scores $\mathbf{P} = \{p_1, ..., p_N\}$ (*e.g.*, the detection score of a region in each detector refinement iteration), a subset $\mathbf{G}$ is selected as the set of positive training regions where $|\mathbf{G}| = C$ and C indicates the per-class object count. We identify a good subset $\mathbf{G}$ using a greedy algorithm applied to a graphical representation of the set of boxes. Each box is represented as a node in the graph, and two nodes are connected if the spatial overlap of their corresponding boxes is below a threshold (See solid line in Fig. 5.2). The greedy algorithm provides an approximation to the following optimization problem:

$$\mathbf{G}^* = \arg\max_{\mathbf{G}} \sum_{b_k \in \mathbf{G}} p_k, s.t.\ |\mathbf{G}| = C,\ a_o(b_i, b_j) < T\ \forall b_i, b_j \in \mathbf{G}, i \neq j. \tag{5.1}$$

To encourage selecting regions containing just one object, we use the asymmetric area of overlap, i.e, $a_o(b_i, b_j) = \frac{area(b_i \cap b_j)}{area(b_j)}$, which has been proposed in [59, 126] to model spatial overlap between two boxes, where b_i is a box previously selected by the greedy algorithm and b_j indicates a box considered for selection. T is the overlap threshold. If the algorithm has previously added a large box to the solution, thresholding on a_o will discourage the selection of its subregions, regardless of their

sizes.[1] So, to deliver a high total score, the algorithm prefers C small high-scoring boxes to one large box, even though the large box may have the highest score.

We conduct region selection after applying non-maximum suppression on a complete set of the detection boxes, so the number of nodes is limited to a reasonable number, and the computation cost is low in practice. The algorithm is summarized in Alg. 2.

Algorithm 2: Count-based Region Selection (CRS)

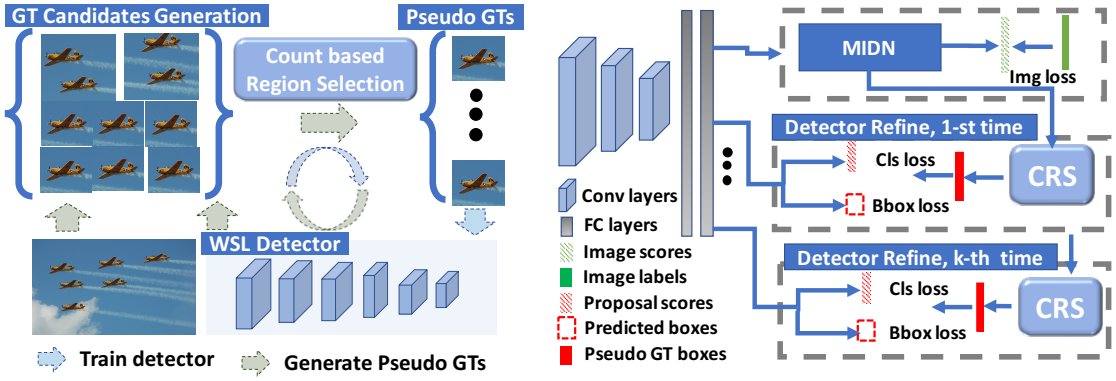
Input: $\mathbf{B} = \{b_1, ..., b_N\}$, $\mathbf{P} = \{p_1, ..., p_N\}$, T, C;
$\mathbf{B}$ is a list of candidate boxes;
$\mathbf{P}$ is the corresponding scores;
T is the overlap threshold;
C indicates the object count;
Initialization: Sort (descend) $\mathbf{B}$ based on $\mathbf{P}$;
$\mathbf{G}^* \leftarrow \emptyset$; $s_{max} \leftarrow 0$;
Output: $\mathbf{G}^*$
for $i \in \{1, ..., N\}$ **do**
  $\mathbf{G} \leftarrow b_i$; $s \leftarrow p_i$;
  for $j \in \{i+1, ..., N\}$ **do**
    if $a_o(b_k, b_j) < T(\forall b_k \in \mathbf{G})$ **then**
      $\mathbf{G} \leftarrow \mathbf{G} \cup \{b_j\}$; $s \leftarrow s + p_j$
      if $|\mathbf{G}| == C$ *or* $j == N$ **then**
        if $s > s_{max}$ **then**
          $s_{max} \leftarrow s$; $\mathbf{G}^* \leftarrow \mathbf{G}$
          break;

5.2.2 Detector Refinement Structures with CRS

5.2.2.1 Alternating Detector Refinement (ADR).

We first integrate CRS into an alternating WSL refinement architecture, where a poor weakly supervised detector can be refined iteratively. The architecture is

[1] The commonly used symmetric intersection-over-union measure would select sufficiently small regions even if they were fully overlapped by an existing large box.



(a) Alternating detector refinement (b) Online detector refinement w/ CRS

Figure 5.3: (a): Count-based Region Selection (*CRS*) is applied to select high-quality positive training regions from the ground-truth (GT) candidate boxes generated by a WSL detector. The WSL detector is then refined using these regions. (b): The Multiple Instance Detection Network(*MIDN*) [5, 6] and multiple detector networks share the same feature representation to refine the detector at all stages together. *Cls loss* indicates the classification loss and *Bbox loss* indicates bounding box regression loss

shown in Fig. 5.3, where a WSL detector alternates between generating high-quality regions as pseudo ground-truth (GT) boxes and refining itself using these GT boxes. Some WSL methods are based on a strategy like this [21, 118]. The major difference is that we use CRS to select multiple high-quality regions as the GT boxes.

Initialization phase. We first generate a set of box candidates from the training data using a pre-trained WSL detector. This set of box candidates is treated as the initialized pseudo GTs and will be refined iteratively afterwards.

Alternating training phase. We use Fast R-CNN [8] as our WSL network. Starting from the initialized pseudo GT boxes, Fast R-CNN alternates between improving itself via retraining with the pseudo GT boxes generated by CRS and generating a refined set of GT candidate boxes on the training images.

5.2.2.2 Online Detector Refinement (ODR).

As argued in [6], the alternating strategy has two potential limitations: 1) it is time consuming to alternate between training on the fixed labels and generating labels by the trained model; 2) separating refinements into different iterations might harm performance since it hinders the procedure from sharing image representations across iterations.

Based on [6], we propose an online detector refinement framework integrated with CRS. An illustration of the proposed method is shown in Fig. 5.3. A Multiple Instance Detection Network (MIDN) and several detector refinement stages share the same feature representation extracted from a backbone structure. The MIDN utilizes an object-category label to supervise its training as in [5, 6]. Each detector refinement network outputs the classification score and predicted bounding box for each region proposal. The predicted boxes with scores at each stage will be used to select pseudo GTs for the next stage refinement. Compared to [6], we have two major differences: 1) we use CRS to generate high-quality regions as pseudo GTs rather than just choosing the top-scoring region; 2) we use both classification loss and bounding box regression loss for detector refinement, just as RCNNs do. Note that the inputs to CRS produced by MIDN are the proposals with scores before the summation over proposals.

5.3 Experiments

We compare with the existing WSL methods which are trained by object class labels to show the advantage of per-class count supervision. It may seem an 'unfair' comparison, since the per-class count provides more information compared to object class. However, we demonstrate via our annotation experiment that the cost of the additional information is very low, which makes it reasonable to determine how much improvement can be gained by adding this information.

5.3.1 Experimental Setup

Datasets and Evaluate Metrics. Comparisons with state-of-the-art methods are conducted on VOC2007 [14] and VOC2012 [15] which contain 20 object categories. For VOC2007, all the models are trained on the *trainval* set which contains 5,011 images and evaluated on *test* set which includes 4,952 images. For VOC2012, models are trained on 5,717 images of the *train* set and evaluated on 5,823 images in the *val* set. We use two widely used metrics for localization evaluation: Correct localization (CorLoc) [119] and Average Precision (AP) [127]. CorLoc evaluates localization accuracy by measuring if the maximum response point of a detection is inside the ground truth bounding box. AP evaluates models by comparing IoU between output and ground truth bounding boxes.

Implementation Details. We fix $T = 0.1$ for all models at all the iterations on both datasets. Note that our experiments show that the method is robust to T, *e.g.,* varying T from 0.1 to 1 with step 0.1, we achieved (Mean, Std) = (47.2%, 0.42%)

mAP. Following [6, 21], we set the total iteration number to 3 and use *VGG16* [90] as the backbone structure for both ADR and ODR. For fair comparison, the existing works also use *VGG16* except for [118] which utilizes *AlexNet*. In ADR, we strictly follow the steps of training Fast-RCNN at each iteration and use all the released default training parameters except that we use the generated pseudo GT boxes instead of the bounding box labels. In ODR, we follow the basic MIDN structure and training process from [6], and use the parameters released by the author. Note that we use the same classification and bounding box regression loss in ODR as in [8].

Variants of Our Approach. *C-WSL:WSLPDA/OICR+ADR* indicates ADR initialized with a pre-trained WSLPDA [19] (or OICR [6]) model where CRS is used to select confident GT boxes in each iteration. Then, a Fast-RCNN is alternatively refined as we mentioned in Sec. 5.2.2.1. *C-WSL:ODR* indicates the structure shown in Fig. 5.3(b). *C-WSL:ODR+FRCNN* denotes a Fast RCNN trained with the top-scoring region generated by *C-WSL:ODR* to improve results (inspired by [6, 19]). *C-WSL** indicates models trained by our annotated counts.

5.3.2 Annotation Time vs. Detection Accuracy

Object counting is very straightforward. The user interface includes an image and 15 buttons indicating the count numbers. We cap object count with 15 since it is very rare to have a count of the same class bigger than 15. Similar to the click experiments [21], an annotator was given a category and was asked to click the

count corresponding to that category. Following [24], given an object category, we measure the response time of counting the object instances from the moment the image appears until the count is determined.

Annotation evaluations are conducted on the full *trainval* set with 20 categories of VOC2007 [14]. The average response time of counting a single object per class per image is $0.90s$. Average response time per image of annotating a single image class is from $1.5s$ to $1.9s$ [128] and that of annotating count given object class is $1.48s$, so obtaining per-class object count from an image only needs $1.48/1.9 = 78\%$ to $1.48/1.5 = 99\%$ more time compared to annotating just the object class.

Annotation time of object counts per image increases as the number of objects increases. However, it might not always be helpful to count all the objects, especially for images with many objects, since these images are more likely to depict complex scenes, *e.g.*, significant occlusions and small object instances, and for such images the generated GT candidates might not include all the objects in the first place. Thus, we evaluate the detection accuracy of our model using at most K per-class objects annotation, where K is the upper bound of per-class object instances that are counted for each image. Obviously, K has positive correlation with annotation time, since annotators may not be able to subitize for high values of K and will need to spend an amount of time proportional to K in order to produce an accurate count. Analysis of mAP and average $CorLoc$ vs. K is shown in Fig. 5.4. The results suggest that the detection accuracy reaches the highest point when at most 3 per-class objects are counted per image. Average annotation time per image for images with at most 3 per-class objects is $1.20s$ which is $63\% \sim 80\%$ overhead

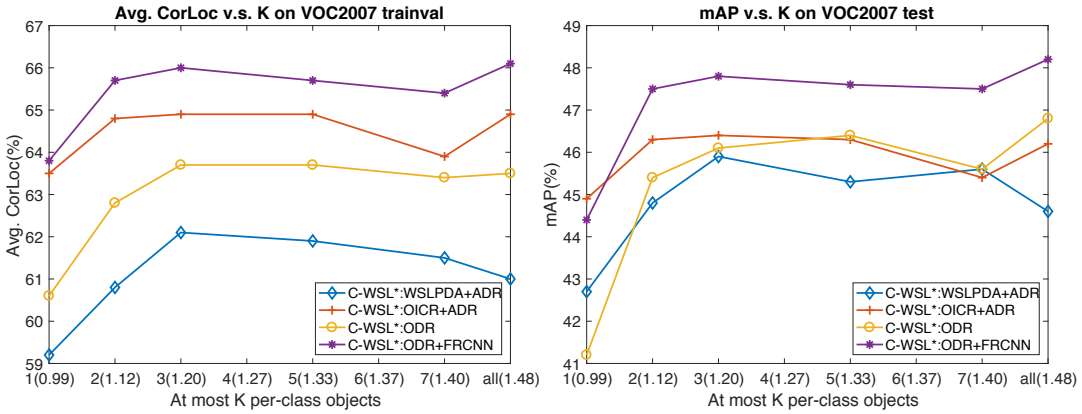


Figure 5.4: Detection accuracy analysis when at most K per-class objects are counted in an image. Average annotation time (in seconds) per image under each K is shown in the parentheses. Detection accuracy becomes stable when K=3

Method	Faster-RCNN [2]	Two-clicks [24]	One-click [24]	C-WSL*:ODR+FRCNN
mAP(%)	69.9	49.1(AlexNet)/57.5(VGG16)	45.9(AlexNet)	48.2(VGG16)
Annotation cost	34.5s/img+anno. train +re-draw rejected boxes	3.74s/img+anno. train +re-click rejected clicks	1.87s/img+anno. train +re-click rejected clicks	1.48s/img

Table 5.1: Accuracy vs. cost among bounding box, clicks and count supervisions on VOC2007. We use [2] as a reference of fully supervised detector

compared to object category annotations. We compare our models trained by our annotated counts and those obtained from the VOC2007 annotations in Table 5.2 and 5.3. The results demonstrate that models trained by the two sets of annotations have comparable performance, which suggests that our annotation is as useful as the VOC2007 annotations. Thus, in the following analysis, we just use (*C-WSL*) VOC2007 annotations.

Accuracy and cost comparisons among box, clicks and count supervisions are shown in Table 5.1. Although the accuracy of our approach does not outperform supervised and two-click methods, we have achieved a significant reduction in annotation cost. We are 38× and 4× faster regarding to response time for labeling a single image. In addition, box and clicks annotations require additional repeated annotator training to accurately locate objects and lengthy quality control processes.

Methods	are	bik	brd	boa	btl	bus	car	cat	cha	cow	tbl	dog	hrs	mbk	prs	plt	shp	sfa	trn	tv	mAP
Cinbis *et al.* [118]	39.3	43.0	28.8	20.4	8.0	45.5	47.9	22.1	8.4	33.5	23.6	29.2	38.5	47.9	20.3	20.0	35.8	30.8	41.0	20.1	30.2
Wang *et al.* [120]	48.8	41.0	23.6	12.1	11.1	42.7	40.9	35.5	11.1	36.6	18.4	35.3	34.8	51.3	17.2	17.4	26.8	32.8	35.1	45.6	30.9
Jie *et al.* [21]	52.2	47.1	35.0	26.7	15.4	61.3	66.0	54.3	3.0	53.6	24.7	43.6	48.4	65.8	6.6	18.8	51.9	43.6	53.6	62.4	41.7
WSDDN [5]	39.4	50.1	31.5	16.3	12.6	64.5	42.8	42.6	10.1	35.7	24.9	38.2	34.4	55.6	9.4	14.7	30.2	40.7	54.7	46.9	34.8
WSDDN+Context [17]	57.1	52.0	31.5	7.6	11.5	55.0	53.1	34.1	1.7	33.1	49.2	42.0	47.3	56.6	15.3	12.8	24.8	48.9	44.4	47.8	36.3
WSDDN-Ens. [5]	46.4	58.3	35.5	25.9	14.0	66.7	53.0	39.2	8.9	41.8	26.6	38.6	44.7	59.0	10.8	17.3	40.7	49.6	56.9	50.8	39.3
WCCN-3stage [20]	49.5	60.6	38.6	29.2	16.2	70.8	56.9	42.5	10.9	44.1	29.9	42.2	47.9	64.1	13.8	23.5	45.9	54.1	60.8	54.5	42.8
WSLPDA [19]	54.5	47.4	41.3	20.8	17.7	51.9	63.5	46.1	21.8	57.1	22.1	34.4	50.5	61.8	16.2	29.9	40.7	15.9	55.3	40.2	39.5
OICR [6]	58.0	62.4	31.1	19.4	13.0	65.1	62.2	28.4	24.8	44.7	30.6	25.3	37.8	65.5	15.7	24.1	41.7	46.9	64.3	62.6	41.2
OICR-Ens.+FRCNN[a] [6]	64.5	64.4	44.1	25.9	16.9	67.8	68.4	33.2	9.0	57.5	46.4	21.7	57.8	64.3	10.0	23.7	50.6	60.9	64.7	58.0	45.5
C-WSL:ODR	**62.7**	**63.7**	**40.0**	**25.5**	**17.7**	**70.1**	**68.3**	**38.9**	**25.4**	**54.5**	**41.6**	**29.9**	**37.9**	**64.2**	**11.3**	**27.4**	**49.3**	**54.7**	**61.4**	**67.4**	**45.6**
C-WSL*:ODR	**62.9**	**64.8**	**39.8**	**28.1**	**16.4**	**69.5**	**68.2**	**47.0**	**27.9**	**55.8**	**43.7**	**31.2**	**43.8**	**65.0**	**10.9**	**26.1**	**52.7**	**55.3**	**60.2**	**66.6**	**46.8**
C-WSL:ODR+FRCNN	**61.9**	**61.9**	**48.6**	**28.7**	**23.3**	**71.1**	**71.3**	**38.7**	**28.5**	**60.6**	**45.4**	**26.3**	**49.7**	**65.5**	**7.2**	**27.3**	**54.7**	**61.6**	**63.2**	**59.5**	**47.8**
C-WSL*:ODR+FRCNN	**62.9**	**68.3**	**52.9**	**25.8**	**16.5**	**71.1**	**69.5**	**48.2**	**26.0**	**58.6**	**44.5**	**28.2**	**49.6**	**66.4**	**10.2**	**26.4**	**55.3**	**59.9**	**61.6**	**62.2**	**48.2**

Table 5.2: Comparison with the state-of-the-art in terms of mAP on the VOC2007 *test* set. Our number is marked in red if it is the best in the column

[a]The numbers are reproduced by using the code released by the author.

Methods	are	bik	brd	boa	btl	bus	car	cat	cha	cow	tbl	dog	hrs	mbk	prs	plt	shp	sfa	trn	tv	Avg.
Cinbis *et al.* [118]	65.3	55.0	52.4	48.3	18.2	66.4	77.8	35.6	26.5	67.0	46.9	48.4	70.5	69.1	35.2	35.2	69.6	43.4	64.6	43.7	52.0
Wang *et al.* [120]	80.1	63.9	51.5	14.9	21.0	55.7	74.2	43.5	26.2	53.4	16.3	56.7	58.3	69.5	14.1	38.3	58.8	47.2	49.1	60.9	48.5
Jie *et al.* [21]	72.7	55.3	53.0	27.8	35.2	68.6	81.9	60.7	11.6	71.6	29.7	54.3	64.3	88.2	22.2	53.7	72.2	52.6	68.9	75.5	56.1
WSDDN [5]	65.1	58.8	58.5	33.1	39.8	68.3	60.2	59.6	34.8	64.5	30.5	43.0	56.8	82.4	25.5	41.6	61.5	55.9	65.9	63.7	53.5
WSDDN+Context [17]	83.3	68.6	54.7	23.4	18.3	73.6	74.1	54.1	8.6	65.1	47.1	59.5	67.0	83.5	35.3	39.9	67.0	49.7	63.5	65.2	55.1
WSDDN-Ens. [5]	68.9	68.7	65.2	42.5	40.6	72.6	75.2	53.7	29.7	68.1	33.5	45.6	65.9	86.1	27.5	44.9	76.0	62.4	66.3	66.8	58.0
WCCN-3stage [20]	83.9	72.8	64.5	44.1	40.1	65.7	82.5	58.9	33.7	72.5	25.6	53.7	67.4	77.4	26.8	49.1	68.1	27.9	64.5	55.7	56.7
SP-VGGNet [121]	85.3	64.2	67.0	42.0	16.4	71.0	64.7	88.7	20.7	63.8	58.0	84.1	84.7	80.0	60.0	29.4	56.3	68.1	77.4	30.5	60.6
WSLPDA [19]	78.2	67.1	61.8	38.1	36.1	61.8	78.8	55.2	28.5	68.8	18.5	49.2	64.1	73.5	21.4	47.4	64.6	22.3	60.9	52.3	52.4
OICR [6]	81.7	80.4	48.7	49.5	32.8	81.7	85.4	40.1	40.6	79.5	35.7	33.7	60.5	88.8	21.8	57.9	76.3	59.9	75.3	81.4	60.6
OICR-Ens.+FRCNN[2] [6]	88.3	78.8	62.8	48.9	38.9	83.2	85.4	50.0	21.9	77.4	45.6	41.9	79.3	91.6	12.6	60.8	86.6	70.2	80.2	79.9	64.2
C-WSL:ODR	**86.3**	**80.4**	**58.3**	**50.0**	**36.6**	**85.8**	**86.2**	**47.1**	**42.7**	**81.5**	**42.2**	**42.6**	**50.7**	**90.0**	**14.3**	**61.9**	**85.6**	**64.2**	**77.2**	**82.4**	**63.3**
C-WSL*:ODR	**85.8**	**81.2**	**64.9**	**50.5**	**32.1**	**84.3**	**85.9**	**54.7**	**43.4**	**80.1**	**42.2**	**42.6**	**60.5**	**90.4**	**13.7**	**57.5**	**82.5**	**61.8**	**74.1**	**82.4**	**63.5**
C-WSL:ODR+FRCNN	**85.8**	**78.0**	**61.6**	**52.1**	**44.7**	**81.7**	**88.4**	**49.1**	**50.0**	**82.9**	**44.1**	**44.4**	**63.9**	**92.4**	**14.3**	**60.4**	**86.6**	**68.3**	**80.6**	**82.8**	**65.6**
C-WSL*:ODR+FRCNN	**87.5**	**81.6**	**65.5**	**52.1**	**37.4**	**83.8**	**87.9**	**57.6**	**50.3**	**80.8**	**44.9**	**44.4**	**65.6**	**92.8**	**14.9**	**61.2**	**83.5**	**68.5**	**77.6**	**83.5**	**66.1**

Table 5.3: Comparison with the state-of-the-art in terms of CorLoc (%) on the VOC2007 *trainval* set. Our number is marked in red if it is the best in the column

Our annotation does not require knowing the location of an object so it avoids the sensitivity to location noise. Consequently, we do not need annotator training and quality control in our experiments.

5.3.3 Comparison with State-of-the-art (SOTA) Approaches

Comparison in terms of *mAP* on the VOC2007 *test* set and *CorLoc* on the VOC2007 *trainval* set are shown in Table 5.2 and 5.3, respectively. Overall, the proposed *C-WSL:ODR+FRCNN* outperforms all the existing SOTA methods using both *CorLoc* and *mAP* measurements.

Methods	are	bik	brd	boa	btl	bus	car	cat	cha	cow	tbl	dog	hrs	mbk	prs	plt	shp	sfa	trn	tv	mAP
WSLPDA [19]	54.5	47.4	41.3	20.8	17.7	51.9	63.5	46.1	21.8	57.1	22.1	34.4	50.5	61.8	16.2	29.9	40.7	15.9	55.3	40.2	39.5
WSLPDA+ADR	57.9	68.3	47.8	20.3	12.2	52.9	67.6	68.8	24.6	50.0	24.9	49.8	54.8	63.5	14.1	27.4	41.2	19.5	57.1	30.7	42.7
C-WSL:WSLPDA+ADR	**60.5**	**70.1**	**52.5**	**24.7**	**24.4**	**63.6**	**71.8**	**58.1**	**26.0**	**66.4**	**26.5**	**34.7**	**55.0**	**65.8**	**8.8**	**31.9**	**51.6**	**20.4**	**60.0**	**41.8**	**45.7**
OICR [6]	58.0	62.4	31.1	19.4	13.0	65.1	62.2	28.4	24.8	44.7	30.6	25.3	37.8	65.5	15.7	24.1	41.7	46.9	64.3	62.6	41.2
OICR+ADR	58.1	61.2	43.3	24.4	19.4	65.5	67.1	34.3	3.6	56.5	45.5	26.4	61.9	60.7	10.4	23.6	49.2	62.1	61.4	64.2	44.9
C-WSL:OICR+ADR	**61.7**	**66.8**	**45.6**	**21.1**	**23.5**	**67.2**	**73.8**	**32.5**	**10.6**	**54.6**	**42.9**	**16.6**	**59.2**	**63.3**	**11.0**	**25.4**	**55.3**	**61.3**	**67.4**	**67.8**	**46.4**

Table 5.4: Comparison with baselines in terms of mAP on the VOC2007 *test* set. The table contains two comparison groups separated by double solid lines. Each group shows how much ADR and C-WSL improve each baseline. Underline is used if the C-WSL variant outperforms its baselines

Methods	are	bik	brd	boa	btl	bus	car	cat	cha	cow	tbl	dog	hrs	mbk	prs	plt	shp	sfa	trn	tv	Avg.
WSLPDA [19]	78.2	67.1	61.8	38.1	36.1	61.8	78.8	55.2	28.5	68.8	18.5	49.2	64.1	73.5	21.4	47.4	64.6	22.3	60.9	52.3	52.4
WSLPDA+ADR	84.6	76.9	69.7	41.0	21.8	68.5	83.2	77.6	34.4	76.7	19.8	73.7	75.2	84.7	26.3	53.8	70.1	22.3	73.8	50.9	59.2
C-WSL:WSLPDA+ADR	**83.3**	**80.0**	**70.9**	**51.6**	**41.2**	**73.6**	**85.3**	**67.7**	**40.7**	**79.5**	**20.9**	**54.7**	**79.6**	**87.1**	**24.5**	**56.8**	**83.5**	**20.7**	**76.0**	**60.2**	**61.9**
OICR [6]	81.7	80.4	48.7	49.5	32.8	81.7	85.4	40.1	40.6	79.5	35.7	33.7	60.5	88.8	21.8	57.9	76.3	59.9	75.3	81.4	60.6
OICR+ADR	85.8	76.9	65.8	49.5	38.5	83.2	84.8	49.7	14.0	79.5	46.8	41.2	80.3	89.2	15.0	60.1	84.5	66.4	78.3	80.6	63.5
C-WSL:OICR+ADR	**85.4**	**78.0**	**65.5**	**49.5**	**43.5**	**84.3**	**87.5**	**48.0**	**23.6**	**80.8**	**43.3**	**38.8**	**79.9**	**92.8**	**15.8**	**60.1**	**87.6**	**66.4**	**81.0**	**80.3**	**64.6**

Table 5.5: Comparison with the baseline detectors in terms of CorLoc (%) on the VOC2007 *trainval* set. The table contains two comparison groups separated by double solid lines. Each group shows how much ADR and C-WSL improve each baseline. Underline is used if the C-WSL variant outperforms its baselines

Table 5.4 and 5.5 compare our variants with the two baseline detectors, *i.e.*, WSLPDA [19] and OICR [6]. The results suggest that even the simple ADR strategy can significantly improve the results. Moreover, if we use object count information, we can largely improve WSLPDA by 6.2% *mAP* (9.5% average *CorLoc*) and OICR by 5.2% *mAP* (4.0% average *CorLoc*). C-WSL improves the results of *WSLPDA+ADR* on 17 (15) out of 20 categories and the results of *OICR+ADR* on 10 (10) out of 20 categories in terms of *mAP* on the VOC2007 *test* set (in terms of *CorLoc* on the VOC2007 *trainval* set).

As stated in Sec. 5.1, the object count information is helpful to avoid a detector localizing on multiple objects. To demonstrate this point, we first calculate the percentage of images that have more than one per-class object (multi-objects percentage) in VOC2007. As shown in Fig. 5.5, "bottle", "car", "chair", "cow", "person", "plant" and "sheep" have a high percentage of images which in-

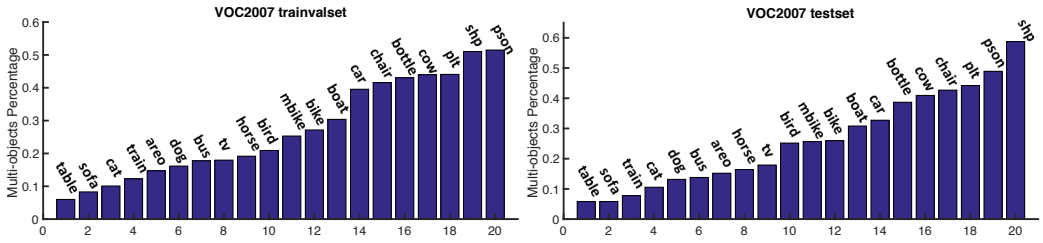


Figure 5.5: Image number of multiple-objects over image number of non-zero objects. Note that "pson" means "person", "plt" means "plant" and "shp" denotes "sheep". C-WSL works better on most classes with high multiple-objects percentage. See Sec. 5.3.3

clude more than one object in the corresponding category. As shown in Table 5.2 and 5.3, *C-WSL:ODR+FRCNN* outperforms SOTA methods for 5 out of these 7 categories. When looking into the effect of object count supervision on WSLPDA and OICR, we see significant improvement on these categories as shown in Table 5.4 and 5.5. Consider the "sheep" category for example. *C-WSL:WSLPDA+ADR* improves *WSLPDA+ADR* by 13.4% *CorLoc* and 10.4% *AP*. *C-WSL:OICR+ADR* improves *OICR+ADR* by 3.1% *CorLoc* and 6.1% *AP*. Fig. 5.6 shows some examples of training regions selected by *OICR+CRS* and *OICR*. *OICR* tends to select regions containing multiple instances, while object count helps to obtain regions including a single instance. Qualitative comparison between our *C-WSL:ODR+FRCNN* and *OICR-Ens.+FRCNN* on the VOC2007 *test* set is shown in Fig. 5.8, demonstrating that our approach achieves more precise localization when multiple per-class objects appear in an image. We will further analyze our approach on images with different numbers of objects in Sec. 5.3.4.

Table 5.6 and 5.7 show the comparison of C-WSL with the SOTA on VOC2012. Note that results of WSLPDA and OICR models are reproduced by running the

Methods	are	bik	brd	boa	btl	bus	car	cat	cha	cow	tbl	dog	hrs	mbk	prs	plt	shp	sfa	trn	tv	mAP
Jie *et al.* [21]	60.9	53.3	31.0	16.4	18.2	58.2	50.5	55.6	9.1	42.1	12.1	43.4	45.3	64.6	7.4	19.3	44.8	39.3	51.4	57.2	39.0
OICR-Ens.+FRCNN [6]	71.0	68.2	52.7	20.1	27.2	57.3	57.1	19.0	8.0	50.6	30.2	34.5	63.3	69.5	1.2	20.5	48.5	55.2	41.1	60.4	42.8
WSLPDA [19]	42.2	27.8	32.7	4.2	13.7	52.1	35.8	48.3	11.8	31.7	4.9	30.4	45.3	51.8	11.5	13.4	33.5	7.2	45.6	38.4	29.1
WSLPDA+ADR	70.0	65.6	46.3	14.4	22.8	57.5	54.2	67.5	16.1	45.0	4.4	40.0	51.7	71.8	5.8	27.7	38.3	11.7	55.2	34.1	40.0
C-WSL:WSLPDA+ADR	**69.8**	**62.8**	**52.7**	**16.7**	**28.3**	**61.1**	**56.6**	**58.0**	**18.5**	**47.8**	**5.1**	**36.3**	**53.3**	**66.8**	**6.8**	**24.2**	**47.1**	**11.0**	**60.1**	**43.4**	**41.3**
OICR [6]	71.0	59.1	42.3	27.4	20.2	58.7	46.4	18.6	18.1	45.7	21.7	20.5	53.1	68.5	1.8	15.7	42.7	40.0	41.0	61.5	38.7
OICR+ADR	67.0	63.1	50.8	12.8	23.8	55.3	55.1	16.1	5.2	47.2	23.4	28.2	55.9	69.2	1.9	21.5	46.5	49.9	35.9	63.8	39.6
C-WSL:OICR+ADR	**71.3**	**68.3**	**50.9**	**17.1**	**24.8**	**60.9**	**56.4**	**13.9**	**14.5**	**54.6**	**22.2**	**25.7**	**57.7**	**70.4**	**1.6**	**20.0**	**55.8**	**46.0**	**35.7**	**62.9**	**41.5**
C-WSL:ODR	**74.0**	**67.3**	**45.6**	**29.2**	**26.8**	**62.5**	**54.8**	**21.5**	**22.6**	**50.6**	**24.7**	**25.6**	**57.4**	**71.0**	**2.4**	**22.8**	**44.5**	**44.2**	**45.2**	**66.9**	**43.0**
C-WSL:ODR+FRCNN	**75.3**	**71.6**	**52.6**	**32.5**	**29.9**	**62.9**	**56.9**	**16.9**	**24.5**	**59.0**	**28.9**	**27.6**	**65.4**	**72.6**	**1.4**	**23.0**	**49.4**	**52.3**	**42.4**	**62.2**	**45.4**

Table 5.6: Comparison with the state-of-the-art in terms of mAP on the VOC2012 *val* set. Our number is marked in red if it is the best in the column. Underline is used if the C-WSL variant outperforms its baselines

Methods	are	bik	brd	boa	btl	bus	car	cat	cha	cow	tbl	dog	hrs	mbk	prs	plt	shp	sfa	trn	tv	Avg.
OICR-Ens.+FRCNN [6]	85.4	81.5	70.4	44.7	46.6	83.6	78.4	33.9	29.3	83.2	51.6	50.5	86.1	88.0	11.0	56.7	82.5	69.1	65.1	83.6	64.1
WSLPDA [19]	80.5	63.7	64.4	34.1	29.3	76.7	71.5	62.8	30.3	76.1	23.0	55.3	75.2	77.7	18.7	56.4	66.7	25.1	66.5	54.8	55.4
WSLPDA+ADR	87.2	79.7	72.4	38.6	40.9	82.6	75.2	79.8	35.1	81.3	18.9	62.1	82.4	83.9	21.6	60.9	75.4	29.5	74.5	55.5	61.9
C-WSL:WSLPDA+ADR	**85.7**	**77.2**	**73.4**	**38.6**	**46.4**	**84.9**	**75.8**	**69.1**	**43.0**	**76.8**	**20.1**	**58.6**	**79.8**	**79.6**	**20.3**	**57.8**	**79.5**	**35.4**	**76.4**	**61.9**	**62.0**
OICR [6]	86.6	80.4	65.2	57.6	42.1	85.4	72.5	28.0	45.7	79.4	46.2	34.0	78.2	87.2	7.5	55.0	83.6	58.5	62.2	84.3	62.0
OICR+ADR	84.5	79.0	72.4	39.0	47.1	83.6	79.9	31.9	25.0	84.5	48.7	48.3	87.8	88.7	13.3	55.0	82.5	67.4	65.1	83.9	63.4
C-WSL:OICR+ADR	**86.6**	**80.8**	**73.9**	**43.2**	**44.4**	**87.7**	**76.2**	**32.2**	**34.0**	**87.1**	**49.1**	**46.2**	**88.2**	**91.2**	**12.1**	**57.1**	**78.4**	**65.5**	**65.1**	**85.3**	**64.2**
C-WSL:ODR	**90.9**	**81.1**	**64.9**	**57.6**	**50.6**	**84.9**	**78.1**	**29.8**	**49.7**	**83.9**	**50.9**	**42.6**	**78.6**	**87.6**	**10.4**	**58.1**	**85.4**	**61.0**	**64.7**	**86.6**	**64.9**
C-WSL:ODR+FRCNN	**92.1**	**84.3**	**69.9**	**58.3**	**53.9**	**86.8**	**80.4**	**30.6**	**52.6**	**83.9**	**54.7**	**45.8**	**83.2**	**90.1**	**12.7**	**56.4**	**86.0**	**64.9**	**66.5**	**84.3**	**66.9**

Table 5.7: Comparison with the state-of-the-art in terms of *CorLoc* on the VOC2012 *train* set. Our number is marked in red if it is the best in the column. Underline is used if the C-WSL variant outperforms its baselines

pretrained model and the code released by the authors. The results suggest that our method outperforms the SOTA method (*OICR-Ens.+FRCNN*) by 2.6% in *mAP* on the VOC2012 *val* set and by 2.8% in *CorLoc* on the VOC2012 *train* set. C-WSL improves the results of *WSLPDA+ADR* on 12 (10) out of 20 categories and the results of *OICR+ADR* on 10 (12) out of 20 categories in terms of *mAP* on the VOC2012 *val* set (in terms of *CorLoc* on the VOC2012 *train* set).

We also evaluated our methods and baselines (pre-trained on the VOC2007

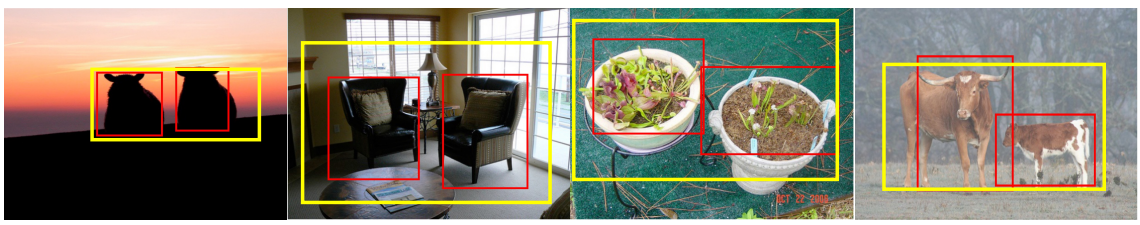


Figure 5.6: Examples of the training regions selected by *OICR+CRS* (red) and *OICR* (yellow). The regions selected by *OICR* contain multiple object instances. Object count information helps to select regions, each covering a single instance

trainval set) on the common 20 classes in MS COCO [16] 35k-val2014 set using COCO mAP@0.5 metric. Although not fine-tuned on COCO, our approaches still outperform the baseline methods. The results are that C-WSL:WSLPDA improves WSLPDA [19] from 17.9% to 19.6%. C-WSL:OICR+ADR improves OICR [6] from 18.7% to 20.1% and C-WSL:ODR+FRCNN improves OICR-Ens.+FRCNN [6] from 19.0% to 20.0%.

5.3.4 Ablation Analysis

Two major components contribute to the success of our approach. One is the iterative training process (alternating/online) and the other one is the per-class object count supervision. In Table 5.4 and 5.5, we can see the improvement by adding ADR and object count into the system. For WSLPDA [19], iterative training (ADR) improves *mAP* by 3.2% and the count information (CRS) increases it by 3%. For OICR [6], ADR helps by increasing 3.7% *mAP* and CRS contributes 1.5%. In the following, we analyze each component in detail.

Number of iterations. ADR performances as a function of the number of iterations using the WSLDPA and OICR models is shown in Fig. 5.7(a). Generally, models improve as the number of iterations increases. When adding object count supervision into the framework, the results of both WSLDPA and OICR models improve faster, which demonstrates the advantage of count information in WSL.

Number of object instances per image. Adding the object count constraint helps a detector focus on a single object rather than multiple objects. To demon-

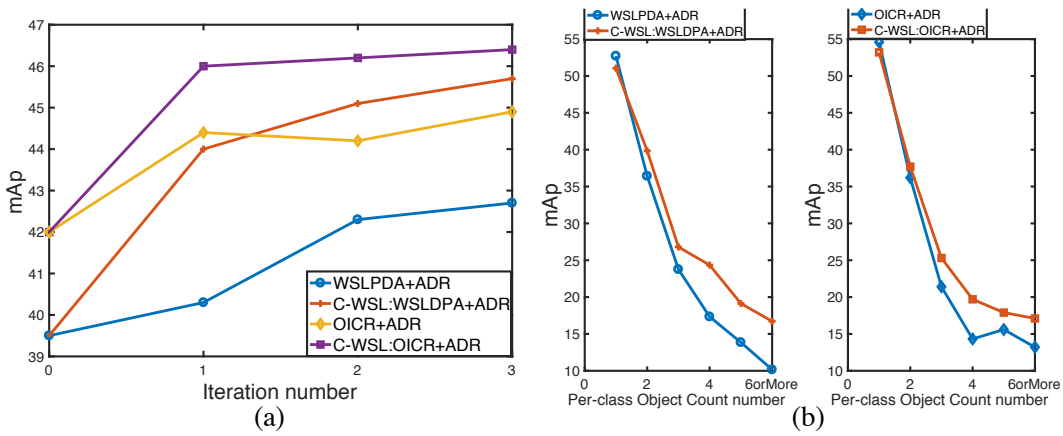


Figure 5.7: (a): model improvement as the number of *ADR* iterations increases on the VOC2007 *test* set. *C-WSL* approaches improve faster than others. (b): Evaluation on images with different per-class object counts on VOC2007. Our approach outperforms the WSL detectors in the presence of multiple instances in a test image

strate this, we partition images in the VOC2007 *test* set based on their per-class object count and re-evaluate our approaches on each subset.

The results are shown in Fig. 5.7(b). For both WSLPDA and OICR, the performance is much better under C-WSL. Generally, the gaps between curves of with and without C-WSL are bigger as the object count number increases.

5.3.5 Error Analysis

The results shown in Table 5.2, 5.3, 5.6 and 5.7 suggest that most existing WSL detectors perform poorly on the "person" category: strongly supervised detectors achieve more than 76% AP on the VOC2007 *test* set (*e.g.*, 76.6% [11] and 76.3% [2]), while the best WSL detection result on "person" is 20.3% (see Table 5.2). This result is likely due to the large appearance variations of persons in the dataset. Without constraints provided by tight bounding boxes, rigid parts are easier to learn and mostly sufficient to differentiate the object from others. So, WSL detectors focus

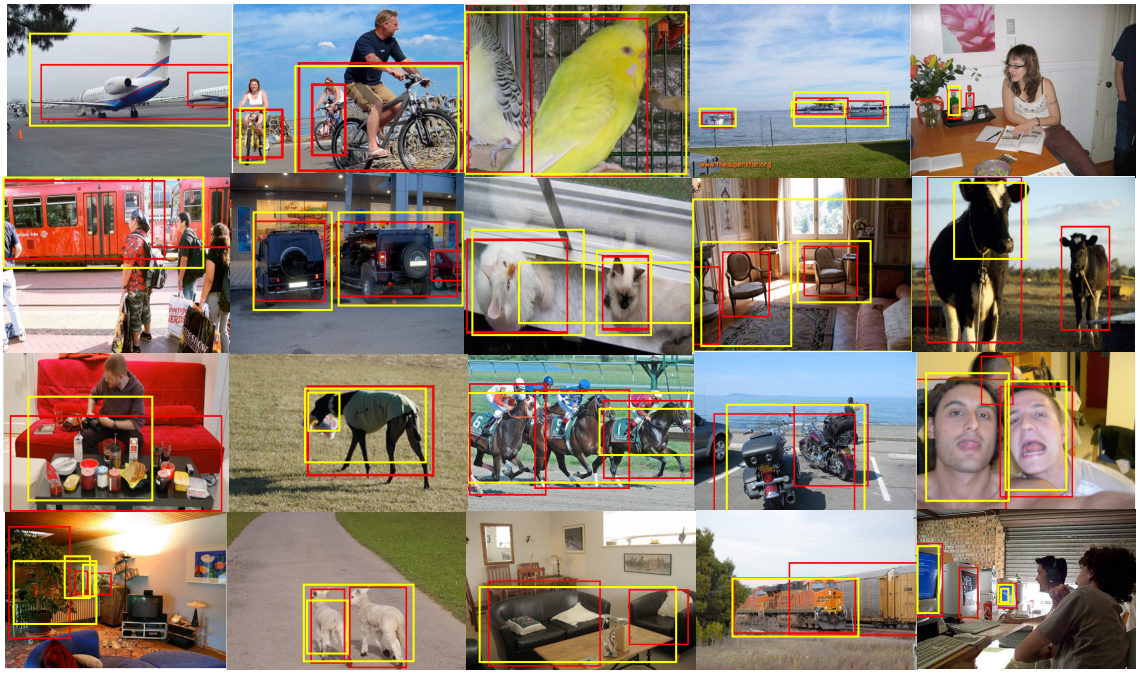


Figure 5.8: Qualitative comparison between our *CWSL:ODR+FRCNN* (red boxes) and *OICR+FRCNN* (yellow boxes) on the VOC2007 *test* set over the 20 classes. Our detector detects much tighter bounding boxes, yields much fewer boxes with multiple objects in them, and finds instances more accurately

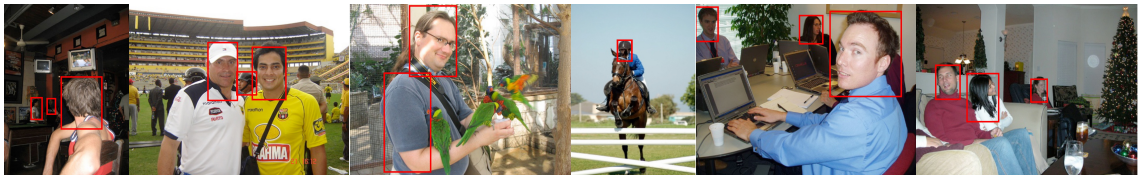


Figure 5.9: Some examples of the common failure cases of our approach (*CWSL:ODR+FRCNN*) on the "person" category of the VOC2007 *test* set

on local parts instead of the whole object as shown in Fig. 5.9.

Intuitively, this can be overcome if we can roughly estimate the size of object instances. We conducted a preliminary experiment as follows. Suppose that we know the size of the smallest instance of an object category in an image and assume all the object parts are smaller than the smallest object. This assumption is not generally true and we use it just as a proof-of-concept. We preprocess the region candidates by removing all boxes whose size is smaller than the smallest object and

then conduct *C-WSL:WSLPDA+ADR* on VOC2007. The *AP* on "person" improves to 40.0% and the *mAP* over all the classes improves to 52.7%.

5.4 Conclusion

We proposed a Count-guided Weakly Supervised Localization (C-WSL) framework where a cheap and effective form of image-level supervision, *i.e.*, per-class object count, is used to select training regions each of which tightly covers a single object instance for detector refinement. As a part of C-WSL, we proposed a Count-based Region Selection (CRS) algorithm to perform high-quality region selection. We integrated CRS into two detector refinement architectures to improve WSL detectors. Experimental results demonstrate the effectiveness of C-WSL. To prove the inexpensiveness of the per-class object count annotation, we conduct annotation experiments on VOC2007. The results show that only a small amount of time is needed to obtain the count information in an image and that we reduce the annotation time of center click and bounding box by more than 2× and 38× respectively.

Chapter 6: WSLLN: Weakly Supervised Natural Language Localization Networks

6.1 Introduction

Extensive work has been done on temporal action/activity localization [25, 26, 27, 28, 29, 30], where an action of interest is segmented from long, untrimmed videos. These methods only identify actions from a pre-defined set of categories, which limits their application to situations where only unconstrained language descriptions are available. This more general problem is referred to as natural language localization (NLL) [4, 7]. The goal is to retrieve a temporal segment from an untrimmed video based on an arbitrary text query. Recent work focuses on learning the mapping from visual segments to the input text [4, 7, 35, 36, 37] and retrieving segments based on the alignment scores. However, in order to successfully train a NLL model, a large number of diverse language descriptions are needed to describe different temporal segments of videos which incurs high human labeling cost.

We propose Weakly Supervised Language Localization Networks (WSLLN) which requires only video-sentence pairs during training with no information of where the activities temporally occur. Intuitively, it is much easier to annotate

video-level descriptions than segment-level descriptions. Moreover, when combined with text-based video retrieval techniques, video-sentence pairs may be obtained with minimum human intervention. The proposed model is simple and clean, and can be trained end-to-end in a single stage. We validate our model on *ActivityNet Captions* and *DiDeMo*. The results show that our model achieves the state-of-the-art of the weakly supervised approach and has comparable performance as some supervised approaches.

6.2 Weakly Supervised Language Localization Networks (WSLLN)

6.2.1 Problem Statement

Following the setting of its strongly supervised counterpart [4, 7], the goal of a weakly supervised language localization (WSLL) method is to localize the event that is described by a sentence query in a long, untrimmed video. Formally, given a video consisting of a sequence of image frames, $\mathbf{V}_i = [I_i^1, I_i^2, ..., I_i^T]$, and a text query Q_i, the model aims to localize a temporal segment, $[I_i^{st}, ..., I_i^{ed}]$, which semantically aligns best with the query. *st* and *ed* indicate the start and end times, respectively. The difference is that WSLL methods only utilize video-sentence pairs, $\{\mathbf{V}_i, Q_i\}_{i=1}^N$, for training, while supervised approaches have access to the start and end times of the queries.

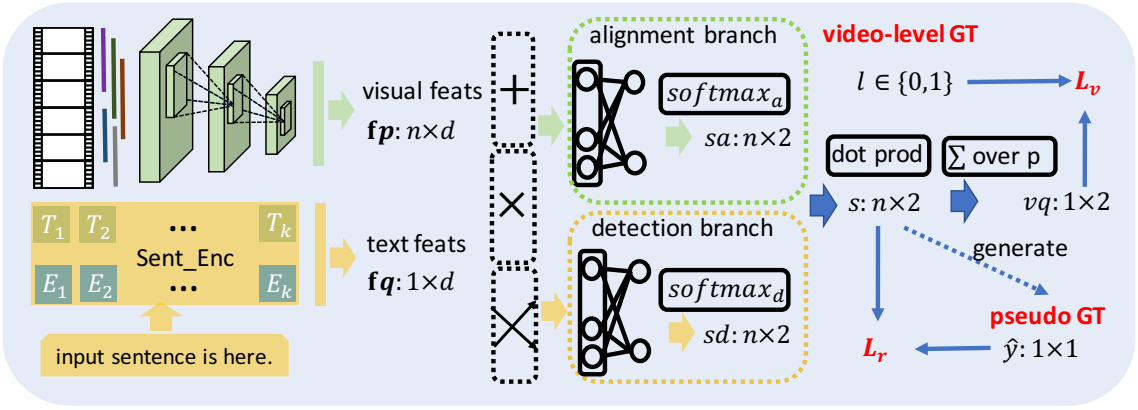


Figure 6.1: The workflow of our method. Visual and text features are extracted from n video proposals and the input sentence. Fully-connected (FC) layers are used to transform the features to the same length, d. The two features are combined by multi-modal processing [7] and input to the two-branch structure. Scores from both parts are merged. Video-level scores, vq, are obtained by summing s over proposals. The whole pipeline is trained end-to-end using video-level and pseudo segment-level labels. $x \times z$ indicates dimensions.

6.2.2 The Proposed Approach

Taking frame sequences, $[I_i^1, I_i^2, ..., I_i^T]$, as inputs, the model first generates a set of temporal proposals, $\{p_i^1, p_i^2, ..., p_i^n\}$, where p_i^j consists of temporally-continuous image frames. Then, the method aligns the proposals with the input query and outputs scores for proposals, $\{s_i^1, s_i^2, ..., s_i^n\}$, indicating their likelihood of containing the event.

Feature Description. Given a sentence query Q_i of arbitrary length, sentence encoders can be used to extract text feature, fq_i, from the query. For a video, $\mathbf{V}_i = [I_i^1, I_i^2, ..., I_i^T]$, features, $\mathbf{fv}_i = [fv_i^1, fv_i^2, ..., fv_i^T]$, are extracted from each frame. Following [4], the visual feature, fp_i^j, of a proposal p_i^j is obtained using Eq. 6.1, where $pool(x, t_1, t_2)$ means average pooling features x from time t_1 to t_2, $||$ indicates concatenation, j_{st}/j_{ed} indicates start/end times of the proposal and $\bar{j}$ means time is

normalized to $[0, 1]$.

$$pool(\mathbf{fv}_i, j_{st}, j_{ed})||pool(\mathbf{fv}_i, 0, T)||[\bar{j}_{st}, \bar{j}_{ed}] \tag{6.1}$$

We see that the feature of each proposal contains the information of its visual pattern, the overall context and its relative position in the video.

Following [7], features of the sentence and a visual proposal are combined as in Eq. 6.2. The feature, fm, will be used to measure the matching between a candidate proposal and the input query.

$$fm = (fp + fq)||(fp \cdot fq)||FC(fp||fq) \tag{6.2}$$

The workflow of WSLLN is illustrated in Fig. 6.1. Inspired by the success of the two-stream structure in the weakly supervised object and action detection tasks [5, 43], WSLLN consists of two branches, *i.e.*, alignment branch and selection branch. The semantic consistency between the input text and each visual proposal is measured in the alignment branch. The proposals are compared and selected in the detection branch. Scores from both branches are merged to produce the final results.

Alignment Branch produces the consistency scores, $sa_i \in \mathrm{R}^{n\times 2} = [sa_i^1, sa_i^2, ..., sa_i^n]$, for proposals of the video-sentence pair. sa_i in Eq. 6.3, measures how well each proposal matches the text. Different proposal scores are calculated independently where $softmax_a$ indicates applying the softmax function over the last dimension.

$$sa_i = softmax_a(\mathbf{W}_a fm_i) \tag{6.3}$$

Detection Branch performs proposal selection. The selection score, $sd_i \in \mathrm{R}^{n\times 2} = [sd_i^1, sd_i^2, ..., sd_i^n]$ in Eq. 6.4, is obtained by applying softmax function over proposals. Through softmax, the score of a proposal will be affected by those of other proposals, so this operation encourages competition among segments.

$$sd_i = softmax_d(\mathbf{W}_d fm_i) \tag{6.4}$$

Score Merging is applied to both parts to obtain the results by dot production, *i.e.*, $s_i = sa_i \cdot sd_i$, for proposals. s_i is used as the final segment-sentence matching scores during inference.

Training Phase. To utilize video-sentence pairs as supervision, our model is optimized as a video-sentence matching classifier. We compute the matching score of a given video-sentence pair by summing s_i^j over proposals, $vq_i = \sum_{j=1}^{n} s_i^j$. Then, L_v is obtained in Eq. 6.5 by measuring the score with the video-sentence match label $l_i \in \{0, 1\}$. Positive video-sentence pairs can be obtained directly. We generate negative ones by pairing each video with a randomly selected sentence in the training set. We ensure that the positive pairs are not included in the negative set.

$$L_v = loss(vq_i, l_i) \tag{6.5}$$

Results can be further refined by adding an auxiliary task L_r in Eq. 6.6 where

$\hat{y}_i = \{0, 1, ..., n-1\}$ indicates the index of the segment that best matches the sentence during training. The real segment-level labels are not available, thus we generate pseudo labels by setting $\hat{y}_i = argmax_j s_i^j[:, 1]$. This loss further encourages competition among proposals.

$$L_r = loss(s_i^j, \hat{y}_i) \tag{6.6}$$

The overall objective is minimizing L in Eq. 6.7, where λ is a balancing scalar. $loss$ is cross-entropy loss.

$$L = loss(vq_i, l_i) + \lambda loss(s_i^j, \hat{y}_i). \tag{6.7}$$

6.3 Experiments

6.3.1 Experimental Settings

Implementation Details. BERT [129] is used as the sentence encoder, where the feature of '[CLS]' at the last layer is extracted as the sentence representation. Visual and sentence features are linearly transformed to have the same dimension, $d = 1000$. The hidden layers for both branches have 256 units. For *ActivityNet Captions*, we take the $n = 15$ proposals over multiple scales of each video provided by [3] and use the C3D [101] features provided by [130]. For *DiDeMo*, we use the $n = 21$ proposals and VGG [90] features (RGB and Flow) provided in [4].

Evaluation Metrics. Following [4, 7], *R@k,IoU=th* and *mIoU* are used for eval-

uation. Proposals are ranked according to their matching scores with the input sentence. If the temporal IoU between at least one of the top-k proposals and the groundtruth is bigger or equal to th, the sentence is counted as matched. $R@k,IoU{=}th$ means the percentage of matched sentences over the total sentences given k and th. $mIoU$ is the mean IoU between the top-1 proposal and the groundtruth.

6.3.2 Experiments on ActivityNet Captions

Dataset Description. *ActivityNet Captions* [130] is a large-scale dataset of human activities. It contains 20k videos including 100k video-sentences in total. We train our models on the training set and test them on the validation set. Although the dataset provides segment-level annotation, we only use video-sentence pairs during training.

Baselines. We compare with strongly supervised approaches, *i.e.*, CTRL [7], ABLR [131] and WSDEC-S [3] to see how much accuracy it sacrifices when using only weak labels. Originally proposed for dense-captioning, WSDEC-W [3] achieves state-of-the-art performance for weakly supervised language localization. Although showing good performance, WSDEC-W involves complicated training stages, and alternates between sentence localization and caption generation for iterations.

Model	WS	IoU=0.1	IoU=0.3	IoU=0.5	mIoU
CTRL	F	49.1	28.7	14.0	20.5
ABLR	F	73.3	55.7	36.8	37.0
WSDEC-S	F	70.0	52.9	37.6	40.4
WSDEC-W	T	62.7	42.0	23.3	28.2
WSLLN	T	75.4	42.8	22.7	32.2

Table 6.1: Comparison results based on $R@1$ on *ActivityNet Captions*. All baseline numbers are reprinted from [3]. WS: weakly supervised.

6.3.2.1 Comparison Results

Comparison results are displayed in Table 6.1. It shows that WSLLN largely outperforms WSDEC-W by $\sim$4% $mIoU$. When comparing with strongly supervised methods, WSLLN outperforms CTRL by over 11% $mIoU$. Using the $R@1, IoU = 0.1$ metric, our model largely outperforms all the baselines including strongly and weakly supervised methods which means that when a scenario is flexible with the IoU coverage, our method has great advantage over others. When th =0.3/0.5, our model has comparable results as WSDEC-W and largely outperforms CTRL. The overall results demonstrate good performance of WSLLN, even though there is still a big gap between weakly supervised methods and some supervised ones, *i.e.*, ABLR and WSDEC-S. $mIoU$ (mean±std) of WSLLN across 3 runs is 32.2 ± 0.05 which demonstrates the robustness of our method.

6.3.2.2 Ablation Study

Effect of λ. We evaluate the effect of λ (see Eq. 6.7) in Table 6.2. As it shows, our model performs stable when λ is set from 0.1 to 0.4. When $\lambda = 0$, the refining module is disabled and the performance drops. When λ is set to a big number, *e.g.*,

$\lambda \rightarrow$	0.0	0.1	0.2	0.3	0.4	0.5
IoU=0.1	64.9	75.4	75.5	75.5	75.5	66.6
IoU=0.3	36.2	42.8	42.9	42.9	42.9	38.3
IoU=0.5	19.4	22.7	22.7	22.8	22.7	20.7
mIoU	27.4	32.2	32.3	32.3	32.3	28.8

Table 6.2: R@1 results of our method on *ActivityNet Captions* when λ in Eq. 6.7 is set to be different values.

0.5, the contribution of L_v is reduced and the model performance also drops.

Effect of Sentence Encoder. WSDEC-W uses GRU [96] as its sentence encoder, while our method uses BERT. It seems an unfair comparison, since BERT is powerful than GRU in general. However, we uses pretrained BERT model without fine tuning on our dataset, while WSDEC-W uses GRU but performed an end-to-end training. So, it is unclear which setting is better. To resolve this concern, we replace our BERT with GRU following WSDEC-W. The R@1 results when IoU is set to be 0.1, 0.3 and 0.5 are 74.0, 42.3 and 22.5, respectively. The mIoU is 31.8. It shows that our model with GRU has comparable results as that with BERT.

Effect of Two-branch Design. We create two baselines, *ie*, *Align-only* and *Detect-only*, to demonstrate the effectiveness of our design. To perform fair comparison, both of them are trained using only video-sentence pairs.

Align-only contains only the alignment branch. For positive video sentence pair, we give positive labels to all proposals. Negative pairs have negative labels for all the proposals. Loss is calculated between proposal scores and the generated segment-level labels.

Detect-only contains only the detection branch. Loss is calculated using the highest detection score over proposals and the video-level label at each training

iteration.

Comparison results are displayed in Table 6.3. It shows that the two baselines underperform WSLLN by a large margin, which demonstrates the effectiveness of our design.

Model	IoU=0.1	IoU=0.3	IoU=0.5	mIoU
Align-only	40.0	18.9	7.5	13.4
Detect-only	33.7	18.3	10.4	13.6

Table 6.3: Ablation study based on R@1 on *ActivityNet Captions*. Both methods are trained using weak supervisions.

6.3.3 Experiments on DiDeMo

Dataset Description. *DiDeMo* was proposed in [4] for the language localization task. It contains 10k, 30-second videos including 40k annotated segment-sentence pairs. Our models are trained using video-sentence pairs in the train set and tested on the test set.

Baselines. To the best of our knowledge, no weakly supervised method has been evaluated on *DiDeMo*. So, we compare with some supervised methods, *i.e.*, MCN [4] and LOR [132]. MCN is a supervised NLL model. LOR is a supervised language-object retrieval model. It utilizes much more expensive (object-level) annotations for training. We follow the same setup of LOR as in [4] to evaluate LOR for our task.

Comparison Results are shown in Table 6.4. WSLLN performs better than LOR in terms of R@1/5. We also observe that the gap between our method and the supervised NLL model is much larger on *DiDeMo* than on *ActivityNet Captions*.

This may be due to the fact that *DiDeMo* is a much smaller dataset which is a disadvantage for weakly supervised learning.

Model	WS	Input	R@1	R@5	mIoU
Chance	–	–	3.75	22.50	22.64
LOR	F	RGB	16.2	43.9	27.2
MCN	F	RGB	23.1	73.4	35.5
MCN	F	Flow	25.8	75.4	38.9
WSLLN	T	RGB	19.4	53.1	25.4
WSLLN	T	Flow	18.4	54.4	27.4

Table 6.4: Comparison results on *DiDeMo*. Following MCN, we set $th = 1.0$ for the IoU threshold. All baseline numbers are reprinted from [4]. WS: weakly supervised.

6.4 Conclusion

We propose WSLLN– a simple language localization network. Unlike most existing methods which require segment-level supervision, our method is optimized using video-sentence pairs. WSLLN is based on a two-branch architecture where one branch performs segment-sentence alignment and the other one conducts segment selection. Experiments show that WSLLN achieves promising results on *ActivityNet Captions* and *DiDeMo*.

Chapter 7: Conclusion

We proposed methods to improve efficiency of object detectors in terms of both deployment at testing time and supervision at training time, and introduced approaches to improve temporal modeling for online action detectors in long, untrimmed videos. These approaches are motivated by the following insights: (1) CNNs are computationally expensive, so methods that improve their deployment efficiency is essential; (2) CNNs are data hungry. Tremendous labeled data is required to successfully train CNN models, so investigating methods to train models using weak supervisions can largely reduce human labor for annotation and (3) Complex temporal information in videos makes action recognition challenging. Thus, improving modeling temporal dependencies are necessary for action detectors.

We introduced (1) a coarse-to-fine strategy to speed up CNN object detectors on large image; (2) a sophiscated structure to improve temporal modeling; (3) an effeceive temporal modeling framework to improve online detection of action start; (4) an efficient approach to refine object detectors using only object counts as weak supervisions and (5) a useful framework to model langugage localization in videos using only video-sentence pairs for training.

There are also other directions can be explored for efficient model training and

effective temporal modeling. Promising future work may include training models in semi-supervised manners; integrating model training and data labeling using active learning techniques and improving temporl modeling using self-supervised signals in videos.

Bibliography

[1] Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. Online action detection in untrimmed, streaming videos-modeling and evaluation. In *European Conference on Computer Vision*, 2018.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[3] Xuguang Duan, Wenbing Huang, Chuang Gan, Jingdong Wang, Wenwu Zhu, and Junzhou Huang. Weakly supervised dense event captioning in videos. In *Advances in Neural Information Processing Systems*, pages 3059–3069, 2018.

[4] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *IEEE International Conference on Computer Vision*, pages 5803–5812, 2017.

[5] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[6] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. *arXiv preprint arXiv:1704.00138*, 2017.

[7] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *IEEE International Conference on Computer Vision*, pages 5267–5275, 2017.

[8] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, 2017.

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[13] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[14] 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.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[17] Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *European Conference on Computer Vision*, pages 350–365. Springer, 2016.

[18] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. *IEEE International Conference on Computer Vision*, 2017.

[19] Dong Li, Jia-Bin Huang, Yali Li, Shengjin Wang, and Ming-Hsuan Yang. Weakly supervised object localization with progressive domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[20] Ali Diba, Vivek Sharma, Ali Pazandeh, Hamed Pirsiavash, and Luc Van Gool. Weakly supervised cascaded convolutional networks. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5131–5139, 2017.

[21] Zequn Jie, Yunchao Wei, Xiaojie Jin, Jiashi Feng, and Wei Liu. Deep self-taught learning for weakly supervised object localization. *arXiv preprint arXiv:1704.05188*, 2017.

[22] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 854–863, 2016.

[23] Alexander Kolesnikov and Christoph H Lampert. Improving weakly-supervised object localization by micro-annotation. *British Machine Vision Conference*, 2016.

[24] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[25] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[26] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *IEEE International Conference on Computer Vision*, 2017.

[27] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *IEEE International Conference on Computer Vision*, 2017.

[28] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. SST: Single-stream temporal action proposals. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[29] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. TURN TAP: Temporal unit regression network for temporal action proposals. *IEEE International Conference on Computer Vision*, 2017.

[30] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[31] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3d network for temporal activity detection. In *IEEE International Conference on Computer Vision*, 2017.

[32] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *European Conference on Computer Vision*, 2016.

[34] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. In *British Machine Vision Conference*, 2017.

[35] Bingbin Liu, Serena Yeung, Edward Chou, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Temporal modular networks for retrieving complex compositional activities in videos. In *European Conference on Computer Vision*, pages 552–568, 2018.

[36] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with temporal language. In *Empirical Methods in Natural Language Processing*, 2018.

[37] Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. *arXiv preprint arXiv:1812.00087*, 2018.

[38] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[39] Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis R Bach, and Jean Ponce. Automatic annotation of human actions in video. In *IEEE International Conference on Computer Vision*, 2009.

[40] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[41] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.

[42] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016.

[43] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4325–4334, 2017.

[44] Zheng Shou, Hang Gao, Lei Zhang, Kazuyuki Miyazawa, and Shih-Fu Chang. Autoloc: Weakly-supervised temporal action localization in untrimmed videos. In *European Conference on Computer Vision*, pages 154–171, 2018.

[45] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[46] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S Davis, and David J Crandall. Temporal recurrent networks for online action detection. *IEEE International Conference on Computer Vision*, 2019.

[47] Mingfei Gao, Mingze Xu, Larry S Davis, Richard Socher, and Caiming Xiong. Startnet: Online detection of action start in untrimmed videos. *IEEE International Conference on Computer Vision*, 2019.

[48] Mingfei Gao, Ang Li, Ruichi Yu, Vlad I Morariu, and Larry S Davis. C-wsl: Count-guided weakly supervised localization. In *European Conference on Computer Vision*, 2018.

[49] Mingfei Gao, Larry S Davis, Richard Socher, and Caiming Xiong. Wslln: Weakly supervised natural language localization networks. *Empirical Methods in Natural Language Processing*, 2019.

[50] Michael Figurnov, Dmitry P. Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. *CoRR*, abs/1504.08362, 2015.

[51] Xiangyu Zhang, Jianhua Zou, Xiang Ming, Kaiming He, and Jian Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[52] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann Lecun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems 27*, pages 1269–1277. Curran Associates, Inc., 2014.

[53] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR*, abs/1511.06530, 2015.

[54] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[55] Ian Endres and Derek Hoiem. Category independent object proposals. In *European Conference on Computer Vision*, pages 575–588. Springer, 2010.

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

[57] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.

[58] Ziming Zhang, Yun Liu, Tolga Bolukbasi, Ming-Ming Cheng, and Venkatesh Saligrama. Bing++: A fast high quality object proposal generator at 100fps. *arXiv preprint arXiv:1511.04511*, 2015.

[59] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2012.

[60] Sebastian Kalkowski, Christian Schulze, Andreas Dengel, and Damian Borth. Real-time analysis and visualization of the yfcc100m dataset. In *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*, pages 25–30. ACM, 2015.

[61] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[62] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[63] Richard Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769, 1956.

[64] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[65] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[66] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

[67] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville. Dynamic capacity networks. In *International Conference on Machine Learning*, pages 2549–2558, 2016.

[68] Mingfei Gao, Ang Li, Ruichi Yu, Vlad I. Morariu, and Larry S. Davis. C-wsl: Count-guided weakly supervised localization. *arXiv preprint arXiv:1711.05282*, 2017.

[69] Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. *IEEE International Conference on Computer Vision*, 2017.

[70] Ang Li, Jin Sun, Joe Yue-Hei Ng, Ruichi Yu, Vlad I. Morariu, and Larry S. Davis. Generating holistic 3d scene abstractions for text-based image retrieval. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[71] Ruichi Yu, Hongcheng Wang, and Larry S. Davis. Remotenet: Efficient relevant motion event detection for large-scale home surveillance videos. *IEEE Winter Conference on Applications of Computer Vision*, 2018.

[72] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[73] Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *CoRR*, abs/1512.04143, 2015.

[74] Roeland De Geest and Tinne Tuytelaars. Modeling temporal structure with lstm for online action detection. In *IEEE Winter Conference on Applications of Computer Vision*, 2018.

[75] Yu Yao, Mingze Xu, Yuchen Wang, David J Crandall, and Ella M Atkins. Unsupervised traffic accident detection in first-person videos. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[76] Grace Edwards, Petra Vetter, Fiona McGruer, Lucy S. Petro, and Lars Muckli. Predictive feedback to V1 dynamically updates with sensory input. *Scientific Reports*, 2017.

[77] Andreja Bubic, D Yves Von Cramon, and Ricarda I Schubotz. Prediction, cognition and the brain. *Frontiers in Human Neuroscience*, 2010.

[78] Andy Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 2013.

[79] Joseph Fruchter, Tal Linzen, Masha Westerlund, and Alec Marantz. Lexical preactivation in basic linguistic phrases. *Journal of Cognitive Neuroscience*, 2015.

[80] Vasili Ramanishka, Yi-Ting Chen, Teruhisa Misu, and Kate Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[81] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014.

[82] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[83] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

[84] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[85] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[86] `http://pytorch.org/`.

[87] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[88] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261*, 2016.

[89] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[90] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[91] Yuanjun Xiong, Limin Wang, Zhe Wang, Bowen Zhang, Hang Song, Wei Li, Dahua Lin, Yu Qiao, Luc Van Gool, and Xiaoou Tang. CUHK & ETHZ & SIAT submission to activitynet challenge 2016. *arXiv:1608.00797*, 2016.

[92] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[93] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[94] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Conference on Neural Information Processing Systems*, 2014.

[95] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. In *International Journal of Computer Vision*, 2018.

[96] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

[97] Achal Dave, Olga Russakovsky, and Deva Ramanan. Predictive-corrective networks for action detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[98] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[99] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[100] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[101] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision*, 2015.

[102] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[103] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. *arXiv:1811.12432*, 2018.

[104] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[105] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[106] `https://github.com/Breakthrough/PySceneDetect`.

[107] `https://github.com/johmathe/Shotdetect`.

[108] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, 2016.

[109] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: Boundary sensitive network for temporal action proposal generation. In *European Conference on Computer Vision*, 2018.

[110] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[111] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[112] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection–snip. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3578–3587, 2018.

[113] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017.

[114] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[115] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2016.

[116] Bharat Singh, Hengduo Li, Abhishek Sharma, and Larry S Davis. R-fcn-3000 at 30fps: Decoupling detection and classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1081–1090, 2018.

[117] Ruichi Yu, Xi Chen, Vlad I. Morariu, and Larry S. Davis. The role of context selection in object detection. In *British Machine Vision Conference*, 2016.

[118] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):189–203, 2017.

[119] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[120] Chong Wang, Weiqiang Ren, Kaiqi Huang, and Tieniu Tan. Weakly supervised object localization with latent category learning. In *European Conference on Computer Vision*, pages 431–445. Springer, 2014.

[121] Yi Zhu, Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Soft proposal networks for weakly supervised object localization. *arXiv preprint arXiv:1709.01829*, 2017.

[122] Dahun Kim, Donggeun Yoo, In So Kweon, et al. Two-phase learning for weakly supervised object localization. *arXiv preprint arXiv:1708.02108*, 2017.

[123] Miaojing Shi, Holger Caesar, and Vittorio Ferrari. Weakly supervised object localization using things and stuff transfer. *arXiv preprint arXiv:1703.08000*, 2017.

[124] Douglas H Clements. Subitizing: What is it? why teach it? *Teaching children mathematics*, 5(7):400, 1999.

[125] Prithvijit Chattopadhyay, Ramakrishna Vedantam, Ramprasaath RS, Dhruv Batra, and Devi Parikh. Counting everyday objects in everyday scenes. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[126] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *British Machine Vision Conference*, 2009.

[127] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[128] Ranjay A Krishna, Kenji Hata, Stephanie Chen, Joshua Kravitz, David A Shamma, Li Fei-Fei, and Michael S Bernstein. Embracing error to enable rapid crowdsourcing. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 3167–3179. ACM, 2016.

[129] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[130] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision*, 2017.

[131] Yitian Yuan, Tao Mei, and Wenwu Zhu. To find where you talk: Temporal sentence localization in video with attention based location regression. *arXiv preprint arXiv:1804.07014*, 2018.

[132] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564, 2016.