DETECTION AND INCREMENTAL OBJECT LEARNING IN VIDEOS

A Dissertation
Presented to
The Academic Faculty

By

Ahmad Humayun

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing, College of Computing

Georgia Institute of Technology

August 2018

DETECTION AND INCREMENTAL OBJECT LEARNING IN VIDEOS

Approved by:

Professor James M. Rehg
School of Interactive Computing
Georgia Institute of Technology

Professor Irfan Essa
School of Interactive Computing
Georgia Institute of Technology

Professor James Hays
School of Interactive Computing
Georgia Institute of Technology

Professor Abhinav Gupta
School of Computer Science
Carnegie Mellon University

Professor Bernt Schiele
Max-Planck-Institut für Informatik
Saarland Informatics Campus

Date Approved: May 18, 2018

Table of Contents

List of Tables

List of Figures

SUMMARY

Unlike state-of-the-art batch machine learning methods, children have a remarkable facility for learning visual representations of objects through a combination of self-directed visual exploration and access to a sparse supervisory signal in the form of spoken object names. Studies of infant development have shown that children are able to locate, track, and differentiate novel object instances from a continuous sequence of visual inputs without requiring dense object labels. This thesis develops methods for on-line visual learning in video which are inspired by infant object learning and are enabled by recent advances in deep learning architectures. We introduce two methods and a dataset to support this thesis.

First, we demonstrate a convolutional neural network for detecting and tracking objects in continuous video. These detections are generated by harnessing the temporal continuity of the visual world, and can be used as space-time trajectories for objects in the scene. This method is capable of generating space-time proposals from streaming video, which presents a starting point for on-line weakly-supervised learning. We show that a network can be trained to detect objects more reliably when given a sequence of frames, while being 2.5 times faster when compared to traditional single frame detectors.

The second part of this thesis studies the incremental learning paradigm in a setting similar to an infant's play environment. To mimic an environment where children pick up, examine, and put down different objects, we develop a novel data generation pipeline which can produce an arbitrary number of learning exposures composed of videos of rotating objects. Enabled by this data generator, we introduce a novel object learning problem, known as self-directed incremental learning, where an agent needs to decide whether a learning exposure corresponds to a previously-seen object or a new object. We present a simple solution to this problem, which has the ability to work with 100 unique objects shown repeatedly to the learner. From our extensive experiments we conclude that the effect of catastrophic forgetting, the main obstacle in adapting batch learning algorithms

to an incremental learning setting, is diminished when learners are repeatedly exposed to different views of the same object.

Chapter 1

Introduction

Consider a child learning to recognize different objects in its environment. Every new object they see presents unique visual stimuli which the brain uses to distinguish it from other categories. Moreover, *sometimes* the child is told by a caregiver what a certain object is called, allowing them to name it whenever it appears in future. Over time they continuously learn new object concepts, with names which help build their vocabulary. Even though these capabilities seem unremarkable to human observers, it employs elaborate mechanisms in the brain, which are yet to be fully understood by cognitive and neuroscientists. This type of learning is certainly remarkable knowing that no vision algorithms exist which can mimic this feat given natural videos.

The main goal of this thesis is to replicate the ability of infants to continuously learn new object concepts. This learning is enabled by two important perceptual abilities. Firstly, studies in cognitive science have shown that the ability to localize objects happens quite early in the development of visual perception [1, 2]. This helps infants build rich visual representations of objects. This learning is further helped by the capability for smooth pursuit from a young age — the capacity to track objects without having to identify (name) them [3]. These findings suggest that the ability to localize and track objects is necessary for learning new object concepts.

The thesis makes three contributions in the direction of continuous object learning. We first develop an unsupervised algorithm to produce object segmentation proposals in images. This technique can be extended by stitching proposals in time to generate candidate object segmentations over the extent of the video. The second contribution is a supervised approach to localize objects in space-time, *i.e*. find and track objects in video. Since it is trained on a fixed category set, it can classify objects — resulting in a complete video object detection algorithm. The last contribution of this thesis is a detailed study of infant learning in a play environment where an agent picks up, examines, puts down different object over time. We explore two different paradigms in this setting: (1) where labels are available for each object; and (2) where the only supervision available to the learner is when was an object picked up or put down. A discussion about these methods follows the thesis statement below.

> *Videos provide an effective signal for detecting objects in space-time, as well as in categorizing objects in a self-directed learning problem.*

1.1 Locating, Tracking, and Identifying Objects in Video

Recently object proposals and deep learning have been successfully combined for detection in images [4, 5]. A natural question is whether these ideas can be extended to video. Interestingly humans learn to locate and identify objects from continuous sensory input - which suggests that a machine might learn a better object detector if it integrates information over time. To test this hypothesis, two ideas are explored in Chapter 3. (1) Whether long-range motion is useful for spatio-temporal localization of objects; and (2) if video proposals can be accurately classified into object categories using a convolutional neural network, harnessing both motion and appearance information. This method would be compared with a more traditional approach to generate video object detections, where single frame detec-

tions are stitched in time. Chapter 2 would describe one part of such an approach, where an unsupervised energy minimization scheme is used to produce image proposals. These proposals can either be directly classified into categories, or used to generate object tracks using a least squares formulation [6], and then classified into categories using some aggregate features.

The convnet video object detection algorithm is motivated by experiments in neuroscience, and psychophysics, which show that motion is valuable for recognition tasks. In early 1970s, Johansson [7] demonstrated that our visual perception is adept at recognizing humans and their different actions given only motion stimuli, which he termed as "biological motion". By just seeing 10-12 dot patterns placed over body joints, viewers not only recognized human movement but also differentiated nuances such as a tired or an elastic gait. It is also known that neurons which respond to biological motion lie in the primate temporal cortex, which is the same neural region responsible for object recognition [8]. Other studies point out that motion is critical for grouping objects and recognition in early stages of human visual development [2]. An interesting application of these ideas was explored by Mitra *et al*. [9] in designing CAPTCHA sequences where objects can only be recognized when viewed in motion - whereas objects remained elusive if a single frame is seen in isolation. In all these experiments, most, if not all, appearance information (color/texture), useful for recognition, was removed in order to reveal the importance of motion.

Despite strong precedent from biological visual systems, few algorithms use motion for object localization or detection. On the other hand, single image object detection has advanced remarkably in the last three years, largely due to the resurgence of supervised learning on deep architectures [10, 4, 11]. To the best of my knowledge, these ideas have yet to be explored for detection in video - a problem where motion could be critical in achieving success.

Chapter 3 introduces a supervised pipeline to exploit motion for detecting objects in

video. The method makes use of ImageNet Vid [12] amd ImageNet Detection [13] datasets to train a fully convolutional neural network to generate space-time object detections. Given input of a fixed number of frames, the network aims to produce detections at multiple locations and scales. This is achieved with a network similar in spirit to the Single Shot MultiBox Detector [14]. Motion would be exploited for inferring objectness and localization at each receptive field, by making use of 3D convolutions. The chapter also extends the notion of anchors [5] to space-time.

1.2 Increment Learning of Objects in Video Inspired by Infant Learning

Consider a robot being manufactured with preloaded knowledge of a small number of object classes. For it to perform well in specific environments, it might need to learn about a different set of object categories. A robot might know how to recognize a spoon and a fork, but the end user might want to train it to recognize chopsticks. As argued before, infants perform this function by continuously learning new object categories in their daily life. For a robot to mimic this capability, it would need to seamlessly learn new object categories whenever the environment provides a learning signal. A batch learning method would not suffice here since storing all training samples over time would be infeasible. The learning would need to be done on-line, where the robot just adjusts decision boundaries given any new training samples.

We explore the issues surrounding on-line learning in a settting inspired by an infant's play environment. Infants learn about objects through their repeated manipulation of a relatively small set of toys and other household items. They acquire object knowledge through play, picking up, examining, and putting down objects, and thereby creating a sequence of learning exposures, each of which consists of a contiguous sequence of frames from a single rotating object. We argue that the temporal nature of this input not only

provides important learning signals, but also makes the seamingly unsurmountable task of un-supervised learning tractable.

We introduce a novel object learning problem, known as self-directed incremental learning, which is inspired by recent findings in developmental psychology. To experiment with different algorithms in this paradigm, we present a data generation system which can produce arbitrary numbers of learning exposures through computer graphics rendering. We present a method for incremental object learning which solves the problem of deciding whether a learning exposure corresponds to a previously-seen object or a new object. We provide extensive experimental results that characterize the difficulty of the self-directed incremental learning problem and highlight its relationship to existing incremental learning methods.

1.3 Thesis Proposal Organization

The thesis is organized over three chapters. The first two chapters discuss how to generate image object proposals in an unsupervised setting, and video object detections in a supervised setting. The last chapter discusses the paradigm of incremental learning in videos of moving/rotating objects, explored in a synthetic infant play environment.

Chapter 2 — Unsupervised Object Proposal Generation: This chapter discusses published research [15] on an unsupervised technique to generate object proposals in an image. It uses a parametric min-cut formulation to generate object segmentations at increasing scales at different seed locations. The research identifies the 'middle child' problem which hampers the ability of parametric min-cut in generating proposals of all sizes. It demonstrates a simple solution to adjust CRF unaries as a function of the geodesic distances on a superpixel graph.

Chapter 3 — Video Object Detection: In this chapter I demonstrate an end-to-end trainable pipeline to generate space-time object detections. I would demonstrate a fully con-

volutional neural network which takes a fixed number of frames as input, and generates video object detections over the given time extent. These detections can be seen as short tracklets, which are stitched together across consecutive time windows by simple bipartite matching to generate object tracks over the extent of the video.

Chapter 4 — Incremental Learning of Objects Inspired by Infant Play: This chapter discusses ideas for incremental learning of object categories in an environment inspired by studies from developmental psychology. To enable this research, we developed a data generation pipeline which can produce an arbitrary number of video sequences of toy-like rotating objects. We call each of these sequences as a learning exposure. In this chapter we study the effects on different incremental learning algorithms when given a single learning exposure to each object instance, as well as multiple exposures to the same object instance. We also introduce and exhaustively study the novel problem of self-directed incremental learning — which we propose as the un-supervised learning problem in an infant play setting. In this paradigm the learner is only aware of when an object was picked up or put down, and the goal of the algorithm is to infer whether a learning exposure contains a new object or one seen before. We demonstrate a simple, yet effective, solution to this self-directed learning problem, which is able to categorize a large number of objects.

Chapter 2

Unsupervised Object Proposals from Improved Seeds and Energies

Locating objects is an important task for any visual perception system. If an agent wants to manipulate an object, it not only needs to know what the object is, but also where it is. The importance of localization (locating objects) is apparent from studies of the mammalian brain. Cells throughout different stages of the human visual cortex organize information retinotopically - adjacent cells in visual cortex correspond to adjacent locations on the retina - allowing humans to spatially localize objects in their visual field [16]. Also, studies have revealed that areas of the brain responsible for object categorization are also cognizant of object location [17], which suggests the need for localization to perform object recognition. One goal of computer vision is to build systems which can act in the visual world, which makes localization an important mid-level representation for many algorithms.

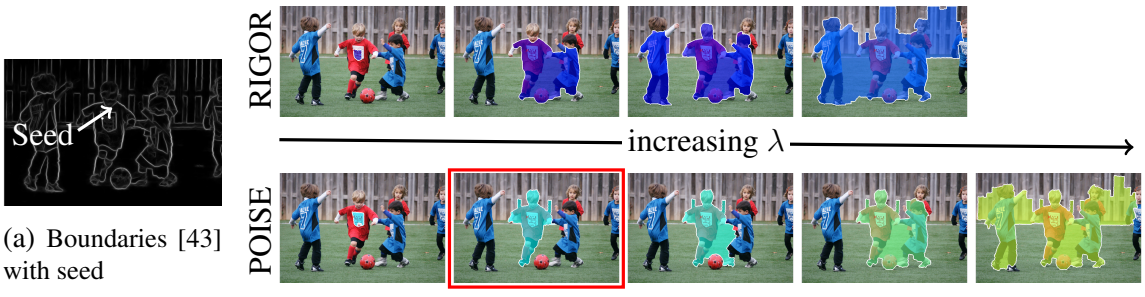
Object detection and semantic segmentation are two critical tasks in computer vision which require localization. Object detection refers to categorizing as well as locating objects, whereas, semantic segmentation refers to pixel-wise categorization of a scene - which needs accurate localization of boundaries. Many of the object detection [18, 4, 19, 20, 21, 22, 23, 24] and semantic segmentation [18, 25, 26, 22] pipelines attempt to localize objects

using a figure-ground object proposal method [27, 28, 29, 30, 31, 32, 33]. These proposal methods were introduced as a replacement for sliding window schemes [34], to generate a smaller pool of possible object locations. Some proposal algorithms can even retrieve the shape of an object, which can help recognition tasks.

Malisiewicz and Efros [34] were the first to suggest generating a large pool of proposals for recognition. Amongst the ensuing methods, Constrained Parametric Min-cut (CPMC) [27] was one of the successful techniques to generate segment proposals. The algorithm produced segments by *parametric min-cut (PMC)* where a few select seed locations are taken as priors for object support. Over recent years many detection pipelines based on convolutional neural networks have adopted object proposals for localization [18, 4, 19, 20, 23]. This has spurred many new proposal methods, each aiming to recall more objects from a smaller pool of localizations. Some of them generate segments by energy minimization via graph-cuts [35, 36, 37, 29]. Other popular methods perform agglomerative clustering [30, 31], or employ edge-based techniques [38, 32] to generate proposals.

Some object proposal methods generate bounding box localizations, while others produce segmentations. Studies have shown merits of segmenting objects for recognition tasks. Malisiewicz and Efros [34] demonstrated that segmentation can result up to 15% gains in classification performance on a pre-deep-learning pipeline. Gu *et al*. [39] also showed the effectiveness of segment based features for both detection and classification. More recently, the efficacy of using segments with convolutional neural network features has been demonstrated for detection [20] and for the task of 'simultaneous detection and segmentation' [22]. Part of the reason for this effectiveness is the explicit separation in representing the object from its context. In light of this evidence, we aim to generate high quality segment object proposals.

We choose a discrete energy minimization approach to generate segment proposals. Our study shows that this approach has the potential to generate competitive results in an unsupervised setting compared to current CRF models, but only if the energy potentials



(a) Boundaries [43] with seed

(b) Parametric min-cut from the seed given in (a)

Figure 2.1: This example demonstrates the *middle child problem*. The seed is placed on the child's jersey, as shown in a. The probabilistic boundary map suggests that it should be possible to produce a segment containing just the child in red. Each row shows parametric min-cuts produced by a method from the displayed seed. The top row demonstrates that RIGOR [35] is incapable of finding this segment and the bottom row shows our results. Unlike RIGOR, we are able to capture the *middle child* (red outlined).

in the model are carefully designed. This technique fits in well with the long history of graphical models in obtaining elegant, yet effective solutions to hard vision problems [40].

We have observed empirically that PMC tends to produce results of extreme size: segments are either similar in size of the seed, or they tend to span almost the full image (see Fig. 2.1 for an example). Segments which are in the middle which often correspond to particularly salient object candidates are frequently missing, and therefore do not get the attention that they deserve. We refer to the absence of these segments as the *middle child problem*. We will demonstrate that this problem is a natural result of energy potentials used in existing PMC formulations [27, 36, 35] and cannot be solved simply by tuning parameters or exploring breakpoints exhaustively [41, 42]. The middle child problem significantly limits the performance of PMC for generating high quality object proposals.

We show why PMC is susceptible to the middle child problem, and propose a simple solution to produce better proposals over the whole size range. PMC generates increasingly large segment proposals as the value for the parameter λ grows [42, 27, 35]. The goal is to control the growth of proposal size as λ increases. We achieve this by tying the unary potentials of image superpixels to the geodesic distance to the current segment. This facilitates the generation of medium-sized segments by lowering their energy at particular

λ values. The approach does not break any necessary condition in the PMC formulation, thereby maintaining the nesting property [42] of the segments produced. The resulting algorithm can generate $\sim$1,000 proposals in $\sim$3.5 seconds.[1]

The paper also introduces a new superpixel merging algorithm for generating seeds. It utilizes an adaptive appearance thresholding strategy to generate a hierarchy of superpixels of varying sizes, so that more superpixels are generated in regions that have more internal variation and less are generated in regions with uniform color. This approach generates a small set of reliable seeds that cover objects of all sizes and diverse appearances, and improves on previous algorithms for small and less salient objects.

These improvements result in a state-of-art object proposal algorithm. Our method requires many fewer proposals than its competitors to obtain the same accuracy. The performance of our algorithm is validated on two segmentation benchmarks: PASCAL VOC and MS COCO [44].

This chapter extends our earlier work [15] in a number of ways. First, we devise a novel metric in §2.4 to measure the difficulty in segmenting a particular object using a general PMC formulation without our solution. This allows us to measure how much each ground-truth suffers from the middle child problem, and verify whether POISE is more capable in capturing such objects. We also show our method's performance in retrieving bounding box proposals, which is similar to the analysis in [45]. This paper also extends evaluation of POISE on MS COCO and adds experiments comparing average recall at different object sizes. Furthermore, we give implementation details of our algorithm.

In §2.1 we review proposal generation methods, and their role in detection pipelines. §2.2 explains in detail the causes and effects of the middle child problem. §2.3 gives an efficient solution to the problem. §2.4 introduces a method to measure if an object would be effected by the middle child problem. Our superpixel seed generation method is explained in §2.5. This is followed by evaluation in §2.7, which quantitatively demonstrates the

[1]Multi-threaded run-time on Intel i7-3930K. Code is available on-line at `http://rehg.org/poise`.

effects of each of our contributions. We conclude in §2.8.

2.1 Related Work

Convolutional neural networks (CNN) have been leading the progress in object detection [18, 4, 22], and part of their success can be attributed to their use of object proposals. Before proposal methods, it was common for classifiers to exhaustively test $\sim 10^6$ sliding window locations [46, 47]. Object proposal methods [30, 27, 32] provide a more manageable set of regions, which in most cases is < 5K. Given a smaller set of regions, it becomes feasible to apply more complex classifiers, increasing accuracy. Recent experiments have also shown that using proposals can reduce false positives in a class-specific object detector like DPM [48].

Malisiewicz and Efros [34] were the first to suggest generating a large pool of proposals for recognition. In the same paper they demonstrated that segmentation can help increase classification accuracy.

2.1.1 ConvNet based Proposals

Over recent years, the winning methods for object detection [13, 44] and semantic segmentation [49, 50] have been based on convolutional neural networks. Some of the leading methods explicitly use proposals to find possible object locations [18, 4, 19, 20, 21, 22, 23, 24]. Following this trend, there have been efforts to generate proposals directly by a convolutional neural networks.

Szegedy *et al*. [21] uses a convnet to generate a fixed number of object proposal boxes. It uses an inception like architecture [51] to predict a predefined number of bounding box coordinates and an objectness score. Faster R-CNN [5] reduces the number of parameters to build a proposal network by using a fully convolutional network. Here, each location in the last convolutional feature map is used to generate multiple boxes, each estimating an

object location at different scale and aspect ratio.

DeepMask [33] was the first paper to generate segmentation proposals from a discriminatively trained deep convnet. Its network architecture includes a pre-trained VGG model [52] which extracts features shared by a segmentation and an objectness scoring branch. Due to multiple pooling layers, segments from DeepMask tend to not stick to true object boundaries. SharpMask [53] fixes this by having multiple refinement stages, each attempting to invert the effect of the pooling layer by learning to double the resolution of the mask generated by the layer above.

Unlike DeepMask and SharpMask, instance-sensitive fully convolutional networks [54] has no final layer to directly infer a segmentation mask of an object. Rather it uses score maps which are sensitive to locations on an object (for instance, one score map would activate at top-left positions of objects). A parameter-less assembling process at the end allows them to generate proposals at each image location.

2.1.2 Multiple Segmentations

Before the introduction of object proposals, a few methods generated multiple segmentations as a preprocessing step to dilineate possible objects. Both techniques generate possible localizations of objects in a scene by providing spatial support for better inference results. The main difference between the two methods is in terms of constraints placed on results - when generating multiple segmentations the result is different partitionings of the image space, whereas segmentation proposals is allowed to generate an arbitrary number of figure-ground segments.

Russell *et al*. [55] uses multiple segmentations for discovering (clustering) objects from a large image corpus. Hoiem *et al*. [56] finds multiple segmentations useful for generating 3D layouts from a single image. Both these papers are motivated by the idea that no bottom-up segmentation would produce correct results for the whole scene, but some segments in some segmentations would capture objects accurately. By scoring each segment for the

task at hand, good segments can be sampled from the pool of image partitionings.

Proposal generation methods either produce bounding boxes [57, 48, 58], or segments [36, 27, 30]. Recent work [18, 59] argues for latter by demonstrating that segmentation-based features significantly increase the mean accuracy on both segmentation and detection challenges in Pascal VOC [49]. Their experiments indicate that both object shape and context are useful for recognition.

Encouraging results for detection have recently spurred new proposal methods. Selective Search [30] is one of the more popular methods and is based on grouping. It performs hierarchical merging of superpixels with different metrics, producing a diverse set of proposals. Yanulevskaya [60] and Bonev *et al*. [31] improve Selective Search by guiding the hierarchical grouping process. Yanulevskaya [60] replaces the metrics measuring similarity between regions with a random forest which selects regions to merge at a certain level. Similar to Selective Search, Bonev *et al*. [31] constructs regions by hierarchical grouping, where segment merging is guided by the PageRank algorithm. They also measure the gain in entropy for each merge to find a small set of final proposals over all hierarchy levels. Instead of grouping by various metrics, our method segments objects by finding global minima of an energy function defined on superpixels. This is similar to other methods performing maximum a posteriori (MAP) inference by graph-cuts for proposal generation [27, 35, 36, 37, 29]. Rantalankila *et al*. [37] produces object segments by performing graph-cuts after locally merging superpixels.

Recently, there have been some attempts to produce proposals by supervised learning. Krähenbühl and Koltun's GOP [38] is a level-set method which produces proposals by thresholding a signed geodesic distance transform. They also make use of supervised regression to find good locations to place seeds. Krähenbühl and Koltun's LPO [29] generates regions from CRF models trained on VOC. We demonstrate better performance than LPO without training any models for proposal generation. Pinheiro *et al*. [33] introduced a CNN trained on COCO to generate segment proposals. POISE's segment boundaries

appear to be qualitatively better than [33], which loses spatial accuracy due to the pooling layers.

The last two years has also seen significant improvement in edge detection [61, 43]. Making use of accurate structure predicted edges, Zitnick and Dollár [57] introduced a method to score whether a bounding box completely contains an object. This is simply done by comparing the number of edge segments contained within to the ones crossing the boundaries of the bounding box. Even though they output bounding box proposals, the set of edges belonging to a high scoring box can be used as a cue for generating segments (see Fig. 1 in [57]).

We refer the reader to Hosang *et al*. [45] for an excellent review of proposal methods. In §2.7 we evaluate several methods using the average recall metric which was introduced in their work.

There have been some efforts to perform object detection and segmentation simultaneously. Hariharan *et al*. [22] extract CNN features on proposals, which are used for classification and segment refinement. Dong *et al*. [59] take a different approach to the problem. They select semantic segments in the image where sliding window detectors and a segment hypothesis method seem to be in agreement.

One main contribution of this chapter is the use of geodesically guided PMC to solve the *middle child problem*. Kolmogorov *et al*. [42] review PMC applications in vision. They demonstrate how PMCs can be used to solve some geometric functionals. Lim *et al*. [62] deal with more general constraints to produce accurate segments, when some ground-truth statistics are available. [63] discusses generating more solutions by decomposing the image. Certainly these methods could be useful for generating proposals, but they typically produce a segment in the order of seconds. Batra *et al*.'s work on Diverse M-Best [64] obtains highly probable solutions beyond MAP by Lagrangian relaxation in MRF models. This is related to our approach, since both methods change unary costs after obtaining the first optimal solution. On the other hand, exemplar-cut [65] changes energies to push solu-

tions toward exemplars. Both these approaches [64, 65] adjust energies to direct solutions away or towards existing solutions/exemplars, whereas we adjust the energy to encourage a more complete set of solutions.

2.2 The Middle Child Problem

This three part section defines and explains the middle child problem in PMC for segmentations. We start by introducing the PMC energy and the equivalent graph. In the second part, we illustrate why the problem exists using a simple model with 3 regions. We generalize this model in the third section, and show that the problem remains. Our example images are constructed from concentric regions which mimics the compositional nature of objects.

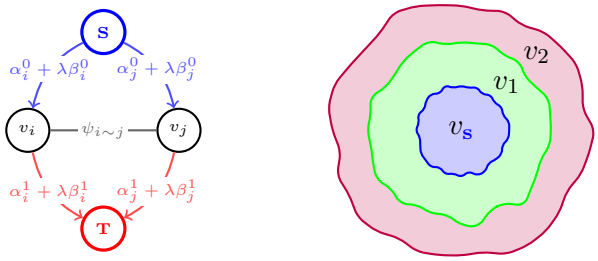
Generating Proposals by PMC: Our algorithm uses graph-cuts from multiple seeds to compute segments. For each seed, a directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is created with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Using this graph, we construct and minimize the Quadratic Pseudo-Boolean (QPB) function,

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left(\theta_i^1 x_i + \theta_i^0 \overline{x}_i \right) + \sum_{(i,j) \in \mathcal{E}} \left(\psi_{ij}^{11} x_i x_j + \psi_{ij}^{01} \overline{x}_i x_j + \psi_{ij}^{10} x_i \overline{x}_j + \psi_{ij}^{00} \overline{x}_i \overline{x}_j \right) .$$

The solution is the boolean vector $\mathbf{x} = [x_1, \ldots, x_n]$. θ_i^ℓ is the unary potential associated with variable v_i when it takes the binary label ℓ. The pairwise potential, $\psi_{ij}^{\ell\eta}$, is used when variables v_i and v_j take binary labels ℓ and η respectively.

Since we use Potts energy, where $\psi_{ij}^{01} = \psi_{ij}^{10}$ (which we will denote as $\psi_{i \sim j}$), we can simplify the function to

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left(\theta_i^1 x_i + \theta_i^0 \overline{x}_i \right) + \sum_{(i,j) \in \mathcal{E}} \left(\psi_{i \sim j} \left| x_i - x_j \right| \right) . \tag{2.1}$$



(a) QPB function set up for graph-cut

(b) Example image to illustrate the middle child problem

Figure 2.2: a shows how a QPB function is represented as a graph, where a min-cut would minimize the function. b is an example image to demonstrate the middle child problem. In all graphs, **S** and **T** are the special source and sink nodes used by min-cut.

To generate object proposals, we convert this to the parametric pseudo-quadratic form, where $\theta_i^\ell = \alpha_i^\ell + \lambda\beta_i^\ell$, which can be represented as the graph given in Fig. 2.2a. We denote the resulting parametric energy as $E_\lambda(\mathbf{x})$. The real-valued PMC parameter λ belongs to a sequence $\lambda_0 < \lambda_1 < \cdots < \lambda_L$. The unary potentials are defined by the values α_i^ℓ and β_i^ℓ. Given these parameters, the energy can be readily minimized by max-flow/min-cut. Min-cut produces two disjoint sets S and T, where node $v_i \in S$ iff $x_i = 1$, and $v_i \in T$ iff $x_i = 0$. The cut is defined by the sum of edge weights from S to T, which can be verified to equal the minimization of (2.1). We are interested in the monotonic case for PMC, where $\beta_i^1 < \beta_i^0$, which can be re-parameterized to get non-decreasing source capacities and non-increasing sink capacities with increasing λ [66]. The monotonic case gives solutions with the nesting property, where if $x_i = 1$ for λ_t, it is guaranteed that $x_i = 1$ for $\lambda_{t+1} > \lambda_t$ [42, 41].

We use PMC to produce multiple segments from each foreground seed at various image locations. For seed nodes, $v_{\mathbf{s}}$, we enforce $x_{\mathbf{s}} = 1$ by setting $\alpha_{\mathbf{s}}^0 = \infty$. All remaining nodes are $v_i \in \mathcal{V}\backslash\{v_{\mathbf{s}}\}$, each representing a superpixel. $\mathcal{E}$ is the set of all superpixel pairs which share a boundary.

2.2.1 Simple Model with 3 Regions

To demonstrate the middle child problem, consider the image in Fig. 2.2b with three concentric regions. The center region, v_s is the seed. Following the formulation in [42], we set $\theta_i^0 = 0$. We assume that the unaries on all pixels are a constant, and set $\alpha_i^1 = C$ and $\beta_i^1 = -1$, where C is some constant. Translating unaries from a pixel to a superpixel graph incurs a constant multiplication factor of the size of the superpixel, z_i. The resulting unary potential is $\theta_i^1 = (C - \lambda)z_i$. This mimics a standard (uniform) graph used by CPMC [27], as well as RIGOR [35]. Suppose, $\psi_{s\sim1}$ and $\psi_{1\sim2}$ are the costs associated to the outer boundaries of the blue and green regions respectively. Let us assume $z_s < z_1 < z_2$ and $\psi_{s\sim1} < \psi_{1\sim2}$ (longer boundaries typically have larger capacities).

We can compute the energy of each configuration of the vector $\mathbf{x}$. Since $\alpha_s^0 = \infty$, v_s would always be in the foreground. We check the remaining four configurations:

		v_2	
		$x_2 = 0$	$x_2 = 1$
v_1	$x_1 = 0$	$E(\mathbf{x}) = \psi_{s\sim1}$	$E(\mathbf{x}) = \psi_{s\sim1} + \psi_{1\sim2} + (C - \lambda)z_2$
	$x_1 = 1$	$E(\mathbf{x}) = \psi_{1\sim2} + (C - \lambda)z_1$	$E(\mathbf{x}) = (C - \lambda)(z_1 + z_2)$

For simplicity, we will refer to the solution $x_1 = \ell, x_2 = \eta$ as $\langle \ell\eta \rangle$, and $\langle 10 \rangle$ is the middle child solution. When $\lambda \geq 0$, notice that the $\langle 01 \rangle$ solution will have higher energy than $\langle 10 \rangle$ because $z_2 > z_1$. Furthermore, when $\lambda = 0$, we will get the $\langle 00 \rangle$ solution, *i.e.* only v_s is in the foreground, as long as $\psi_{s\sim1} < \psi_{1\sim2} + Cz_1$ and $\psi_{s\sim1} < (z_1 + z_2)C$. $\min E_\lambda(\mathbf{x}) = \psi_{s\sim1}$ in this case. When $\lambda \geq C$, we will obtain the solution $\langle 11 \rangle$, *i.e.* all the regions are in the foreground, and $\min E_\lambda(\mathbf{x}) \leq 0$.

The key question is whether it is possible to obtain solution $\langle 10 \rangle$ from some real-valued λ? For this to be true, two conditions must hold for some λ:

1. $\psi_{1\sim2} + (C - \lambda)z_1 < \psi_{s\sim1}$ 2. $\psi_{1\sim2} < (C - \lambda)z_2$

These two conditions imply that $E(\mathbf{x})$ for $\langle 10 \rangle$ should be less than the energies of the $\langle 00 \rangle$

and $\langle 11 \rangle$ solutions at some λ. The first condition is discounted by our initial condition $\psi_{1\sim 2} > \psi_{s\sim 1}$, and will only be true if $C < \lambda$. The second condition can be true when $\lambda < C$, implying that we will never obtain the middle segment. In practice, one might obtain the segment in the middle if its boundaries have less total capacity than the boundaries it encloses, *i.e.* $\psi_{1\sim 2} < \psi_{s\sim 1}$. Since in a superpixel graph the image boundary/edge strength is inversely proportional to the pairwise potential $\psi_{i\sim j}$, this condition requires that a medium sized segment boundary must be stronger than its internal boundaries. This is not true in presence of strong internal structure (*e.g.* a striped shirt) in conjunction with weak object edges.

2.2.2 General PMC with $n + 1$ Regions

We now demonstrate that the middle child problem also exists for graphs of more general form with $n + 1$ regions (the seed, $v_{\mathbf{s}}$ contributes the $+1$), as illustrated in Fig. 2.3. We would use capacities from $\mathbf{S}$ and $\mathbf{T}$ as $e_i + \lambda f_i$ and $g_i - \lambda h_i$ respectively, as prescribed in Gallo *et al*. [41]. Here, e_i, f_i, g_i, h_i are all functions of vertex v_i, returning non-negative values. Moreover, $g_i \geq \lambda h_i, \ \forall \lambda$ to disallow negative capacities on sink arcs. We are interested in segments that form a single connected component, growing outward from $v_{\mathbf{s}}$. The aim is to produce all segments $\langle 1 \ldots 10 \ldots 0 \rangle$, where the last 1 happens at index t. This translates to the cut given in Fig. 2.3, which is equivalent to the whole region inside the solid green boundary belonging to v_t. In this section $\psi_t \equiv \psi_{t\sim t+1}$.

First, let us look at the energies of different solutions. For $\langle 0 \ldots 0 \rangle$, where only $v_{\mathbf{s}}$ is in the foreground,

$$E_\lambda(\mathbf{x}) = \psi_{s\sim 1} + \sum_{i=1}^{n} (e_i + \lambda f_i) \ . \tag{2.2}$$

For $\langle 1 \ldots 10 \ldots 0 \rangle$, where the cut passes through ψ_t, and $t \in \{1, \ldots, n-1\}$ (as illustrated

in Fig. 2.3),

$$E_\lambda(\mathbf{x}) = \psi_t + \underbrace{\sum_{i=1}^{t} (g_i - \lambda h_i)}_{\text{Unaries cut from } \mathbf{T}} + \underbrace{\sum_{i=t+1}^{n} (e_i + \lambda f_i)}_{\text{Unaries cut from } \mathbf{S}} \ . \tag{2.3}$$

Similarly, for $\langle 1 \dots 1 \rangle$, the full image solution is

$$E_\lambda(\mathbf{x}) = \sum_{i=1}^{n} (g_i - \lambda h_i) \ . \tag{2.4}$$

To get a middle segment (a segment enclosed by the solid green boundary), the following conditions need to hold:

1. *Eq*. 2.3 should be less than *Eq*. 2.2:

   $\psi_t + \sum_{i=1}^{t} (g_i - \lambda h_i) \ < \ \psi_{s\sim 1} + \sum_{i=1}^{t} (e_i + \lambda f_i)$

2. *Eq*. 2.3 should be less than energies of smaller segments, where $1 \leq k_1 < t$:

   $\psi_t + \sum_{i=k_1+1}^{t} (g_i - \lambda h_i) \ < \ \psi_{k_1} + \sum_{i=k_1+1}^{t} (e_i + \lambda f_i)$

3. *Eq*. 2.3 should be less than energies of larger segments, where $t < k_2 \leq n$:

   $\psi_t + \sum_{i=t+1}^{k_2} (e_i + \lambda f_i) \ < \ \psi_{k_2} + \sum_{i=t+1}^{k_2} (g_i - \lambda h_i)$

4. *Eq*. 2.3 should be less than *Eq*. 2.4:

   $\psi_t + \sum_{i=t+1}^{n} (e_i + \lambda f_i) \ < \ \sum_{i=t+1}^{n} (g_i - \lambda h_i)$

We can think of v_{k_1} as a variable between $v_{\mathbf{s}}$ and v_t. For instance, this could be the variable associated with the orange region surrounding $v_{\mathbf{s}}$ in the Fig. 2.3 image. Similarly, we can think of v_{k_2} as the yellow region surrounding v_t. To make it easier to analyze these constraints, we introduce some new variables. The first set of variables is for the sum $\sum_{i=l_b}^{l_u} (e_i - g_i)$. The following illustration gives variable symbols, each surrounded by two lines. Each variable is for the sum, where l_b and l_u is defined by the labels on the surrounding two lines. For instance, $C = \sum_{i=t+1}^{k_2} (e_i - g_i)$:

$1 \quad A \quad k_1 \quad B \quad t \quad C \quad k_2 \quad D \quad n$

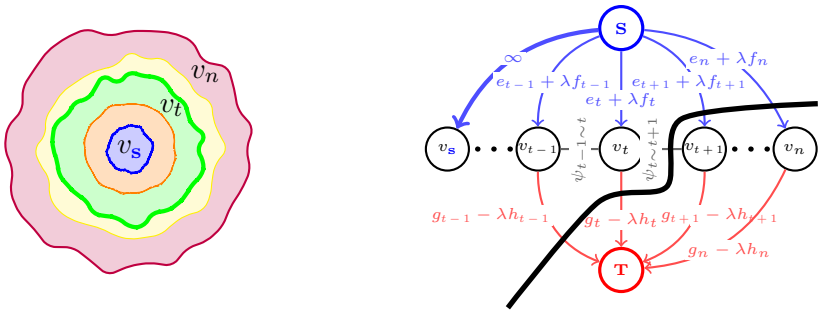


Figure 2.3: Generalization of Fig. 2.2b to multiple middle regions and unaries defined in Gallo *et al*. [41]. The image (left) and the corresponding graph (right) are given. The black curve on the graph shows the $\langle 1 \dots 10 \dots 0 \rangle$ cut, which is equivalent to the segment inside the thick green boundary on the left.

Since, both e_i and g_i are non-negative, all variables A, B, C, D possibly could be negative. The next set of four variables, M, N, P, Q are defined for the sum $\sum_{i=l_b}^{l_u} (f_i + h_i)$. They are defined over the same limits, *e.g.* $N = \sum_{i=k_1+1}^{t} (f_i + h_i)$. Note that these variables can only have non-negative values. Furthermore, for simplicity, we will use $\Psi = \psi_{t\sim t+1}$.

After some simple algebra, and replacing variables, we can convert the four constraints to:

1. $\frac{\Psi - \psi_{s\sim 1}}{M+N} - \frac{A+B}{M+N} < \lambda$
2. $\frac{\Psi - \psi_{k_1\sim k_1+1}}{N} - \frac{B}{N} < \lambda$
3. $\lambda < \frac{\psi_{k_2\sim k_2+1} - \Psi}{P} - \frac{C}{P}$
4. $\lambda < -\frac{\Psi}{P+Q} - \frac{C+D}{P+Q}$

Let us suppose we have no control over the pairwise potentials, and we can only adjust unaries so that λ has a feasible non-negative value that satisfies these constraints. One way to achieve this is to make the L.H.S. in the first two conditions negative, and the R.H.S. in the last two conditions positive. Then, there would be some non-negative λ which will satisfy these constraints. Following this strategy, condition 1 requires $A + B > \Psi - \psi_{s\sim 1}$, and condition 2 requires $B > \Psi - \psi_{k_1\sim k_1+1}$. Moreover, to have positive R.H.S in conditions 3 and 4, we require $C < \psi_{k_2\sim k_2+1} - \Psi$ and $C + D < -\Psi$ respectively.

There are certain conclusions one can draw from this setup. Firstly, functions f_i and h_i have no influence over the chances of obtaining the middle segments. On the contrary, e_i and g_i are essential in obtaining any middle segments. To increase the chances to have a feasible λ, we need $e_i \gg g_i$ where $1 \leq i \leq t$, and $g_i \gg e_i$ where $t+1 \leq i \leq n$. Of course, this cannot be simultaneously true for all $t \in \{1, \ldots, n-1\}$, hence it needs to be adjusted for each individual t. This observation vouches for the geodesics based solution we give in the next section.

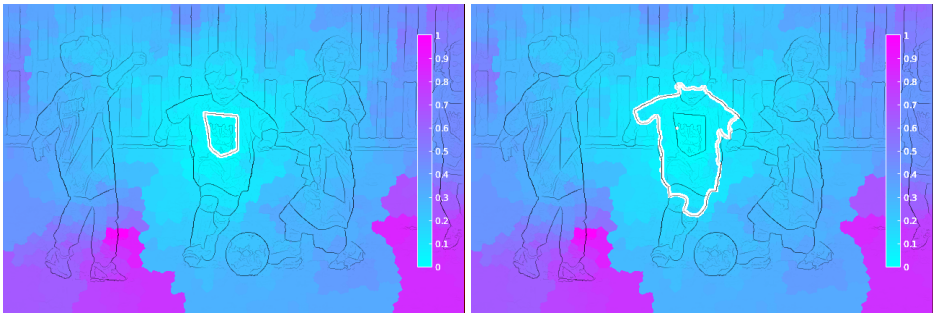
2.3 Biasing PMC for Obtaining Medium Sized Segment Proposals

In the previous section, we identified a problem with the structure of standard graph-cut energies that results in missing medium-sized segments. We propose to solve this problem by biasing the solutions in a sequence of optimizations to obtain segments which are close to the last cut. These optimizations are performed on a fixed set of PMC parameters $\lambda_0 < \lambda_1 < \cdots < \lambda_L$. The parameter λ_l is used in minimizing $E_{\lambda_l}(\mathbf{x})$ to produce $\mathbf{x}_l = [x_1^{(l)}, \ldots, x_n^{(l)}]$. Given the solution $\mathbf{x}_l$, we want to set the unaries in way that minimizing $E_{\lambda_{l+1}}(\mathbf{x})$ produces only a slightly larger segment $\mathbf{x}_{l+1}$. This requires the energy of *Eq*. 2.4 to be larger than *Eq*. 2.3.

To enforce these constraints for obtaining segments $\mathbf{x}_{l+1}$ which are slightly larger than $\mathbf{x}_l$ (the last parametric solution), we change our unaries to the following form:

$$\alpha_i^\ell + \lambda_{l+1}\beta_i^\ell + f_i(\mathbf{x}_l) \tag{2.5}$$

This additional term $f_i(\mathbf{x}_l)$ guides the PMC to produce segments of all sizes. The function needs to be designed such that it raises the source unaries, θ_i^0, for superpixels which are spatially close to the last cut. Similarly we would like to raise the sink unaries, θ_i^1, for



(a) $\mathbf{x}_l$ overlayed on $\phi_i(\mathbf{x}_{l-1})$ (b) $\mathbf{x}_{l+1}$ overlayed on $\phi_i(\mathbf{x}_l)$

Figure 2.4: This shows the geodesic distances $\phi_i(\mathbf{x}_*)$, which is used to bias unary potentials to produce medium sized segments. The resulting cut is overlayed on each figure as a white boundary. Note that all superpixels where $x_i^{(l)} = 1$ (cut in a), the next computed $\phi_i(\mathbf{x}_l) = 0$ (color in b), since now it is inside the previous cut.

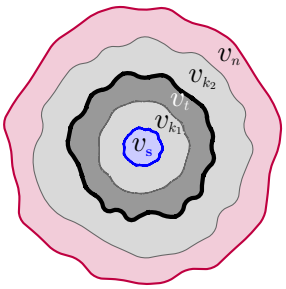
superpixels which are further away. Such a scheme would ensure that the energy of solutions that are slightly larger than the last cut decreases in comparison to segments which are much larger.

We find that the geodesic distance between superpixels is a good metric to guide our PMC. To construct $f_i(\mathbf{x}_l)$, we compute geodesics on an undirected graph with edge weights given by image edge strength - so two superpixels sharing a weak edge have a short geodesic distance. We precompute the $n \times n$ all-pairs shortest paths g_{ij}. In performing PMCs, for each variable we can retrieve the minimum shortest path to any superpixel in the last cut $\mathbf{x}_l$, i.e.

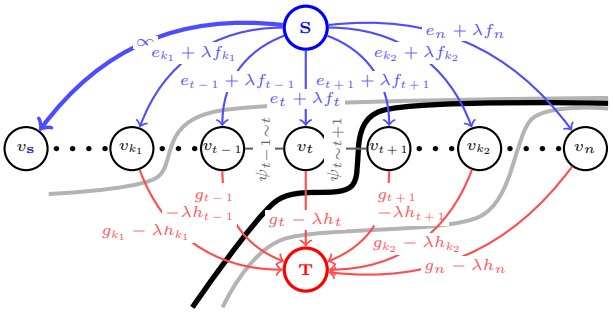
$$\phi_i(\mathbf{x}_l) = min_{j \in \mathcal{V} \,:\, x_j^{(l)}=1} g_{ij} \tag{2.6}$$

This is visualized in Fig. 2.4 for two consecutive cuts. We finally compute $f_i(\mathbf{x}_l) = h(\phi_i(\mathbf{x}_l))$, where $h(\cdot)$ is a linear function, allowing us to raise source unaries if $\phi_i(\mathbf{x}_l) < \tau$, and raise sink unaries if $\phi_i(\mathbf{x}_l) > \tau$. We empirically tune the geodesic threshold, τ, on the VOC'12 training set.

In our experiments, we noticed that the raw geodesic distance can be adversely affected by leaks in object boundaries. Ideally, one would like to compute the K shortest paths between any two superpixels, in order to avoid using erroneous boundaries. Since such a



(a) Example image to illustrate the middle child metric



(b) The cuts used for computing the middle child metric

Figure 2.5: This graph illustrates the cuts involved in computing the middle child metric. The graph construction is similar to Fig. 2.3.

scheme would be expensive to compute, we resort to dropping 50% of the weakest edges in the superpixel graph before computing the geodesic distances. Since dropping the weakest edges can disconnect the graph, we avoid dropping edges in the graph which belong to a maximal spanning tree.

In practice, medium sized segments lie typically between 400 to 4,000 pixels. Our experiments demonstrate (Fig. 2.9) that *our solution is superior to all others in this regime*.

2.4 How Hard is it to Obtain the Middle Child Solution?

It is useful to construct a metric on how "middle child" a particular segment is, i.e. how difficult would it be to obtain a particular segment. Let us reconstruct the argument given in §2.2.2. Given that we can obtain a cut through $\psi_{k_1 \sim k_1+1}$, and a larger cut through $\psi_{k_2 \sim k_2+1}$ (at two different paramaters $\lambda_{k_1} < \lambda_{k_2}$), we want to find how hard is it to obtain a cut through $\psi_{t \sim t+1} : k_1 < t < k_2$. This is depicted in Fig. 2.5, where we are trying to find how hard is it to get the cut in black, given that we have both the cuts in grey: Again, our goal is to get the middle child segment $\langle 1 \ldots 10 \ldots 0 \rangle$, where the cut passes through $\psi_{t \sim t+1}$, and $t \in \{1, \ldots, n-1\}$, given that we have a smaller and a larger cut (given in grey). Hence, we can write the two conditions which would achieve this middle child solution :

1. The energy of middle child solution should be less than the smaller segment at λ_{k_1}:

$$\psi_{t\sim t+1} + \sum_{i=k_1+1}^{t} (g_i - \lambda h_i) < \psi_{k_1\sim k_1+1} + \sum_{i=k_1+1}^{t} (e_i + \lambda f_i) \tag{2.7}$$

2. The energy of middle child solution should be less than the larger segment at λ_{k_2}:

$$\psi_{t\sim t+1} + \sum_{i=t+1}^{k_2} (e_i + \lambda f_i) < \psi_{k_2\sim k_2+1} + \sum_{i=t+1}^{k_2} (g_i - \lambda h_i) \tag{2.8}$$

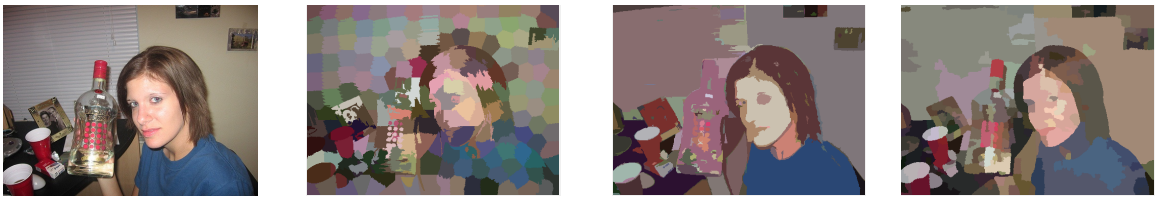
We can combine both these conditions, to see what λ would allow us to obtain the "middle child" cut:

$$\begin{aligned} \frac{\psi_{t\sim t+1} - \psi_{k_1\sim k_1+1} + \sum_{i=k_1+1}^{t} (g_i - e_i)}{\sum_{i=k_1+1}^{t} (f_i + h_i)} &< \lambda \\ < \frac{\psi_{k_2\sim k_2+1} - \psi_{t\sim t+1} + \sum_{i=t+1}^{k_2} (g_i - e_i)}{\sum_{i=t+1}^{k_2} (f_i + h_i)} & \end{aligned} \tag{2.9}$$

If the gap between these inequalities is ≤ 0 then the middle child solution is not obtainable. We can shorten these inequalities to $\zeta_{k_1}^{t} < \lambda < \zeta_{t}^{k_2}$. Let's define a function $\Gamma_{k_1,t,k_2} = \zeta_{k_1}^{t} - \zeta_{t}^{k_2}$. If $\Gamma_{k_1,t,k_2} < 0$, there exists a λ from which we can obtain a middle child solution. Hence, Γ_{k_1,t,k_2} gives a measure of how hard it is to obtain the middle child solution.

For each ground-truth object segment, we can find a seed superpixel $v_{\mathbf{s}}$ which is completely contained in the object. Using the parametric cuts produced from this seed, we can find the last segment which is completely contained inside the ground-truth segment - let us call this segment c_{k_1}. Similarly, we can find the first segment which completely contains the ground-truth segment - let us call this segment c_{k_2}. Since we have the ground-truth, we can find the superpixels combination which has the highest overlap with the ground-truth. This segment can be represented by the cut $c_{k_*} : k_1 < k_* < k_2$. We can compute the hardness of getting this solution by measuring Γ_{k_1,k_*,k_2}.

Amongst all the parametric min-cuts, we can find the cut $c_t : k_1 \leq t \leq k_2$ which



Original Image SLIC Superpixels [67] FH Superpixels [68] Our Superpixels

Figure 2.6: Illustration of superpixel results. All algorithms produce 190 ± 1 superpixels (according to the default settings of FH [68]). SLIC [67] regularization is 0.05 (higher regularization would lose more detail). Superpixels are colored by their mean color plus a small random perturbation to reveal the differences among ones with similar colors. SLIC severely lacks detail by spending the budget evenly across the image. The FH algorithm produces many superpixels on very small textures and some superpixels are highly irregular in shape. Our merging method largely alleviated the problems of FH, and hence can represent more meaningful parts (e.g., the bottle cap, the mouth of the person) while preserving boundaries more effectively.

has the highest overlap with the ground-truth segment. This highest overlap can be used to guage the performance of the algorithm.

2.5 Segment Seeds from Merging Superpixels

Careful seed placement is important for good object proposal performance. In order to capture the majority of objects with a small number of proposals, it is preferable to place fewer seeds in regions with more uniform color and more seeds in regions that have more internal variation. Previously, seeds have been placed on all superpixels [28], a regular grid [27, 35], via diversified optimization [38], etc.

We propose seeding based on a hierarchical merging of watershed superpixels. Since watershed superpixels already combine areas with uniform color, it offers a nice starting point for obtaining different spatial resolutions in different areas. Our merging process is considerably faster than many optimization approaches, as only very simple operations are involved. In principle, any merging algorithm can be used, but we propose a new superpixel merging algorithm. The new algorithm is similar with the widely used Felzenszwalb-Huttenlocher (FH) algorithm [68], but with an adaptive thresholding scheme to improve

the regularity of the superpixels in creating a hierarchy.

A basic idea, similar to FH, is that when generating superpixels of different sizes, smaller superpixels should be merged together unless they have very distinctive appearance. On the other hand, two large superpixels should not be merged when they have a moderate difference in appearance. We implement a novel adaptive thresholding scheme for this purpose. At each iteration, a "desired superpixel size" S_d is computed to set the adaptive threshold. S_d is initialized to $\frac{S}{N_d}$, where S is the number of pixels in the image and N_d is the user-specified desired number of superpixels. In subsequent iterations, S_d is chosen to satisfy:

$$S_d\Big(N_d - \sum_i \mathbb{I}(|v_i| > S_d)\Big) = S - \sum_{v_i, |v_i| > S_d} |v_i| \tag{2.10}$$

where $|v_i|$ represent the size of the superpixel v_i, and $\mathbb{I}(\cdot)$ is the indicator function. In other words, S_d is equal to the average size of the remaining superpixels, after removing superpixels with sizes larger than S_d. This can be solved easily via an iterative procedure.

After obtaining the desired size, the adaptive threshold T_{ik} for superpixel v_i at iteration k is set to

$$T_{ik} = T_0 + kT_s \exp\left(-\sigma \frac{|v_i|}{S_d}\right) , \tag{2.11}$$

where T_0 is an initial threshold and T_s is the step size. $\sigma > 0$ is the parameter governing the tradeoff between large and small superpixels, so that T_{ik} is higher for smaller superpixels. The algorithm is not sensitive to T_0 and T_s which can be chosen simply to be sufficiently small. However, a larger T_s reduces computation time, hence is more desirable if there is no adverse impact on performance.

After obtaining the adaptive threshold, the edge and color distance between each connected superpixel pair are computed, and the pair is merged if both distances are smaller than the T_{ik} of the smaller superpixel in the pair. As the iteration advances, the threshold becomes larger and more small superpixels are merged since their relative penalty becomes

larger after more iterations.

Within each iteration, we compute a merge graph M, with an edge on each superpixel pair that ought to be merged. This merge graph is complemented by the conflict graph C, which has an edge on each superpixel pair that are incident to each other but should not be merged. We start with the superpixel with the highest degree on M and proceed to iteratively merge all its neighbors without conflicts. If there are conflicts, we choose the one with the highest degree on M among the conflicting superpixels to merge.

Most merging schemes have a clean-up routine for removing small superpixels. For our algorithm, every 5 iterations we run one "small superpixel merging" process, which is almost the same as normal merging, with the only difference being that the color difference from a large superpixel to a small one is only computed within a small vicinity of the latter. This is because the large superpixel might contain very distinct colors because of merging, and the mean color might have differed a lot from the smaller one. However if their colors are similar in the vicinity of the smaller superpixel, then the two should be merged.

We then generate one seed at the center of each merged superpixel, which has the capability of representing a complete picture of the scene with a moderate number of seeds.

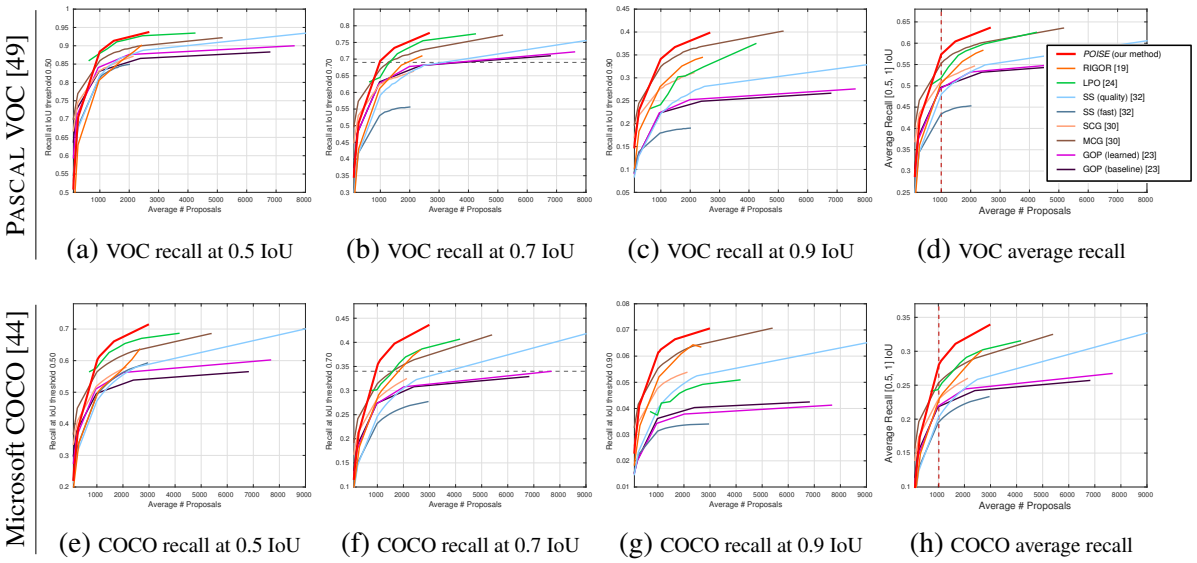
2.6 POISE Pseudo-code

Algorithm 1: POISE proposal generation

Compute boundaries B for image I using [43].
From B generate superpixels $\mathcal{V} = \{v_1, \ldots, v_n\}$ using StructEdges toolbox [43].
For every pair of superpixels v_i, v_j which share a boundary, create an edge (i, j).
$\forall (i,j) \in \mathcal{E}$ compute pairwise potentials $\psi_{i \sim j}$ using [35], and pairwise superpixel distances $d_{ij} = \frac{1}{2} c_{ij} + \frac{1}{2} \psi_{i \sim j}$ where c_{ij} is ℓ_2 distance between average $L^*a^*b^*$ colors between superpixels v_i, v_j.
Compute $n \times n$ geodesic distance $g_{ij} \ : \ 1 \leq i, j \leq n$ by dijkstra on $\mathcal{G}' = \langle \mathcal{V}, \mathcal{E}' \rangle$ with edge values d_{ij}. Edge set $\mathcal{E}'$ is obtained by removing 50% of the weakest edges in $\mathcal{E}$, while ensuring $\mathcal{G}'$ is connected.
Initialize proposals set $P := \{\}$.
for *each graph unary type* **do**
    Initialize graph proposals set $P' := \{\}$.
    Find seed locations $\mathcal{V}_\mathbf{s} \subseteq \mathcal{V}$ using the method in §2.5.
    for $\forall v_\mathbf{s} \in \mathcal{V}_\mathbf{s}$ **do**
        $\mathbf{x}_l := [x_1, \ldots, x_n]$ where $x_i = \begin{cases} 1 & \text{if } v_i \in v_\mathbf{s} \\ 0 & \text{otherwise} \end{cases}$.
        $\forall v_i$ compute unary parameters e_i, f_i, g_i, h_i.
        Set seed $v_\mathbf{s}$ source capacity $\theta^0_\mathbf{s} := \infty$.
        for $\forall \lambda_k \in \lambda_0 < \lambda_1 < \cdots < \lambda_L$ **do**
            /* Get src./sink unary capacities for $\forall v_i$ */
            for $\forall v_i \in \mathcal{V} \setminus v_\mathbf{s}$ **do**
                $\theta^0_i := e_i + \lambda_k f_i$
                $\theta^1_i := g_i + (\lambda_L - \lambda_k) h_i$
                $\phi_i(\mathbf{x}_l) := min_{j \in \mathcal{V} \, : \, x^{(l)}_j = 1} g_{ij}$
                if $\phi_i(\mathbf{x}_l) < \tau$ **then**
                    $\theta^0_i := \theta^0_i + h(\phi_i(\mathbf{x}_l))$
                else
                    $\theta^1_i := \theta^1_i + h(\phi_i(\mathbf{x}_l))$
            /* Compute max-flow/min-cut */
            $\mathbf{x}_l := \arg\min_\mathbf{x} \sum_{i \in \mathcal{V}} (\theta^1_i x_i + \theta^0_i \overline{x}_i) + \sum_{(i,j) \in \mathcal{E}} (\psi_{i \sim j} |x_i - x_j|)$
            $P' := P' \cup \mathbf{x}_l$
    /* Filter proposals P' */
    Separate connected components in P'.
    Discard proposals in P' smaller than 12 pixels.
    If $|P'| > \gamma$ then randomly select γ proposals.
    Discard duplicates in P' - proposals with 0.95 overlap are considered as duplicates.
    $P := P \cup P'$
Discard duplicates in P - proposals with 0.95 overlap are considered as duplicates.

2.7 Experiments

We conduct experiments on the validation sets of PASCAL VOC 2012 and Microsoft COCO [44]. Both have pixel-level annotations for certain object classes. There are $1,449$ images in VOC 2012, with $3,427$ ground-truth objects in 20 categories. COCO has $40,137$ images and $288,397$ ground-truth objects, with 80 categories that are currently available. Our algo-



(a) VOC recall at 0.5 IoU (b) VOC recall at 0.7 IoU (c) VOC recall at 0.9 IoU (d) VOC average recall

(e) COCO recall at 0.5 IoU (f) COCO recall at 0.7 IoU (g) COCO recall at 0.9 IoU (h) COCO average recall

Figure 2.7: These graphs compare different object proposal methods based on recall against number of proposals at three IoU thresholds. For each segment ground-truth we select the proposal with the highest segmentation IoU. We use this to compute recall, which is the fraction of ground-truths having a corresponding proposal with an IoU score higher than the IoU threshold. [45] gives a similar comparison between methods for bounding box IoU. Note, the y-scale of each graph is different.

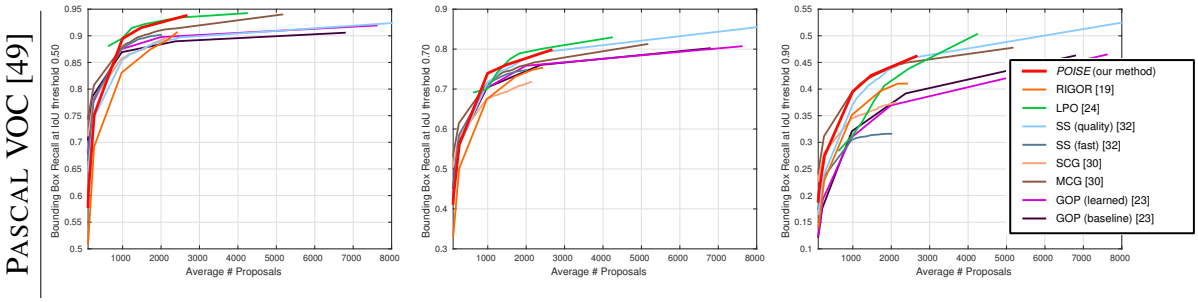
rithm is implemented in MATLAB with many crucial functions written in C++. We utilize StructEdges [43] for boundary detection and sticky superpixels [43] as nodes in the graph. Pairwise terms are computed from trained boosted regressors from RIGOR [35].

We report a number of metrics that have been widely used in previous evaluations. Suppose we want to evaluate a segment pool $\mathbf{S} = \{S_1, \ldots, S_n\}$ against m ground-truth segments. First of all, each segment proposal $S_i \in \mathbf{S}$ is evaluated w.r.t. each ground-truth using the IoU overlap score

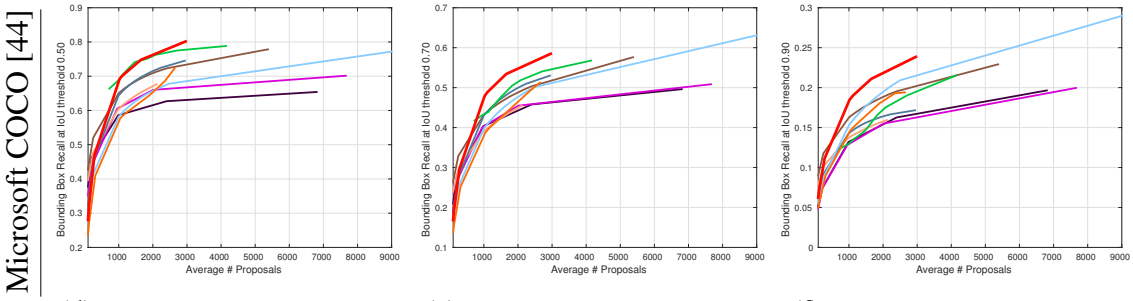
$$\mathrm{IoU}(S_i, GT_j) = \frac{|S_i \cap GT_j|}{|S_i \cup GT_j|}\,, \tag{2.12}$$

The best object overlap within the pool $\mathbf{S}$ is computed as

$$\mathrm{IoU}(\mathbf{S}, GT_j) = \max_i \mathrm{IoU}(S_i, GT_j)$$



(a) VOC BB recall at 0.5 IoU (b) VOC BB recall at 0.7 IoU (c) VOC BB recall at 0.9 IoU



(d) COCO BB recall at 0.5 IoU (e) COCO BB recall at 0.7 IoU (f) COCO BB recall at 0.9 IoU

Figure 2.8: These graphs compare different object proposal methods based on bounding-box recall against number of proposals at three IoU thresholds. See Fig. 2.7 for segmentation recall against number of proposals. A similar bounding-box recall analysis is present in [45] for PASCAL VOC 2007 test set and Microsoft COCO 2014 validation set. Note, the y-scale of each graph is different.

We report the average best overlap (ABO), which is $\mathrm{IoU}(\mathbf{S}, GT_j)$ averaged over all of the ground-truth objects in the dataset, as well as plotting recall under different IoU levels against the number of segments in the pool $|\mathbf{S}|$. In addition, we follow [45] in reporting the average recall under all IoU levels in $[0.5, 1]$. It is claimed that such an average recall measure correlates the best with downstream results on object detection [45]. Finally, we report the mean best covering over all images in the dataset:

$$\mathrm{Cov}(\mathbf{S}, \mathbf{GT^I}) = \frac{\sum_j |GT_j| \mathrm{IoU}(\mathbf{S}, GT_j)}{\sum_j |GT_j|}$$

where $\mathbf{GT^I}$ denotes all ground-truth objects in the same image. Covering measures the capability to extract larger segments and explain the scene as a whole.

We compare against recent methods SS (Selective Search) [30], SCG and MCG [32],

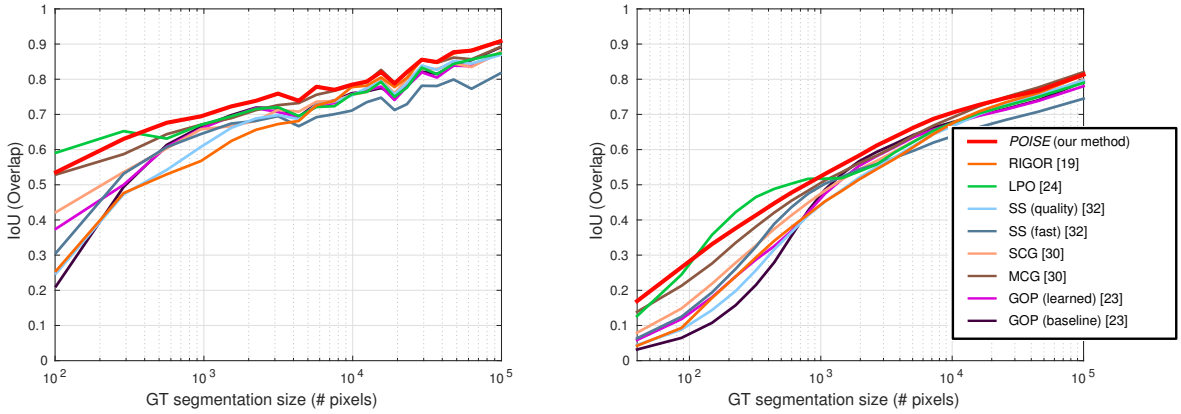
Table 2.1: Detailed PASCAL VOC and COCO results of different algorithms. We consider two scenarios: the first is to generate $\sim 69.0\%$ recall (or $\sim 34.0\%$ recall in case of COCO) at an IoU threshold of 0.70; the second is the limit performance by allowing all algorithms to generate the maximal amount of proposals. Our method, POISE is able to obtain similar performance with much fewer proposals than the competitors.

	PASCAL VOC [49]					Microsoft COCO [44]				
Method	Recall at 0.70 IoU	Avg. # Proposals	ABO	Cov	Average Recall	Recall at 0.70 IoU	Avg. # Proposals	ABO	Cov	Average Recall
	$\sim 69.0\%$ **recall at IoU threshold 0.70**					$\sim 34.0\%$ **recall at IoU threshold 0.70**				
GOP (learned) [38]	0.678	1,992	0.748	0.814	0.532	0.340	7,659	0.544	0.740	0.267
RIGOR [35]	0.682	1,715	0.752	0.840	0.557	0.345	1,894	0.518	0.744	0.270
SS (quality) [30]	0.681	2,482	0.757	0.828	0.549	0.323	2,496	0.532	0.744	0.259
LPO [29]	0.682	1,237	0.759	0.822	0.544	0.331	1,470	**0.567**	0.734	0.267
MCG [32]	0.692	1,291	0.768	0.835	0.570	0.340	1,587	0.551	**0.750**	0.272
POISE	0.694	**995**	**0.771**	**0.843**	**0.574**	0.355	**1,018**	0.550	0.744	**0.280**
	Limit performance at IoU threshold 0.70					**Limit performance at IoU threshold 0.70**				
GOP (learned) [38]	0.722	7,609	0.769	0.829	0.566	0.340	7,659	0.544	0.740	0.267
RIGOR [35]	0.709	**2,411**	0.777	0.844	0.583	0.384	**2,642**	0.568	0.753	0.300
SS (quality) [30]	0.772	10,641	0.801	0.840	0.618	**0.456**	15,950	**0.631**	0.770	**0.355**
LPO [29]	0.776	4,233	0.805	0.859	0.626	0.406	4,146	0.602	0.769	0.315
MCG [32]	0.772	5,157	0.808	0.850	0.635	0.415	5,376	0.600	0.770	0.325
POISE	**0.778**	2,650	**0.811**	**0.864**	**0.636**	0.435	2,959	0.617	**0.774**	0.339

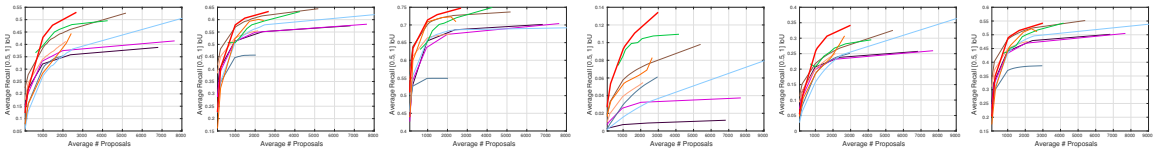
GOP [38], RIGOR [35] as well as the very recent LPO approach [29].

Table 2.1 shows detailed performance of different algorithms under two settings: one where all algorithms generate about 69.0% recall at an IoU threshold of 0.70; and the second where algorithms are allowed to generate maximal number of proposals. One can see that our method generates much fewer proposals in any of the two scenarios while having comparable performance to the best competitors. Fig. 2.7a-2.7d shows the plots of segment recall at different overlap thresholds on the VOC dataset. Likewise, Fig. 2.7e-2.7h shows the results on the COCO dataset. We use linear instead of log scale [45, 32] to highlight that our method needs far fewer proposals to reach high recall. It can be seen that *our method consistently outperforms the competitors when the number of proposals is more than* 700, which is the range of settings most likely to be chosen users of proposal algorithms. POISE is superior to most other superpixel aggregation and edge-based approaches because it seeks solutions from a global energy function which solves the middle child problem.

Fig. 2.9 shows the IoU score broken down in terms of the size of the ground-truth



(a) PASCAL VOC 2012 validation (b) Microsoft COCO 2014 validation



(c) VOC average recall small/medium/large objects (d) COCO average recall small/medium/large objects

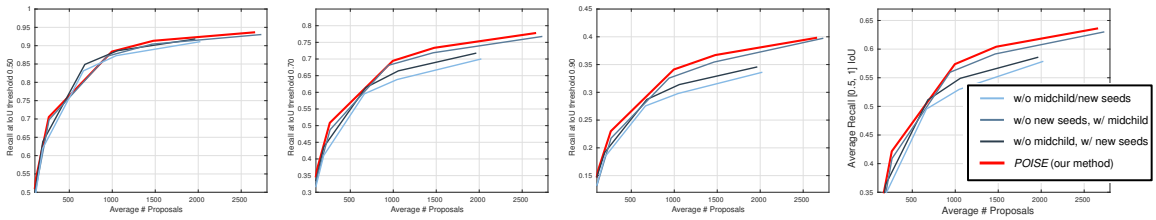
Figure 2.9: IoU comparison of various methods at different pixel sizes for PASCAL VOC 2012 validation and Microsoft COCO 2014 validation segmentation ground-truths, at $\sim 1,000$ # proposals.

segment. It can be seen that our method significantly outperforms all other approaches in objects with the sizes from 400 to $4,000$ pixels. This shows the effectiveness of our solution to the middle child problem, as well as the benefit of better seed placement. The only regime in which we are slightly worse than RIGOR is when the segment size grows to more than $60,000$ pixels, which is approaching the size of the entire image for a typical PASCAL VOC image. Even at that ground-truth size we still outperform all of the other competitors.

2.7.1 Ablation Study

This chapter has two contributions: a solution to the middle child problem; and a superpixel seeds generation method. In this section we will describe the results of an ablation study to identify the quantitative contribution of each of these two components. We compare four different variants of the algorithm: (1) "**w/o midchild/new seeds**" where neither the

middle child solution in §2.3 or the new seeds in §2.5 are used; (2) "**w/o new seeds, w/ midchild**" where we use the geodesics middle child solution in §2.3, but not §2.5; (3) "**w/o midchild, w/ new seeds**" where we use the new seeds in §2.5, but not §2.3; and (4) the **POISE** method corresponding to the full algorithm in §2.3 and §2.5.



(a) VOC recall at 0.5 IoU (b) VOC recall at 0.7 IoU (c) VOC recall at 0.9 IoU (d) VOC average recall

Figure 2.10: These graphs show the ablation study for our algorithm. We compare three different schemes, as given in §2.7.1. This analysis is similar to Fig. 2.7, where we show recalls of each scheme against different number of proposals it generated. We vary the number of seeds to generate results at different number of proposals. We also show the average recall between [0.5, 1] IoU for all schemes in d.

We generate results for all these variants individually over the complete validation set for PASCAL VOC 2012. The average recall results are plotted in Fig. 2.10. The general trend observable from these results is that the improved seeds (w/o midchild, w/ new seeds) help to move the graph left by reducing the number of proposals to reach the same recall. This is the result of requiring fewer number of seeds to localize most objects in the scene. On the other hand, the middle child solution (w/o new seeds, w/ midchild) moves the graph upward, indicating that adjusting the unaries by geodesics helps to obtain more accurate segmentations. Combining both improvements gives POISE the ability to increase recall while using fewer proposals.

2.8 Conclusion

In this chapter we identified and solved the *middle child problem*— namely how to use parametric min-cuts to generate medium-sized segments for object proposals. We demonstrated that the problem arises from the intrinsic structure of the standard energy landscape

and cannot be solved through parameter tuning. Our solution is an adaptive energy function which biases the min-cut solution in a sequence of proposals so that the next segment is close to the previous one from the standpoint of geodesic distance. In addition, we introduced a novel method for generating proposal seeds which is more effective than previous methods for small numbers of seeds. The resulting method, known as POISE (for "Proposals for Objects from Improved Seeds and Energies"), is demonstrated to outperform all competing methods in generating high-quality segments with a small proposal pool on the PASCAL VOC and Microsoft COCO datasets.

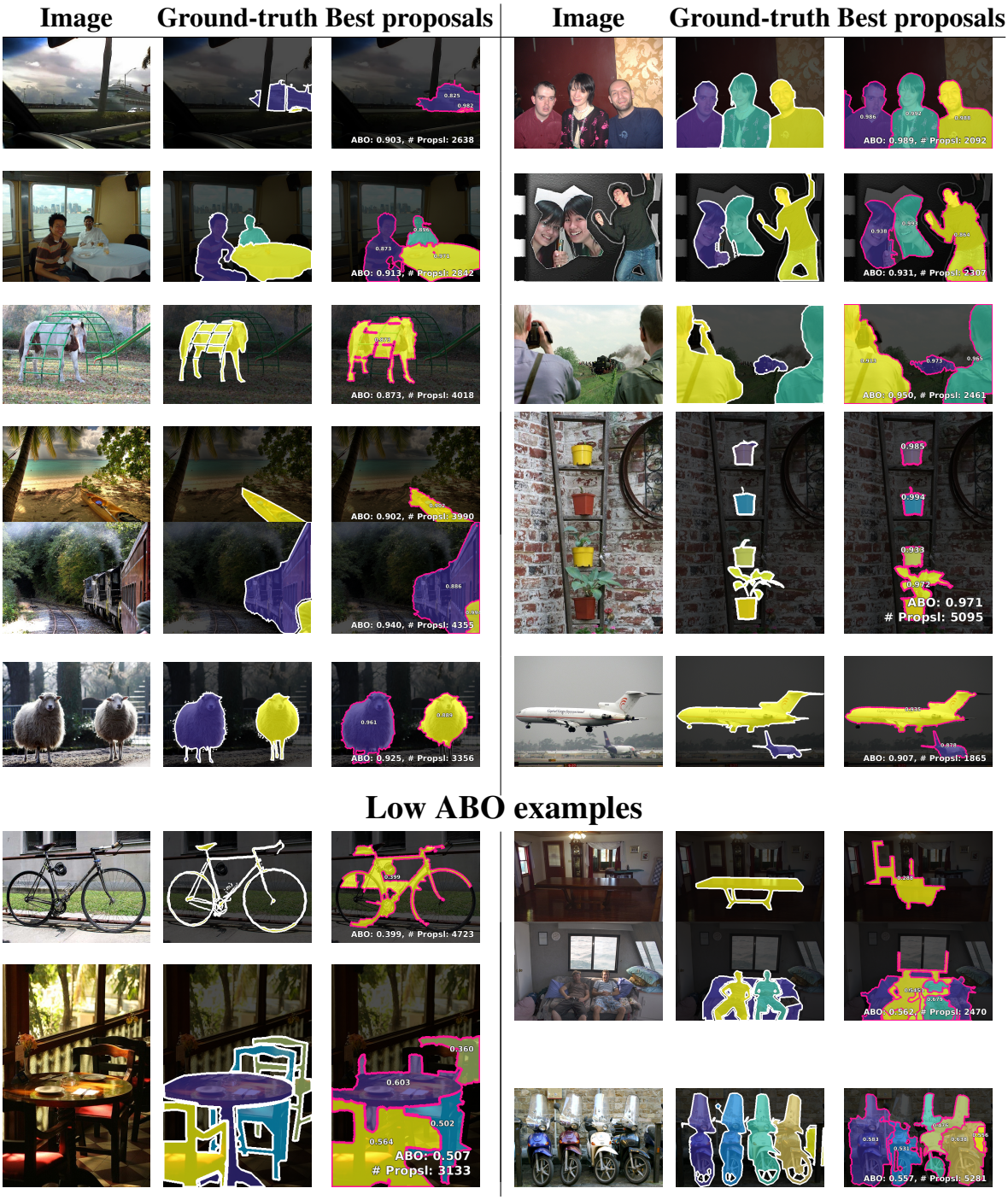


Figure 2.11: Qualitative evaluation on PASCAL VOC 2012 [49] validation set. For each image we display the ground-truth and the proposals which obtain the best segmentation IoU score. The IoU score is displayed on top of each best proposal. Each image also displays the average best overlap (ABO), and the total number of proposals generated for that image. The results here are generated when POISE is set to propose 1000 objects on average.

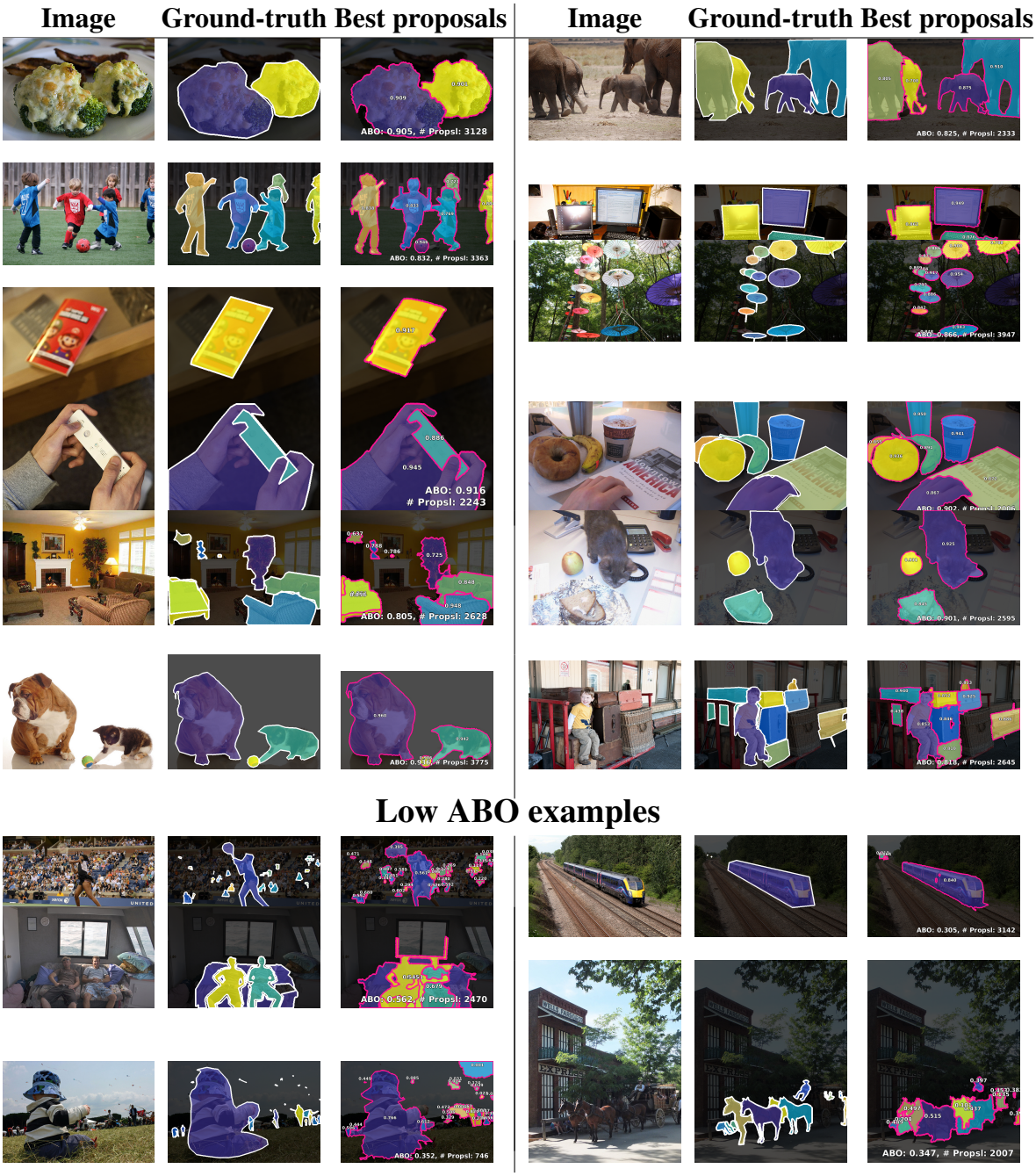


Figure 2.12: Qualitative evaluation on Microsoft COCO 2014 [44] validation set. See caption in Fig. 2.11 for more details.

Figure 2.13: Qualitative comparisons against four different methods [35, 32, 29, 30]. The images are from PASCAL VOC 2012 [49] validation set and Microsoft COCO 2014 [44] validation set. These proposals were generated when each method was proposing 1000 objects on average.

Chapter 3

Fully Convolutional Video Object Detection

In this chapter I would introduce a pipeline to generate space-time object proposals for an unconstrained video stream. As an application, I would demonstrate their use in object detection. More formally, this chapter would offer a solution to the following problem:

> *Given a video stream, generate space-time bounding boxes which have a high chance of localizing objects in the scene. Furthermore, classify each proposed object as one of the predefined set of categories, and track the object on-line over the length of the video.*

In short, the problem asks for localizing object in a video, and demonstrating their use for detection and tracking in a stream of video. Note that since the problem supplies a continuous stream of frames, any long term batch processing would be infeasible. An effective method might observe a small number of frames for inference, but in this setting it would not be possible for a method to look at hundreds of frames before giving an output. Moreover batch methods which require looking at the complete video before giving the final output [69, 70, 71, 72] would be inappropriate for this problem. Also an algorithm would need to link results (detections) from previous steps to the current inference. Typical

examples for this technique generate proposals every frame, and attempt to link them to spatio-temporal object proposals resident in memory [6, 73].

Image object detection is a problem which has received wide interest in recent years. This has resulted in many non-deep [47, 74] and deep models [10, 5, 23] aiming to improve accuracy and speed. This chapter would explore whether these ideas can be extended to locate objects in a fixed set of frames by using simple 3D convolutional extensions to current detection pipelines. More concretely, this chapter would propose a supervised pipeline for *video object detection* — the problem of locating, classifying, and tracking all objects throughout its lifetime in a streaming video. The object categories that the algorithm would be trained and tested on would be limited by the annotations in the dataset (see §3.2). First, a fully convolutional architecture would be described in §3.3 to localize objects in time. The training would proceed by minimizing a loss function which would encourage the method to locate and consistently classify objects in time given supervision over fixed set of categories (see §3.4). Once trained, this model would be used for inference on a fixed number of frames (details in 3.6). This pipeline would be evaluated on its ability to recall and classify objects in a fixed time window (§3.7.2) and to track these objects over the complete length of the video (§3.8.7).

3.1 Related Works

3.1.1 Video Object Proposals

Following the lead of image object proposal algorithms, recently many techniques have been proposed to generate proposals in video. Unlike our proposed convnet architecture, most of the techniques are unsupervised, and start by inferring object localizations in each frame before stitching them temporally. Our proposed approach infers spatio-temporal object locations over a set of frames in a single forward pass. Proposals from different time windows are linked by simple bipartite matching.

Rather than directly inferring spatio-temporal proposals, most methods generate video proposals by linking image proposals across video frames. Lee *et al*. [69] begins by generating object proposals every frame using [36], and scoring them by appearance and motion cues. High scoring proposals are spectrally clustered to produce sets of possible foreground objects. Shape and color models are created for each foreground set, and used to generate foreground likelihood maps. These maps are used as unary potentials for segmenting objects in a space-time MRF.

Banica *et al*. [70] take a similar approach by temporally linking CPMC proposals [27]. Each hypothesis is ranked using saliency, flow, and image proposal matching scores. These hypothesis are refined by running binary graph-cut on every frame for each hypothesis. Li *et al*. [6] offer an on-line unsupervised method to generate video segment proposals. Like [70], it first generates a pool of CPMC proposals in the current frame, and computes appearance features for each segment. Using the appearance model, a proposal is associated to an object track using a least squares tracking formulation. Finally it performs composite statistical inference for refining segmentations of longer tracks. Wu *et al*. [73] builds on this technique by adding ability to track objects undergoing complete occlusion.

Similar to [6, 73], Fragkiadaki *et al*. [71] also uses optical flow to delineate object boundaries for generating better image proposals [38]. This method was the first to employ a convnet to help generate video proposals — in their pipeline a convnet is used to filter static unimportant objects. The remaining moving proposals are linked temporally by computing pairwise trajectory affinities [75], and the final confidence rank for video proposals is an aggregation of convnet scores.

Sharir and Tuytelaar [72] is similar to our proposed approach in generating proposals for fixed size time windows. Though, unlike our approach, it generates spatio-temporal proposals in each time window by linking together image object proposals [36]. These proposals are then linked across frames by finding a minimal distance path based on color histogram intersection. The hypotheses generated are then used as high order potentials

in a CRF, where inference helps produce better segmentations. Proposals across temporal windows are combined by finding high overlaps on adjacent frames. All video proposals are ranked based on motion and objectness scores given by [36]. In contrast, our model generates proposals by asking the network to predict an objectness score. Since the objectness score is trained discriminatively, it produces a small number of proposals per time window (20-50), which helps avoid simple association mistakes.

Xiao and Lee *et al*. [76] proceeds by using high scoring box proposals [57] and iteratively refining models to detect harder instances of the object. These image proposals are clustered with other temporally non-adjacent proposals, and clusters are selected based on distinct appearances. Detectors are trained for each cluster group to iteratively add proposals from missing frames in a cluster. Finally these box proposals are converted to segmentations using a grab-cut scheme, where features are generated through a hypercolumn representation [77]. Like most other methods [69, 70, 71, 72], this algorithm also requires multiple passes over all frames to generate video proposals.

Another set of methods aim to generate spatio-temporal proposals by iteratively clustering pixels into supervoxels. Van den Bergh *et al*. [78] introduces one such method, which extends the SEED superpixel algorithm [79] to generate supervoxels in an on-line setting. It is an energy minimization approach where supervoxels are encouraged to have homogeneous color. By introducing noise in their optimization they can generate multiple partitionings — these multiple results tend to agree at true object boundaries, and hence are used for computing an objectness score. Space-time proposals are made by tracking windows which align well with supervoxel boundaries. Oneata *et al*. [80] generate supervoxels by hierarchically merging SLIC superpixels [67]. It generates proposals by starting with a seed supervoxel, and then iteratively adding supervoxels probabilistically (similar to Randomized Prim's algorithm [81]), where the likelihood is learned by a logistic discriminant classifier based on color, flow, size, and compactness features. Unlike supervoxel based methods, our proposed method does not need to generate a mid-level representations

in order to generate object localizations.

3.1.2 Video Detection

There has been growing interest in combined object detection and tracking pipelines since the introduction of the ImageNet video detection dataset [12]. Both the video detection benchmark/challenge and the availability of a large dataset has encouraged research in supervised learning for object recognition in videos. Yet, most of the methods introduced are an amalgam of algorithms, which typically associate and smooth frame level object detector responses. In contrast the proposed pipeline generates spatio-temporal object detections in a single convnet over fixed time windows, which are linked together across different time windows using simple bipartite matching.

Kang *et al*. [82] aims to generate better image object detections by propagating evidence from other frames in the video. Initially they generate proposals every frame which are filtered by RCNN [10] and classified by GoogLeNet [51]. To reduce false negatives, each high confidence detection is tracked forward and backward in time by a fully convolutional network [83]. Temporal 1D convolution over detection and tracking scores is trained to predict recall above 0.5 with GT. Compared to a still image detector, the performance of this method rises from 45.3% to 47.5% mean AP by using only $1/38$ proposals. Kang *et al*. [84] improve this method by combining results from multiple image detectors, suppressing low confidence classes, and propagating detections using optical flow. This was the winning entry in ImageNet Vid challenge in 2015 (compared to methods not using additional data), with a mean AP of 73.8% on the validation set [12]. Even though both methods generate video object detections, the ImageNet video benchmark in 2015 was limited to frame level detection performance. In 2016 a video detection metric was introduced. The challenge was won by a method employing an on-line multi-object tracking framework using frame-level detections from [84]. One key element missing from these methods is the ability to take temporal information into account for classification (a large number of objects are

mis-classified as background by in image object detectors [23]). Our proposed pipeline would naturally incorporate information from multiple frames before inferring a class for the object.

Kang *et al*. [85] is more similar to our approach in using space-time anchors to generate object localizations. It first generates object proposals using RPN [5] over a reference frame, and then it pools features from the same spatial location over multiple frames. It uses features these multi-frame pooled features to train a regressor to find the movement of the object lying in the receptive field of this "spatial anchor." It pools features from boxes resulting from the movement prediction in an encoder-decoder LSTM to classify the object into one of the 30 ImageNet VID categories. The major difference this work has to our technique is in how their prior anchor locations are used to predict locations for an object. Our 3D convolutional network is discriminatively trained to infer whether a certain space-time anchor lies over an object, and if so, what category does the object belong to. Unlike [85] which has two separate networks for predicting localizations and classes, our pipeline is a single network which produces detections by exploiting space-time information in a short time window.

Han *et al*. [86] introduces a simple method for improving non-maximal suppression of object detectors using information available in video. After high overlap detections are linked in adjacent frames, the method prunes image detection results by selecting temporally linked bounding boxes which give the highest aggregate confidence scores. Since our proposed method would be generating space-time proposals, NMS can naturally run spatio-temporally.

Recently, Drayer and Brox [87] introduced a video object detection and segmentation method. It first generates a set of proposals [57] classified by RCNN [10], which are then linked by dynamic programming on a graph where links are formed by similarity between proposal boxes. Each linked set of proposals is spatio-temporally segmented with the help of appearance and motion cues extracted from bounding boxes. Like other methods, this

also locates objects per frame, requiring explicitly smoothing inferences across time — whereas in our proposed model, temporal smoothing occurs as the model is trained to minimize the localization loss over multiple frames simultaneously.

Feichtenhofer *et al*. [88] proposes a unified approach to detect and track objects. The method extends R-FCN [89][1] by adding a tracking loss which aims to regress object co-ordinates across frames. It makes use of the idea of learnable correlation filters [90, 91] between a set of frames to find how each object moves over time. Unlike our approach, all frames are passed through a ResNet-101 [92] to obtain a per frame convolutional feature map. On the other hand, a 3D convolutional network which takes multiple frames as input could build flow-like or correlation filters as part of the bottom-up process.

Liu and Zhu [93] propose a CNN interlaced with LSTM for integrating temporal information for video object detection. Multiple convolutional layers are tasked with generating feature maps which are fused with other feature maps transformed by a recurrent network from previous frames. Like our technique, their model is also based on the SSD detection architecture [14] - albeit they make use of Mobilenet-SSD [94] to make the technique applicable to computation constrained settings.

The winning entry from 2017 ImageNet VID challenge [95] makes use of optical flow to aggregate features from multiple frames. They make use of video recognition feature aggregation ideas in [96] to generate deep features from adjacent sampled frames, and propagating them using optical flow. Once the features are propagated via flow, features from multiple frames are combined according to the cosine similarity to features in the reference frame. They show that feature aggregation by flow guidance gives a boost of 2-4 mAP points on ImageNet VID validation. Similar to [96] is the MemNet [97] approach which produces detections using current frame's feature representation, which is aggregated with the warped feature representations accumulated from previous frames. The warping, like

[1]Region-based Fully Convolutional Networks [89] pool responses from position-sensitive feature maps for detecting objects. Since it is a data-driven process, the pooling is designed in a way to encourage certain feature maps to respond to particular locations relative to an object (top-left, right, *etc*.). This results in relatively translation-equivariant object features, while also maintaining some position sensitive information.

in [96], is performed using optical flow. Unlike these methods, which require explicit computation of flow, our 3D convolutional network can learn to devote filters to discriminate between different object motions and aggregate features across frames.

More recently the idea of deformable convolution [98] has been employed for video object detection [99]. Bertasius *et al*. uses deformable convolutions to condition the receptive field on adjacent frames for improving the feature representation for detection on a reference frame. They demonstrate a performance gain of around 2.7 mAP when compared to a single frame baseline. Their backbone convolutional network is based on ResNet-101 [92] and R-FCN [89]. The model is trained to deform feature maps from multiple adjacent frames and then combine them into a single feature map by weighing them according to their cosine distance to the reference frame's feature map. This has two advantages over our method: (1) during inference time, it can arbitrarily use more or less frames than what the network was trained for; also (2) deformable convolution allows the method to pool features conceptually in the direction of the motion of the object. But unlike our approach, based on 3D convolutions, temporal information in this method is collapsed by a simple feature aggregation technique. This makes the model incapabale of constructing features which lie at different temporal scales.

3.1.3 Action and Pose Detection

Even though action and pose detection aim to more fine-grained classification (inference on the pose or motion of humans), one can design solutions to these problems with object detection like architectures. I will discuss two methods here, one for each problem, whose architecture designs are quite similar to ours.

Hou *et al*. [100] proposes a 3D convolutional network for the problem of action detection. The goal of this method to localize humans performing different actions, while keeping track of them over the length of the video. They too divide the video into T size clips, where each clip generates action proposals - and these are stitched together to create

proposals over the length of the video. This approach first collapses all the temporal dimensions to first localize actions in space. Then it predicts the movement in time by pooling features from the shallower part of the network which contains all frame level information. There are 3 major differences to our approach: (1) our anchors predict localizations for all the frames in the clip in the first stage, whereas [100] first predicts spatial locations before predicting temporal movement; (2) [100] uses ROI pooling in space-time, whereas our architecture avoids pooling by using an SSD [14] style model; and (3) we predict both localizations and classifications from the same prediction layer, and [100] performs classification once it has features from all the clips of the proposal.

Girdhar *et al*. [101] also proposes a 3D convolutional network where the goal is track keypoints for each human over the video. It is more similar to our approach than [100] since it makes localization predictions by an anchor box which spans spatio-temporally. But, like [100], they also use a pooling mechanism before making final predictions for keypoint locations and bounding boxes. Another difference in our approach is that we collapse all temporal dimensions before the prediction layer, whereas the base model [101] outputs a feature map of the same temporal size as the input. They also only demonstrate results with a network trained on 3 frame inputs, whereas we show results with networks trained with different temporal input sizes.

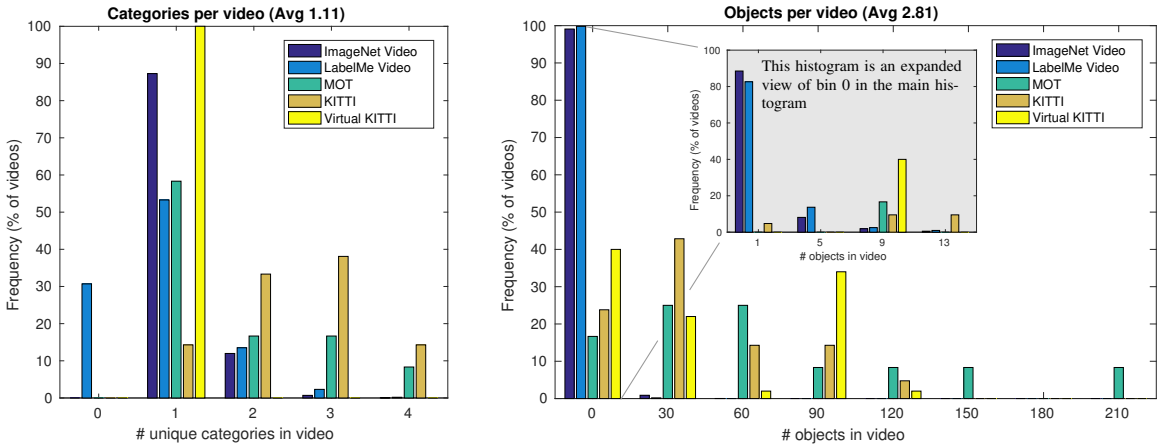
3.1.4 Motion-based Features

Part of this proposed work aims to demonstrate that supervised feature learning can be useful for object recognition in video. The aim of the proposed network would be to learn features for producing high quality, temporally consistent localizations, and classifications. The pipeline would employ 3D convolutional filters in order to learn more generic video representations, like moving occlusion boundaries and T-junctions. This would be analogous to first two layers of a network learning edges and corners when trained on images [102]. Several video recognition problems have successfully used spatio-temporal

feature learning. Action recognition has been the key application of deep networks tailored for videos. Some of these methods have shown competitive performance using convolutional RBM architectures [103, 104] or ISA [105] which learn spatio-temporal features from un-labeled data. Other action recognition employ 3D convolutions [106, 107, 108], and specialized CNNs built on optical flow input [109]. There have been several other problems in video where feature learning has been experimented with, including video classification [110, 111], and optical flow prediction [112, 113].

C3D [110] is noteworthy since it informs some of the choices we make for constructing 3D convolutional architectures. It did a detailed study using a VGG derivative with 3D convolutions, showing excellent performance on action recognition and other video tasks. This study was one of the first to show the efficacy of using $3 \times 3 \times 3$ convolutions for extracting temporal information. Tran *et al*. [114] builds on C3D by introducing 3D convolution variants of ResNet-18 and ResNet-34 [92]. Taking lessons from these works, we construct different 3D convolution variants for ResNet-34 for the task of video object detection. Recently, Carreira and Zisserman [115] did a detailed study on what different architectures work best for action recognition. They showed that when using an Inception [51] model, having a two stream architecture (on flow and RGB), both with 3D convolutions, works best for action recognition. We make use of their technique for inflating 2D filters for initialize 3D convolutions in several of our experiments.

Amongst the recent non-learning based works on video features, Park *et al*. [116] requires mention because the problems it discusses are relevant for object localization and detection. More concretely, [116] discusses conditions that can arise in video which affect the generation of motion features for object detection. It generates object features after factoring out camera motion and global object motion, which they apply to pedestrian detection using a HOG-SVM detector [47].



(a) # of object categories per video

(b) # of objects annotated per video

Figure 3.1: Each bar graph shows the fraction of videos in each dataset which have a set number of categories, or the number of annotated objects lie in a particular range. [6, 73]

3.2 Training Data

The video proposal generation and detection pipeline in this chapter would be built using convnet. The features for this convnet would be learned using supervision — hence it is important to have a discussion about the available datasets.

Table 3.1 shows five datasets where object instance level annotation is given for all frames. The table shows important metrics like the total number of objects and the average track length for an object. The metric for 'average number of objects per video' helps us gauge what datasets would be useful for testing algorithms in cluttered scenes. MOT [117, 118] clearly takes the lead here, but is hindered by the low number of object categories. A more detailed analysis is given in Fig. 3.1b, which additionally shows that Virtual KITTI [119] also has a variety of cluttered scenes. Moreover, datasets which have wider variety of object categories per video can be potentially used to learn co-occurrence statistics, which are shown to be useful for object detection. These statistics can be viewed in detail in Fig. 3.1a, which shows that KITTI has the most variability in the number of categories per video.

All datasets, except LabelMe Video [120], were annotated with a fixed set of categories.

Table 3.1: Different video datasets where object instance level annotation is available.

Dataset	# Videos	Avg. # Frames per Video	# Annotations	# Objects	Avg. Track Length per Object	Avg. # Objects per Video	Avg. # Objects per Frame	# Categories	# Deformable Object Categories	Avg. # Categories per Video
Imagenet Vid [12]	4417	294.0	2 005 418	9220	217.5	2.09	1.54	30	23	1.13
Training	3862	290.6	1 731 913	7911	218.9	2.05	1.54	30	23	1.13
Validation	555	317.3	273 505	1309	208.9	2.36	1.55	30	23	1.12
LabelMe Video [120]	518	262.0	182 382	977	186.7	1.89	1.34	15	6	0.88
MOT [117, 118][2]	12	642.9	159 784	883	181.0	73.58	20.71	4	1	1.75
KITTI [121]	21	381.3	44 834	867	51.7	41.29	5.60	7	1	2.52
Virtual KITTI [119]	50	425.2	67 363	2152	31.3	43.04	3.17	1	0	1.00
All	5018	293.2	2 459 766	14 098	174.5	2.81	1.67	34	25	1.11

Table 3.2: Number of objects (tracks) for each category across different video datasets. The $\times$ marker indicates that it has been verified that this category of objects exists in the dataset, but is not annotated.

	Relatively Non-Deformable Objects									Deformable Objects							
Dataset	Airplane	Bicycle	Bus	Car	Motorcycle	Stroller	Train	Truck	Watercraft	Antelope	Bear	Bird	Cattle	Dog	Domestic Cat	Elephant	Fish
Imagenet Vid [12]	606	491	165	1475	309	0	203	0	273	350	189	665	296	645	228	290	0
Training	422	403	140	1246	278	0	176	0	237	313	166	596	244	556	194	250	0
Validation	184	88	25	229	31	0	27	0	36	37	23	69	52	89	34	40	0
LabelMe Video [120]	4	25	13	385	45	11	3	6	31	0	0	4	0	12	0	0	47
MOT [117, 118]??	0	19	$\times$	32	1	$\times$	$\times$	0	0	0	0	0	0	0	0	0	0
KITTI [121]	0	$\times$[3]	5	636	2	1	12	10	0	0	0	0	0	0	0	0	0
Virtual KITTI [119]	0	0	0	2152	0	0	0	0	0	0	0	0	0	0	0	0	0
All	610	535	187	4656	358	12	216	33	304	350	189	669	296	657	229	291	47

	Deformable Objects																
Dataset	Fox	Giant Panda	Hamster	Horse	Lion	Lizard	Monkey	Person	Rabbit	Red Panda	Sheep	Snake	Squirrel	Tiger	Turtle	Whale	Zebra
Imagenet Vid [12]	162	189	127	233	143	114	344	$\times$	125	195	188	108	181	78	152	312	384
Training	143	168	111	213	134	102	272	$\times$	112	187	179	96	152	67	135	264	355
Validation	19	21	16	20	9	12	72	$\times$	13	8	9	12	29	11	17	48	29
LabelMe Video [120]	0	0	0	0	0	0	0	389	0	0	0	0	1	0	1	0	0
MOT [117, 118]??	0	0	0	0	0	0	0	831	0	0	0	0	0	0	0	0	0
KITTI [121]	0	0	0	0	0	0	0	201	0	0	0	0	0	0	0	0	0
Virtual KITTI [119]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
All	162	189	127	233	143	114	344	1421	125	195	189	108	182	78	153	312	384

For LabelMe Video [120] it was left to the discretion of the annotator to mark whichever objects, using whatever label names. It is possible to combine all the datasets to train a model, but that would require significant effort in normalizing the categories across different datasets. For instance KITTI [121] separates trucks from cars and buses, but ImageNet VID [12] does not.

For our study, it is important to have a large variety of categories with reliable temporally dense annotations. For this reason we choose ImageNet Vid [12] for all our experiments. The final set of categories, and their counts per dataset are given in Table 3.2. Since motion is an important aspect of this chapter, both Table 3.1 and Table 3.2 make a distinction between deformable and non-deformable object categories (note that all categories can move nevertheless both due to object and camera motion). Note that we are using the 2015 release for ImageNet VID, both for training and validation, in order to keep our results comparable to other works on video object detection.

3.3 Architecture

The proposed architecture produces video object detection predictions using a convolutional network. The input to the network is T frames of the video, where all frames are resized to fixed resolution. The ouput is a fixed number of video object detections, where each detection is specified by $4T$ numbers giving the extent of the object's bounding box over T frames. In addition, each prediction is accompanied with a likelihood of it belonging to one of M categories or the background. The feed forward stage is followed by non-maximum suppression to produce the final set detections.

The same model can be easily adapted to produce object proposals over T frames, rather than predicting a category label. This would be done by replacing the classification layer with a node that produces an objectness score, which would indicate the likelihood of the proposal belonging to an actual object.

The neural network architecture, depicted in Fig. 3.2, is composed of a base network which produces a feature map of size $512 \times 1 \times 38 \times 38$ (where 38×38 specifies the spatial resolution, and a temporal resolution of 1). The base network is constructed in a way to collapse all temporal information to a single size before it reaches the first prediction layer. We build our architecture this way because (1) empirically we see that any late fusion, i.e.

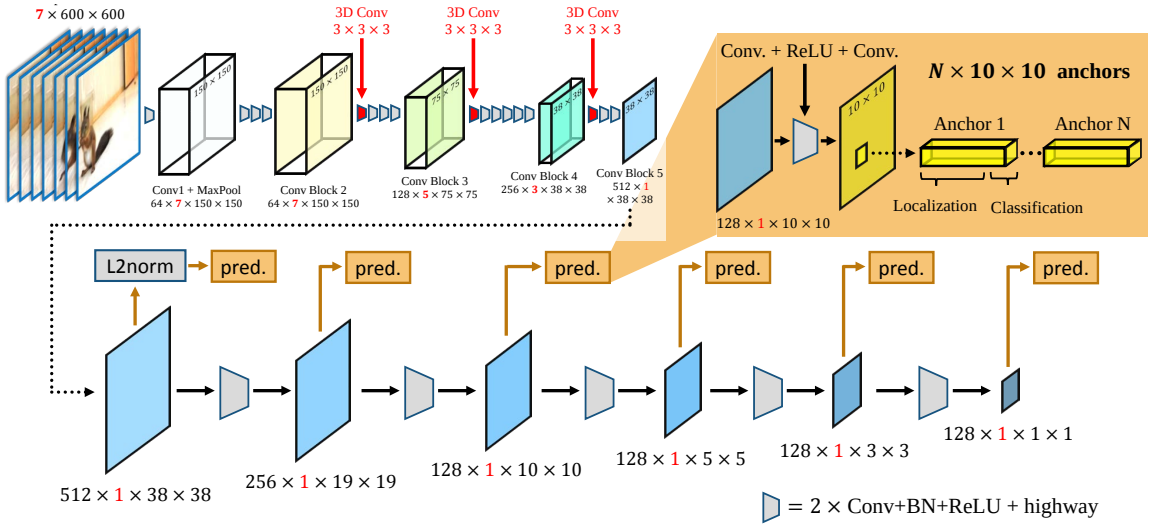


Figure 3.2: This schematic shows the video detection architecture based on a ResNet-34 [92] backbone model. The pipeline shown finds detections at different locations, scales, and aspect ratios in 7 frames of the input video, but it can consume different number of frames by changing the number of 3D convolutions in the backbone architecture. Every location in the displayed feature map produces a fixed number of detection predictions. The idea of predicting bounding boxes at locations on multiple feature maps is inspired by Single Shot MultiBox Detector [14].

maintaining all temporal information till the prediction does not increase the accuracy; and (2) collapsing temporal information in the base network makes it feasible to train multi-frame networks without drastically reducing the batch size. This feature map is passed through a 2D convolution and ReLU non-linearity for producing a $256 \times 1 \times 38 \times 38$ feature maps. Each of the 1444 locations on this feature map produces N_1 video proposal predictions in its receptive field. In the proposed architecture $N_1 = 4$, where each of these predictions aims to produce a moving bounding box close to a preset *space-time anchor*. The spatial size of each space-time anchor is proportionate to the size of the receptive field at the current feature map. The idea of anchors was popularized for image detection by Faster RCNN [5], and here it is extended to localization of objects in video. The $N_1 = 4$ space-time anchors are rectangular tubes in time of different aspect ratios and sizes. These aspect ratios and sizes are learned in a data-driven way from the ImageNet Vid training dataset. Space-time anchors are further discussed in §3.3.1.

At this point, the architecture has a single output prediction feature map produced by successive non-linear, pooling, and convolution operations. This mechanism is repeated 5 more times, each time producing a feature map of a different size. Hence, there are 6 feature maps, where each location produces N_i video proposals at feature map i, resulting in a total of 8732 video proposal predictions. As the network gets deeper, the receptive field of the feature map gets larger - hence, each location in smaller feature maps are trained to predict proposals for larger objects in the T frame sequence. The last 1×1 feature map is trained to predict either a large object spanning the whole video, or an object moving across the whole video extent in T frames. The $N_6 = 4$ predictions associated with the last feature map are shown in the bottom half of Fig. 3.2. The idea of using multiple locations on feature maps to generate bounding box proposals has been explored recently on images [21, 5, 11]. The Single Shot MultiBox Detector [14] extended that idea to use feature maps at different depths in the network to produce predictions of different scales. To the best of my knowledge this is the first extension of these ideas to video.

The network is run independently over every consecutive non-overlapping set of T frames, producing detections over each time window. Of course, since an object can last beyond a single time window seen by the network, detections need to be matched across time windows. Matching detections across multiple time windows would correspond to tracking the object over its lifetime. We perform simple bipartite matching on each available object category individually to produce long term tracks.

For future work, it might be fruitful to use the fixed size feature representation for each object for finding a matching score between objects across time windows. It would be feasible to train for a loss which enforces that the inner product between fixed sized feature representations of the same object across different time windows to be small.

Since the network is responsible for producing predictions over multiple frames, it generates spatio-temporal localization of objects in a unified pipeline which is end-to-end trainable. This approach for learning to generate short tracklets and detect objects is a departure

from recent video detection approaches where typically an image object detector is followed by a tracking pipeline [82, 122, 84, 86]. These approaches aim to detect, track objects and smooth predictions by an amalgam of algorithms.

3.3.1 Space-time Anchors

As discussed in §3.3, space-time anchors are the spatio-temporal extension of default bounding boxes used in Faster RCNN [5] and Single Shot Multibox Detector [14]. Anchors are employed for detection so that individual locations on a feature map can be used to localize multiple objects at different aspect ratios and scales. If not used, each feature map location can only be trained to regress a single prediction. This would be a harder learning problem, albeit having more training data (with higher variability) for each output prediction. In other words, anchors make localization learning easier by reducing the variations in scale and aspect ratio for each prediction. Anchors can also be seen as dividing the learning problem of detecting objects over different predictors - helping reduce the sample variability that each predictor has to contend with. Experiments on Faster RCNN [5] show that mAP rises by 3% when using 3 anchors of different aspect ratios, rather than a single anchor. mAP rises by another 1.1% when using 3 scales for each anchor, *i.e*. a total of 9 anchors. The notion of anchors is derived from pre convnet detection pipelines, where filters at varying scales and aspect ratios were typically used in a sliding window fashion. For instance, DPM [47] pools features at multiple pyramid levels and aspect ratios in order to detect objects of different shapes at varying distances.

As discussed in §3.4, by using space-time anchors, each prediction regresses an offset from an anchor it was trained for, rather than directly regressing to a ground-truth bounding box. Space-time anchors in this architecture only vary in aspect ratio, since scale is handled by having prediction layers at multiple convolutional feature maps with varying receptive field.

As mentioned before, the anchor sizes and aspect ratios are chosen in a data-driven way

before training begins. Finding anchors in a data-driven way has been known to increase the performance of detection models [123]. We take all bounding box annotations (height and width of objects) from ImageNet Vid training dataset, and divide them into 6 groups by the bounding box area. The division is done in a way to split the range of areas equally across all groups. The anchors generated for each group would be used for a particular prediction feature map. After collecting all the bounding boxes for a certain area group, we run k-means to find anchor sizes / aspect ratios for the respective prediction layer. Using 4 to 6 anchors per prediction layers ensures that around 94% of the objects have an overlap of above 0.5 with some space-time anchor.

In images, objects can appear at different aspect ratios and scales. In video, objects, in addition, can also move in different directions. Varying the global motion of an object would dramatically change filter responses. This suggests that some global object motion needs to be estimated before finding bounding box offsets every frame. Currently the architecture only uses rectangular space-time anchors, where the hope is that an object does not drift too far from the anchor location in T frames. These anchors, of course, might be sub-optimal for objects which move from one corner to another in T frames. Moreover, motion would introduce another source of variability in the training data for a single anchor, making it a harder learning problem. Also, once the model has learned to track objects going in a certain direction, it would be favorable if the model generalizes to tracking similar objects moving in other directions. If a model is trained to track a person moving left, it should be able to infer objects moving right. Some possible solutions we can explore in the future are:

1. Perhaps, the simplest solution is some temporal data augmentation. Currently, we randomly augment samples with horizontal flips, which would move the object in the opposite horizontal direction. Furthermore, sub-sampling frames at a fixed rate (for instance, selecting every second frame) would simulate samples with faster motion.

2. It might be possible to first estimate a flow field in convnet [112] which would give

dominant movement direction in each receptive field, helping decide the dominant motion of an anchor, before a prediction is made relative to it.

3. Another possibility is to first regress a direction vector (a starting and ending location in the first and last frame respectively) giving the general motion of the object, which could be used to regress bounding box offsets from linear motion.

4. If each frame in the video was moved to keep the object of interest centered, perhaps the network would learn more discriminative filters. This can be immitated by shifting the frames linearly in a single direction, and then only training with samples which closely match a spatio-temporally rectangular anchor. For instance, by moving every subsequent input frame 1 pixel south, 1 pixel east, it would center an object linearly moving 1 pixel north, 1 pixel west per frame. This procedure would be repeated using multiple preset directions, ensuring that the model only sees objects with most of its dominant motion eliminated. This scheme can be seen as the temporal counterpart of scale-space pyramid used for input to detectors. For instance, Viola Jones [46] applies the detector at 12 re-scaled versions of the input image, enabling it to find faces of different sizes. This *temporal frame movement* scheme should similarly allow finding objects moving in different directions.

5. Another approach is to learn deformable convolutions [98] such that the filters follow the motion of each individual object. This can be done in a way where the network learns to predict a 2D offset for each 3×3 spatial slice of each $3 \times 3 \times 3$ filter in the network. To allow for greater flexibility in the model, we can learn a 3D offset, where each filter slice can also move in time. This would allow the network to learn the same filter for slow or fast moving objects. An implementation of these ideas might help build time-equivariant feature representations in a convolutional neural network for various video tasks.

Currently, our algorithm uses data augmentation as given in solution (1), which helps in-

crease the number of samples seen by each anchor, and also possibly deal with linear global motion patterns.

3.4 Training

Before describing the training scheme in §3.5.2, details about all loss functions is given. Each video in ImageNet Vid can have multiple object annotations. Each video $\mathbf{V}$ has a set of objects $\{\boldsymbol{o}_{1*}, \ldots, \boldsymbol{o}_{m*}\}$, and each object $\boldsymbol{o}_{i*}$ is annotated with a list of bounding boxes. Each bounding box $\mathbf{b}_{i*}^{[t]} = \left[x_{i*}^{[t]}, \;\; y_{i*}^{[t]}, \;\; x_{i*}^{[t]} + w_{i*}^{[t]} - 1, \;\; y_{i*}^{[t]} + h_{i*}^{[t]} - 1\right]$, where $\left(x_{i*}^{[t]}, y_{i*}^{[t]}\right)$ is the top-left corner of the bounding box, and $w_{i*}^{[t]}$, $h_{i*}^{[t]}$ denote the width and height of the object at frame t. The ground-truth of object $\boldsymbol{o}_{i*}$ is a concatenation of all its annotations, and will be represented as $\left[\mathbf{b}_{i*}^{[q_i]}, \;\; \ldots, \;\; \mathbf{b}_{i*}^{[r_i]}\right]$. The variables q_i, r_i denote the starting and ending frame number for object $\boldsymbol{o}_{i*}$.

Every object can be divided into multiple training examples, where each sample is T frames long. We can use object $\boldsymbol{o}_{i*}$ to generate a training sample $\boldsymbol{s}_{i*}^{j}$, which would be defined by the subset sequence of the original annotation $\mathbf{b}_{i*}^{j} = \left[\mathbf{b}_{i*}^{[s_j]}, \;\; \ldots, \;\; \mathbf{b}_{i*}^{[e_j]}\right]$. Here, s_j, e_j are the starting and ending frame numbers for the sample, and $e_j - s_j + 1 = T$. Also note, $\mathbf{b}_{i*}^{j} \in \mathcal{N}_0^{4T}$, where $x_{i*}^{[t]}, w_{i*}^{[t]} \in [0, W-1]$, and $y_{i*}^{[t]}, h_{i*}^{[t]} \in [0, H-1]$, where W, H are the resized width and height of the video input to the network. Wherever needed, a sample would be denoted as $\boldsymbol{s}_{i*}^{[s_j, e_j]} \equiv \boldsymbol{s}_{i*}^{j}$. Hence, temporally consecutive samples for a particular object $\boldsymbol{o}_{i*}$ could be $\left\{\boldsymbol{s}_{i*}^{[5,\,20]}, \;\; \boldsymbol{s}_{i*}^{[21,\,36]}, \ldots\right\}$.

Each space-time anchor (see §3.3.1) in the model is indexed by $k \in [1, \;\; 8732]$, and it produces a localization prediction $\mathbf{p}_k$, and a classification prediction $\mathbf{c}_k$ or an objectness $\mathbf{q}_k$. Every ground-truth sample $\boldsymbol{s}_{i*}^{j}$ is associated to one of these anchors by maximizing the time volumetric bounding box overlap (the intersection over union between all pixels across T frames in the anchor and the object's bounding box). In practice, we find the best matching object for each of the 8732 locations. If the matching overlap is above 0.5, the anchor

is considered as a positive sample. To avoid missing objects, in addition we consider the anchor with the highest overlap with an object to be also marked as a positive sample. If the anchor at k is associated with a ground-truth sample $\boldsymbol{s}_{i*}^{j}$, then the prediction produced from that anchor would be referenced by $\mathbf{p}_k^{i,j}$, and $\mathbf{c}_k^{i,j}$. The predictions produced from a majority of anchors would not be associated to any ground-truth, and would be indicated as $\mathbf{p}_k^{-}$, and $\mathbf{c}_k^{-}$. Predictions $\mathbf{p}_k$ and $\mathbf{c}_k$ are produced by a simple convolutions and non-linearities over a common $\mathbf{f}_k$ feature representation[4]. These operations can be expressed as $\mathbf{p}_k = \mathbf{W}_k\mathbf{f}_k$ and $\mathbf{c}_k = \mathbf{U}_k\mathbf{f}_k$. Note that the feature representation $\mathbf{f}_k^{i,j}$, which is used to compute $\mathbf{p}_k^{i,j}$, $\mathbf{c}_k^{i,j}$, could be seen as feature representation for the object sample $\boldsymbol{s}_{i*}^{j}$.

3.4.1 Localization Loss

In order to spatio-temporally localize objects, the loss function would need to minimize some distance metric between $\mathbf{p}_k^{i,j}$ and $\mathbf{b}_{i*}^{j}$. The simplest loss would be to directly compute $\left\|\mathbf{p}_k^{i,j} - \mathbf{b}_{i*}^{j}\right\|_F$ after normalizing values to $[0, 1]$. Our initial experiments showed that this is a hard loss to minimize, hence, we would adopt the same output parameterization used in recent image object detection works [5, 14, 4, 10], where for each bounding box the network is trained to predict a scale-invariant offset from the space-time anchor center, and the ratio of height/width relative to the anchor in log-space. To compute these parameterizations, the center location of the sample $\boldsymbol{s}_{i*}^{j}$ needs to be computed in all frames $t \in [s_j, e_j]$:

$$\oplus_{i*}^{[t]} = x_{i*}^{[t]} + \left(w_{i*}^{[t]} - 1\right)/2, \qquad \ominus_{i*}^{[t]} = y_{i*}^{[t]} + \left(h_{i*}^{[t]} - 1\right)/2 \tag{3.1}$$

The anchor that is associated to this sample would also have an anchor center $\left(\oplus_{k\Psi}^{i,[t]}, \ominus_{k\Psi}^{i,[t]}\right)$, and an anchor width/height $w_{k\Psi}^{i,[t]}$, $h_{k\Psi}^{i,[t]}$. Symbols $(\bullet)_{k\Psi}^{i,[t]}$, refer to some variable for the anchor k, which is associated with ground-truth bounding box $\mathbf{b}_{i*}^{[t]}$. Currently we are using

[4]The feature representation $\mathbf{f}_k$ is produced by a non-linear operation over the corresponding location on the convolutional feature map output.

spatio-temporally rectangular anchors, hence the anchor center, width, and height does not change from one frame to the next, but we will use this notation for generality. Given these values for ground-truth and the corresponding anchor, we compute offsets for the ground-truth to the anchor on every frame:

$$\delta_{x,i*}^{[t]} = \left(\textcircled{:}_{i*}^{[t]} - \textcircled{:}_{k\Psi}^{i,[t]}\right) / w_{k\Psi}^{i,[t]}\,, \qquad \delta_{y,i*}^{[t]} = \left(\ominus_{i*}^{[t]} - \ominus_{k\Psi}^{i,[t]}\right) / h_{k\Psi}^{i,[t]}\,, \tag{3.2}$$

$$\delta_{w,i*}^{[t]} = \log\left(w_{i*}^{[t]} / w_{k\Psi}^{i,[t]}\right)\,, \qquad \delta_{h,i*}^{[t]} = \log\left(h_{i*}^{[t]} / h_{k\Psi}^{i,[t]}\right) \tag{3.3}$$

The network would produce a prediction $\mathbf{p}_k^{i,j} \in \mathbb{R}^{4T}$ relative to this anchor k. The list of values in the prediction would be $\mathbf{p}_k^{i,j} = \left[p_{x,k}^{i,[s_j]},\ p_{y,k}^{i,[s_j]},\ \ldots,\ p_{h,k}^{i,[e_j]}\right]$. The localization loss is computed by a smooth ℓ_1 metric between the ground-truth and prediction offsets:

$$\mathcal{L}_{\text{loc}}\left(\mathbf{p}_k^{i,j},\, \mathbf{b}_{i*}^{j}\right) = \frac{1}{4T} \sum_{t=s_j}^{e_j} \sum_{l=\{x,y,w,h\}} \text{smooth}_{\ell_1}\left(p_{l,k}^{i,[t]},\, \delta_{l,i*}^{[t]}\right) \tag{3.4}$$

$$\text{smooth}_{\ell_1}\left(v_1, v_2\right) = \begin{cases} 0.5(v_1 - v_2)^2 & \text{if } |v_1 - v_2| < 1 \\ |v_1 - v_2| - 0.5 & \text{otherwise} \end{cases} \tag{3.5}$$

As described above, $\mathbf{p}_k^{i,j} = \mathbf{W}_k \mathbf{f}_k^{i,j}$. Hence, given just the feature representation $\mathbf{f}_k^{i,j}$, the actual bounding box prediction for object $\boldsymbol{s}_{i*}^{j}$ can be computed by inverting the formulas in *Eq*. 3.2, 3.3, and 3.1:

$$\textcircled{:}_i^{[t]} = p_{x,k}^{i,[t]} w_{k\Psi}^{i,[t]} + \textcircled{:}_{k\Psi}^{i,[t]}\,, \qquad \ominus_i^{[t]} = p_{y,k}^{i,[t]} h_{k\Psi}^{i,[t]} + \ominus_{k\Psi}^{i,[t]}\,, \tag{3.6}$$

$$w_i^{[t]} = w_{k\Psi}^{i,[t]} \exp\left(p_{w,k}^{i,[t]}\right)\,, \qquad h_i^{[t]} = h_{k\Psi}^{i,[t]} \exp\left(p_{h,k}^{i,[t]}\right)\,, \tag{3.7}$$

$$x_i^{[t]} = \textcircled{:}_i^{[t]} - \left(w_i^{[t]} - 1\right)/2\,, \qquad y_i^{[t]} = \ominus_i^{[t]} - \left(h_i^{[t]} - 1\right)/2 \tag{3.8}$$

3.5 Classification Loss

With each spatio-temporal localization, the network also needs to predict the class of the object. This can be done by training the prediction layer with a $(M+1)$-way softmax classification layer $\mathbf{c}_k = \Big[c_{0,k},\, \ldots,\, c_{M,k}\Big]$. For ImageNet Vid $M = 30$ (see §3.2). Each anchor k's ground-truth object class label would be denoted as $\mathbf{c}_{k*} = \Big[c_{0,k*},\, \ldots,\, c_{M,k*}\Big]$: $c_{l,k*} \in \mathbb{B}$ [5]. If anchor k is associated with an object sample $\boldsymbol{s}_{i*}^{j}$, it should predict its object class, *i.e.* $c_{l,k*} = 1$ iff $\boldsymbol{s}_{i*}$ belongs to class l. In case anchor k is not associated with any object, then it should predict $c_{l=0,k*} = 1$ (note, $l = 0$ denotes 'background'). Hence, $\forall k,\ \sum_{l=0}^{M} c_{l,k*} = 1$. The cross-entropy loss for the softmax classification layer is,

$$\mathcal{L}_{\text{clsf}}(\mathbf{c}_k,\, \mathbf{c}_{k*}) = -\sum_{l=0}^{M} c_{l,k*} \log\left(\frac{e^{c_{l,k}}}{\sum_{l'=0}^{M} e^{c_{l',k}}}\right). \tag{3.9}$$

3.5.1 Objectness Loss

The detection pipeline above can be trivially extended to generate proposals. Rather than asking the model to classify an object into one of the pre-defined classes, the network can be trained to discriminate between objects and the background. Currently our model only has a classification loss, but we provide this exposition for completeness. The architecture can be changed by using a two-class softmax layer for classifying whether each prediction belongs to an object or not. This typically is referred to as an *objectness* score [48, 45]. Alternatively, the model can be trained to regress a single value by logistic regression. If an anchor k is associated with a ground-truth $\boldsymbol{s}_{i*}^{j}$, the ground-truth objectness score $q_{k*} = 1$; otherwise $q_{k*} = 0$. The classifier prediction can be represented as follows,

$$\mathbf{U}_k \mathbf{f}_k = \mathbf{q}_k = \Big[q_{k+}\,,\ \ q_{k-}\Big]^{\mathrm{T}} \tag{3.10}$$

[5] $\mathbb{B}$ denotes the class of binary numbers.

Now cross-entropy loss for the softmax classifier can be written as follows

$$\mathcal{L}_{\text{obj}}(\mathbf{q}_k,\, q_{k*}) = -q_{k*} \log \left(\frac{e^{q_{k+}}}{e^{q_{k+}} + e^{q_{k-}}} \right) \; - (1 - q_{k*}) \log \left(\frac{e^{q_{k-}}}{e^{q_{k+}} + e^{q_{k-}}} \right) , \qquad (3.11)$$

where the first half of the equation penalizes low objectness score despite the presence of an object, and the second half of the equation penalizes high objectness score when no object is present.

3.5.2 Training Scheme

The training starts by selecting a video $\mathbf{V}$, and randomly selecting a T frame sample. Each sample's T frames are forward passed through the network. Training proceeds by finding which anchors are associated with any available ground-truths as described in §3.3.1. This results in some anchors being associated with some object ($q_{k*} = 1$), and most are associated with the background ($q_{k*} = 0$). Since there are a lot more negative anchors than positives, we employ hard negative mining to pick 3 negatives samples for every positive chosen (for all experiments except focal loss experiments). This positive/background sample ratio is similar to RPN training [5]. Moreover, we also ensure that we select at least 5 negative anchors from each sample, so that we get some learning signal when the frames do not have any objects present in them. Once we know which anchors to use for training, the model can be optimized using the following loss:

$$\mathcal{L}(q_{k*},\, \mathbf{c}_k,\, \mathbf{c}_{k*},\, \mathbf{p}_k^{i,j},\, \mathbf{b}_{i*}^{j}) \;\; = \;\; \mathcal{L}_{\text{clsf}}(\mathbf{c}_k,\, \mathbf{c}_{k*}) \;\; + \;\; q_{k*} \cdot \mathcal{L}_{\text{loc}}\big(\mathbf{p}_k^{i,j},\, \mathbf{b}_{i*}^{j}\big) \qquad (3.12)$$

The majority of samples would be associated with no ground-truth *i.e.* $q_{k*} = 0$. These samples would just activate the objectness loss $\mathcal{L}_{\text{clsf}}(\bullet)$, and be trained to predict background over those anchors. We minimize this loss function using stochastic gradient descent.

In most of our experiments, each mini-batch contain 16 video samples, unless stated otherwise. Every mini-batch gets samples from randomly chosen videos. Although we

make sure that samples are not chosen in a way to bias learning toward certain object classes or certain videos. This happens because some samples occur much more frequently than others (there are only 67 tracks for tigers, but 1,246 for cars in ImageNet Vid training set), and some videos are longer than the rest. To tackle this, at the start every epoch, we pick a maximum of 15 samples per video, and limit the number of samples containing a certain category to 10,000.

3.6 Testing

At inference time, the network iterates over all T frame windows and incrementally builds object track proposals. Each T frames input to the network generates object detections, which are matched to the tracks and kept in memory. Object tracks are naturally created and eventually die as a result of the object matching process.

Every iteration the network is input T consecutive frames, which are resized to the resolution $W \times H$. The network would produce some localization/objectness predictions which are first pruned by a low confidence threshold of 0.01. This is followed by non-maximal suppression using the average per-frame overlap between object localizations (following the methodology in image detection algorithms [5, 14]). The reduced set of detections are sorted by their objectness score.

Now let's suppose the network had already produced detections on all frames preceding this set of T frames, and they have been used to generate tracks. The next step is to match the current set of detections to all the active tracks in the video. This matching is simply done by bipartite matching separately over all valid object classes. So, for instance, only 'dog' detections can be matched to 'dog' tracks. This also reduces the bipartite matching complexity - allowing us to run this tracking scheme at well over 100 frames per second.

To reduce false positive tracks, we employ two schemes to filter tracks. After tracks don't get matched to any detections, the track naturally dies. It add some robustness to the

tracking process, we employ a kalman filter to keep predicting the localization for tracks not matched to detections. We kill the track once it doesn't get matched for the third consecutive time. Note that this implies that the track can stay alive for $2T$ frames before it is marked as dead. Once the track is killed, we decide whether it was a valid track based on two criterion. If the average confidence of all the detections associated with that track were below a certain threshold, the track is marked as invalid. Secondly, if the final track length was below a certain number of frames, it is also marked invalid in this case. All tracks surviving this filtration processes are kept as the final set of tracks.

Most proposal generation methods perform inference in a batch [71, 80, 82, 84], or have to make multiple passes over videos before giving the final result [124]. Note that the inference in the proposed algorithm is run on-line, and can produce results with a delay of T frames.

3.7 Evaluation Metrics

3.7.1 Evaluating Space-time Proposals

For every video $\mathbf{V}_u$ in the validation/testing dataset, we have a set of objects $\left\{\boldsymbol{o}_{u,1*},\ \ldots,\ \boldsymbol{o}_{u,m_u*}\right\}$. The annotation for each object $\boldsymbol{o}_{u,i*}$ is represented as $\left[\mathbf{b}_{u,i*}^{[s_i]},\ \ \ldots,\ \ \mathbf{b}_{u,i*}^{[e_i]}\right]$, where each bounding box $\mathbf{b}_{u,i*}^{[t]} \in \mathbb{N}_0^4$, and $s_{u,i}$, $e_{u,i}$ are starting and ending frame numbers for the ground-truth. The forward pass of $\mathbf{V}_u$ through the network produces a set of n_u variable length object tracks/video proposals $\left\{\boldsymbol{o}_{u,1},\ \ldots,\ \boldsymbol{o}_{u,n_u}\right\}$. Each video proposal $\boldsymbol{o}_{u,k}$, running from frame $s_{u,k}$ to $e_{u,k}$, has bounding box predictions $\left[\mathbf{b}_{u,k}^{[s_k]},\ \ \ldots,\ \ \mathbf{b}_{u,k}^{[e_k]}\right]$. Given two bounding boxes $\mathbf{b}_{u,i*}^{[t]}$, $\mathbf{b}_{u,k}^{[t]}$ in the same frame, we can compute the intersection and union

quantities as follows,

$$\left|\mathbf{b}_{u,i*}^{[t]} \cap \mathbf{b}_{u,k}^{[t]}\right| = \max\left(0,\ \min\left(x_{u,i*}^{[t]} + w_{u,i*}^{[t]},\ x_{u,k}^{[t]} + w_{u,k}^{[t]}\right) \ -\ \max\left(x_{u,i*}^{[t]},\ x_{u,k}^{[t]}\right)\right) \times \max\left(0,\ \min\left(y_{u,i*}^{[t]} + h_{u,i*}^{[t]},\ y_{u,k}^{[t]} + h_{u,k}^{[t]}\right) \ -\ \max\left(y_{u,i*}^{[t]},\ y_{u,k}^{[t]}\right)\right) \tag{3.13}$$

$$\left|\mathbf{b}_{u,i*}^{[t]} \cup \mathbf{b}_{u,k}^{[t]}\right| = w_{u,i*}^{[t]} h_{u,i*}^{[t]} \ +\ w_{u,k}^{[t]} h_{u,k}^{[t]} \ -\ \left|\mathbf{b}_{u,i*}^{[t]} \cap \mathbf{b}_{u,k}^{[t]}\right| \tag{3.14}$$

Image object proposals are typically evaluated using overlap score (intersection over union / jaccard index) [45]. A similar overlap score can be computed between a video proposal $\boldsymbol{o}_{u,k}$ and a ground-truth $\boldsymbol{o}_{u,i*}$ can be computed in two different ways. As used in [6], one can compute an average of all overlaps per frame as follows,

$$\mathcal{J}_1\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right) = \frac{1}{\mathcal{T}_{i,k}} \sum_{t=\max(s_{u,i},\, s_{u,k})}^{\min(e_{u,i},\, e_{u,k})} \frac{\left|\mathbf{b}_{u,i*}^{[t]} \cap \mathbf{b}_{u,k}^{[t]}\right|}{\left|\mathbf{b}_{u,i*}^{[t]} \cup \mathbf{b}_{u,k}^{[t]}\right|}, \tag{3.15}$$

$$\text{where} \qquad \mathcal{T}_{i,k} = \max(e_{u,i},\ e_{u,k}) - \min(s_{u,i},\ s_{u,k}) + 1\,.$$

This overlap score penalizes all frames equally, regardless of whether the object was small or large in a particular frame.

For some applications it is important to have a score where the penalty is proportional to the size of the object in a frame. Many vision systems would require more accurate prediction of objects when nearby and, hence, occupying relatively larger portion of the receptive field (think of self-driving cars recognizing nearby cars, or a robot trying to manipulate objects in reach). For such applications, it is important to weight overlap of proposals with the ground-truth according to its size in a specific frame,

$$\mathcal{J}_2\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right) = \frac{\sum_{t=\max(s_{u,i},\, s_{u,k})}^{\min(e_{u,i},\, e_{u,k})} \left|\mathbf{b}_{u,i*}^{[t]} \cap \mathbf{b}_{u,k}^{[t]}\right|}{\sum_{t=s_{u,i}}^{e_{u,i}} w_{u,i*}^{[t]} h_{u,i*}^{[t]} \ +\ \sum_{t=s_{u,k}}^{e_{u,k}} w_{u,k}^{[t]} h_{u,k}^{[t]} \ -\ \sum_{t=\max(s_{u,i},\, s_{u,k})}^{\min(e_{u,i},\, e_{u,k})} \left|\mathbf{b}_{u,i*}^{[t]} \cup \mathbf{b}_{u,k}^{[t]}\right|}\,. \tag{3.16}$$

We would use this IoU metric for evaluating our models in §3.8.

As typically done in image object proposal methods, for each ground-truth a single maximum overlap score is selected across all video proposals,

$$\mathcal{J}\left(\boldsymbol{o}_{u,i*}\right) = \max_{k} \mathcal{J}\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right) \tag{3.17}$$

Now, a single average overlap number can be easily generated by computing the maximum overlap for each ground-truth object across all videos, $\frac{1}{\sum_u m_u} \sum_{u,i} \mathcal{J}\left(\boldsymbol{o}_{u,i*}\right)$. Note, $\sum_u m_u$ is the total number of object ground-truths in the complete dataset. This score can easily be adjusted to weight objects which have longer tracks if it is important to measure long-term tracking performance.

Recall is another popular measure to gauge ability to generate object proposals, given a fixed overlap threshold $\mathcal{T}_\mathcal{J}$. It can be computed as $\frac{1}{\sum_u m_u} \sum_{u,i} \left[\mathcal{J}\left(\boldsymbol{o}_{u,i*}\right) > \mathcal{T}_\mathcal{J}\right]$, where $[\bullet]$ is the iverson bracket notation. It might be useful to penalize recall more for incorrect localizations on longer tracks — which would happen when a method fails to track an object for extended periods:

$$\mathcal{R}_{\mathcal{T}_\mathcal{J}} = \frac{\sum_{u,i} \left[\mathcal{J}\left(\boldsymbol{o}_{u,i*}\right) > \mathcal{T}_\mathcal{J}\right] \left(e_{u,i} - s_{u,i} + 1\right)}{\sum_{u,i} e_{u,i} - s_{u,i} + 1} \tag{3.18}$$

We compute a class agnostic $\mathcal{R}_{.5}$ for all our experiments in §3.8.

Since recall is based on a single threshold $\mathcal{T}_\mathcal{J}$, it either rewards loose localization (at low $\mathcal{T}_\mathcal{J}$) or penalizes reasonable proposals missing some extremity of an object (at high $\mathcal{T}_\mathcal{J}$). Hosang *et al*. [45] demonstrated that averaging recall in the range $[0.5, 1.0]$ not only remedies this problem, but correlates closely with detection performance. Recently, average recall has been adjusted to operate in the range $[0.5, 0.95]$ for the Microsoft COCO challenge [125] — this avoids penalizing methods at recall thresholds closer to 1.0. It is possible to extend these benefits to benchmarking video proposal methods by averaging the recall score given in *Eq*. 3.18.

Every proposal also has an average objectness score $q_{u,k}$ computed by aggregating objectness scores over all $L_{u,k}$ predictions made for that video proposal $\boldsymbol{o}_{u,k}$:

$$q_{u,k} = \frac{1}{L_{u,k}} \sum_{j=1}^{L_{u,k}} \frac{q^j_{u,k+}}{q^j_{u,k+} + q^j_{u,k-}} \tag{3.19}$$

This score can be used to sort all video proposals by confidence. This is useful for observing overlap and recall measures when limited to a certain number of proposals — better scoring functions will allow high overlap and recall even with a low limit on the number of proposals. Given a certain threshold $\mathcal{T}_n$ for the number of proposals that can be chosen per video, $q_{u,k}$ can be either used to select top $\mathcal{T}_n$ scoring proposals, or as a distribution to sample proposals from per video. To compare competing proposal methods recall and average overlap can be plotted against varying $\mathcal{T}_n$. Successful methods would not only have a good limit performance (where $\mathcal{T}_n = \infty$), but also perform well when allowed only a small number of proposals. For our experimental results in §3.8, we show $\mathcal{R}_{.5}$ scores at both $\mathcal{T}_n = \infty$ and $\mathcal{T}_n = 10$. We found in our experiments that 10 proposals/detections reaches quite close to the limit performance of the model.

Using these metrics, the proposed method would be compared to competing video proposal methods, such as [80, 71, 6, 73].

3.7.2 Evaluating Detection Performance

Image detectors are evaluated typically using the mean average precision (mAP) metric [49, 13]. It is designed to penalize detector outputs which misses object instances, duplicate predictions, and generates false positives. Average precision (AP) is computed individually for each object category in the dataset, and then averaged across categories to give mAP. The recall score in §3.7.1 can be easily extended to compute average precision for video object detectors.

A scheme was described how to extend the proposed video proposal architecture to

predict object classifications §3.4. During inference time (see §3.6), the method produces classification predictions every T frames for an object track. These classifications need to be aggregated to produce a single classification per object track. One possibility is to sum normalized classification scores output from the softmax layer, similar to the aggregation of confidence scores (see *Eq*. 3.19). We instead take a more simple approach. While attempting to stitch together T frame detection predictions, we only consider objects of the same category in the matching process. This naturally generates tracks for one object category at a time - and the process also dictates what is object class of the track.

Given the set of all video proposals, ground-truths, and their classes, we can compute recall and precision over all videos. Recall $\mathcal{S}_{\mathcal{T}_{\text{conf}},\mathcal{T}_{\mathcal{J}}}$ is the fraction of ground-truths which were correctly detected given an overlap threshold of $\mathcal{T}_{\mathcal{J}}$, above a prediction confidence of $\mathcal{T}_{\text{conf}}$. Similarly, precision $\mathcal{P}_{\mathcal{T}_{\text{conf}},\mathcal{T}_{\mathcal{J}}}$ is the fraction of detections that are correct. They can be computed for each class label l as,

$$\mathcal{S}_{\mathcal{T}_{\text{conf}},\mathcal{T}_{\mathcal{J}}}(l) = \frac{\sum_u \sum_{\forall k\,:\,c_{u,k\diamond}=l} \left[q_{u,k} > \mathcal{T}_{\text{conf}}\right] \overbrace{\left[\exists i \in \mathcal{I}_{l,u*} \;:\; \mathcal{J}\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right) > \mathcal{T}_{\mathcal{J}}\right]}^{\boldsymbol{o}_{u,k} \text{ has a matching ground-truth of class } l}}{\sum_u \left|\mathcal{I}_{l,u*}\right|}, \tag{3.20}$$

$$\mathcal{P}_{\mathcal{T}_{\text{conf}},\mathcal{T}_{\mathcal{J}}}(l) = \frac{\sum_u \sum_{\forall k\,:\,c_{u,k\diamond}=l} \left[q_{u,k} > \mathcal{T}_{\text{conf}}\right] \left[\exists i \in \mathcal{I}_{l,u*} \;:\; \mathcal{J}\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right) > \mathcal{T}_{\mathcal{J}}\right]}{\sum_u n_u}, \tag{3.21}$$

where $\mathcal{I}_{l,u*}$ is the set of class l ground-truths in a video $\mathbf{V}_u$, *i.e*. the set of ground-truths satisfying this condition $\left[l = \arg\max_{l' \in [0,\ldots,M]} c_{l',u,i*}\right]$ (see §3.4). Hence, $\left|\mathcal{I}_{l,u*}\right|$ refers to the number of class l ground-truths in video $\mathbf{V}_u$. The set $\{\forall k \;:\; c_{u,k\diamond} = l\}$ indicates only video proposals in $\mathbf{V}_u$ classified with category l. Note, that the term $\left[\exists i \in \mathcal{I}_{l,u*} \;:\; \mathcal{J}\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right) > \mathcal{T}_{\mathcal{J}}\right]$ can only be activated once across all video proposals with label l, *i.e*. there is one-to-one matching between video proposals and ground-truths. Note that precision/recall measures can be extended to penalize methods having false-negatives for ground-truths with longer

tracks (similar to *Eq*. 3.18).

Given recall and precision at a range of confidence thresholds $\mathcal{T}_{\text{conf}}$, average precision $\mathcal{A}_{\mathcal{T}_\mathcal{J}}(l)$ for class l can be computed by the scheme described in [49] — which is similar to taking the area under the precision-recall graph. As briefly mentioned before, average precisions can be averaged across all classes $l \in \{1, M\}$ [6] to produce mAP score, *i.e.* $\frac{1}{M}\sum_l \mathcal{A}_{\mathcal{T}_\mathcal{J}}(l)$. Similar to the average recall score for proposals, mAP can also be averaged across a range of thresholds $\mathcal{T}_\mathcal{J}$ [125]. This more accurately measures a detectors performance for correct classification and localization. For our experiments in §3.8, we give mAP score at threshold $\mathcal{T}_\mathcal{J} = 0.5$, and an averaged mAP at 10 equally spaced thresholds in the range $[0.5, 0.95]$, a scheme which was made standard for Microsoft COCO detection challenge [125].

3.8 Results

This section gives detailed ablation studies for the video object detection model presented in this chapter. All models tested are 3D convolution variants of ResNet-34 [92] - where the number of 3D convolutions depends on the number of input frames the model is trained for. All experiments were done using ImageNet Vid training and validation. Details about the dataset are given in §3.2. Every experiment after §3.8.2 also uses motion simulated samples from ImageNet Detection [13]. Also, the weights of all video models are initialized using the single frame detector for all layers that are common between the two models. 3D convolutions are initialized from 2D weights using the inflation technique described in [115]. All models are trained for 90-100K iterations using stochastic gradient descent. Each model is trained with a batch size of 16, unless specified otherwise (models with larger number of input frames have a higher memory footprint). The input frames are resized to $600x600$, and we use horizontal flips, color jittering in HSL space, and random cropping

[6] Background class $l = 0$ is not used for computing precision-recall measures.

augmentations during training. All results are based on outputs from a single trained model (no model ensembles).

For each experiment, we give several metrics for evaluating performance. As an example we would refer to the metrics given in Table 3.7. For each model we first specify the total number of learnable parameters in the model, as well as the number of 3D convolutional operations. For some experiments the number of training samples might be larger than other experiments, hence we specify the number of samples whenever useful for analysis. We also give the final training and testing losses - which is useful to observe when comparing models with the same loss, as well as getting a sense of the generalization gap. We also give the contribution of the localization and classification loss in the total loss. As mentioned in §3.7.1, we give the final IoU $\mathcal{J}_2\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right)$ achieved over anchors matched to objects on all validation samples. The classification accuracy measures a model's ability to correctly classify the object as one of the M positive classes. For a detector, of course this does not paint the complete picture since the model also needs to separate object from background. To get a sense of performance on this task, we also give the "+ve objectness accuracy" and "-ve objectness accuracy." These numbers show the percentage of positive and negative anchors in validation samples the model is able to correctly classify respectively. We also give the average number of detections each model produces after NMS, and the percentage of objects it was able to recall at the overlap threshold 0.5 in a class agnostic way ($\mathcal{R}_{.5}$). Since it can be hard to compare recall of different models producing varying number of detections, we also give the recall when you select 10 highest scoring detections.

The gold standard metric for detection performance is mAP, as described in §3.7.2. We compute mAP at an overlap threshold of 0.5, as well as the average over the range $[0.5, 0.95]$, where the IoU is computed over all the T frames the detector is trying to predict for ($\mathcal{J}_2\left(\boldsymbol{o}_{u,i*}, \boldsymbol{o}_{u,k}\right)$). To compare to single frame detectors, we also consider each multi-frame detector's output over each frame individually and compute the single frame mAP at

overlap threshold of 0.5.

3.8.1 Effect of Normalizing for Localization Loss

In this experiment we train two models, both accepting a 3 frame input, and producing object detection predictions over all 3 frames. The difference between the two models is that one uses is the loss defined in *Eq*. 3.12, wheres the other model is trained with a localization loss which is normalized by the number of frames:

$$\mathcal{L}(q_{k*},\, \mathbf{c}_k,\, \mathbf{c}_{k*},\, \mathbf{p}_k^{i,j},\, \mathbf{b}_{i*}^{j}) \;=\; \mathcal{L}_{\text{clsf}}(\mathbf{c}_k,\, \mathbf{c}_{k*}) \;+\; \frac{1}{3} \cdot q_{k*} \cdot \mathcal{L}_{\text{loc}}\big(\mathbf{p}_k^{i,j},\, \mathbf{b}_{i*}^{j}\big) \tag{3.22}$$

Table 3.3: Effect of normalizing loss with the number of frames being predicted for. Both models were trained for 3 frame input, and make predictions over all 3 frames.

Method	# Parameters	# 3D Conv. Ops.	Train Loss	Test Loss	Test Localization Loss	Test Classification Loss	IoU	Classification Accuracy	+ve Objectness Accuracy	-ve Objectness Accuracy	Avg. # dets.	Recall	Recall @ 10 dets.	mAP $_{.5}$	Single Frame mAP $_{.5}$	mAP $_{.5:.95}$
No normalization	29.1M	3	2.69	5.27	2.37	2.90	**.801**	.731	.446	**.961**	19	.903	**.876**	.598	**.599**	**.369**
$1/3$ normalization	29.1M	3	1.76	3.69	0.82	2.87	.795	**.738**	**.457**	**.960**	18	.894	.870	**.602**	**.602**	**.369**

Table 3.3 shows results from this experiment. As expected, the total test loss and localization loss is larger when not normalizing. But, interestingly it does not have any adverse effect on the performance. Although, if we use an un-normalized loss for larger number of frames, the localization loss might start to over shadow the classification loss, and perhaps lead to a lower detection performance. Since we do not find any evidence of improvement from an un-normalized localization loss, we choose to use the normalized loss for all following experiments. This also allows us to easily compare the test loss across models trained for different T.

3.8.2 Motion Simulated Samples from ImageNet Detection [13]

Most competing methods for video object detection train for a single frame model, giving them access to many image datasets. Most methods working with the ImageNet Vid dataset [12] choose to expand their training set with samples from ImageNet Detection dataset [13]. This image dataset is a logical choice because all 30 ImageNet Vid categories also exist in ImageNet Detection dataset. We also learned from training Image detectors on ImageNet Vid that adding ImageNet Detection to the training set can add ~ 4 mAP to the detector's performance.

In order to gain the benefit of ImageNet Detection, we would need to *simulate* video samples from images. We adopt a simple scheme where we take an image, and crop a moving a rectangle from the image so as to mimic some planar camera motion. The motion pattern itself is sampled from the movement of objects in ImageNet Vid training. We need to generate different number of frames T from each image depending on the network input.

Table 3.4: Effect of training with simulated video clips from ImageNet Detection [13] training set, in addition to ImageNet Vid. All 4 models were trained for 3 frame input, and make predictions for the middle frame, hence $\text{mAP}_{.5}$ is computed over single frame predictions.

Method	# Parameters	# 3D Conv. Ops.	# Training Samples	Train Loss	Test Loss	Test Localization Loss	Test Classification Loss	IoU	Classification Accuracy	+ve Objectness Accuracy	-ve Objectness Accuracy	Avg. # dets.	Recall	Recall @ 10 dets.	$\text{mAP}_{.5}$	$\text{mAP}_{.5:.95}$
No ImageNet Det	28.7M	3	.27M	1.66	3.63	0.77	2.86	**.806**	.739	**.448**	**.964**	19	.904	.877	.613	.390
1 video clip/im	28.7M	3	.44M	1.86	3.39	0.76	2.63	**.806**	.761	.440	.962	37	.932	.886	.643	.403
2 video clip/im	28.7M	3	.62M	1.86	3.35	0.75	2.60	**.807**	**.769**	.442	.963	35	.935	**.892**	.653	.409
4 video clip/im	28.7M	3	.97M	1.92	3.36	0.76	2.60	**.805**	.765	.434	**.966**	43	.936	.884	**.657**	**.413**

Table 3.4 shows the effect of adding motion simulated samples from ImageNet Detection. To understand how much ImageNet Detection data is useful, we simulate varying number of multi-frame samples from each image in the dataset. We observe that 4 samples simulated from every ImageNet Detection image (the ones that have at least one object from

the M overlapping classes) drastically improves the multi-frame detector performance. It is apparent from both the classification loss and accuracy that adding samples from ImageNet Detection greatly improves the class discriminative ability of the model. Although we also see that we pay cost of decreasing +ve objectness score with increasing number simulated samples. All models in the following experiments are trained with 4 motion simulated samples per image.

3.8.3 More Convolution Layers before each Prediction Layer

Here we experiment whether adding more capacity to the model before each of the 6 prediction branches improves detection performance. In our standard model, our first prediction layer is preceded by a 256 channel convolution, and all other prediction layers are preceded by 128 channel convolutions. We adjust this model by replacing all these 6 layers such that each prediction layer is preceded by two convolutions (256 and 128 channels) with non-linearities. One noteworthy downside to this experiment is that lesser number of weights are initialized from the single frame model, because the new additional convolution layers did not previously exist.

Table 3.5: Effect of adding more convolutional filters (with ReLU) right before each prediction layer. The "standard model" has a single 128 or 256 channels convolutional layer added to the main branch's feature map before it is used to make a prediction. The "more conv. layers" model has two convolutions (256 channels followed by 128) before it is used to make predictions. Both models were trained for 3 frame input, and make predictions over all 3 frames. 1.78M samples from ImageNet Vid + ImageNet Detection were used for training.

Method	# Parameters	# 3D Conv. Ops.	Train Loss	Test Loss	Test Localization Loss	Test Classification Loss	IoU	Classification Accuracy	+ve Objectness Accuracy	-ve Objectness Accuracy	Avg. # dets.	Recall	Recall @ 10 dets.	mAP$_{.5}$	Single Frame mAP$_{.5}$	mAP$_{.5:.95}$
Standard model	28.8M	2	1.97	3.44	0.82	2.61	**.793**	**.768**	**.440**	**.962**	38	.922	**.876**	**.645**	**.646**	**.393**
More conv. layers	28.9M	2	2.19	3.51	0.85	2.66	.789	.759	.427	**.964**	41	.922	**.876**	.632	.633	.383

Table 3.5 shows the comparative results between the standard model, and the new model

with additional convolution layers before each prediction layer. It is clear from the results that adding layers with scratch weights before the prediction layers hampers the detectors final performance. Even though, it is possible that training these additional layers for the single frame detector, and using them as pre-trained weights, might improve the performance of all models considered in our experiments. Due to the nature of the current results, we choose to use the standard model for the experiments that follow.

3.8.4 Focal Loss [126]

Recently, Focal Loss [126] has been shown to improve performance of detectors based on SSD-like architectures. This scheme is attractive because ROI-pooling based detectors [5, 89] are known to be slower due to the pooling operation - which SSD architectures avoid by directly making detection predictions on multiple scale feature maps. The goal of Focal loss is to direct the model's loss more toward positive or negative samples which incur a larger loss. It encourages using all the negative samples available rather than doing hard negative mining. For our experiments we select 20 times more hard-negatives than positives from each video sample. For all other experiments, we choose to sample only 3 negatives for each positive.

Table 3.6: Effect of using focal loss [126] for the classification loss $\mathcal{L}_{\text{clsf}}(\bullet)$. The focal loss model was trained with $\gamma = 1$, and uses 20 times more negative samples than positives, whereas the standard model, like in all other experiments, uses 3 times more negatives than positives. Both models were trained for 3 frame input, and make predictions for the middle frame, hence $\text{mAP}_{.5}$ is computed over single frame predictions. 1.78M samples from ImageNet Vid + ImageNet Detection were used for training.

Method	# Parameters	# 3D Conv. Ops.	Train Loss	Test Loss	Test Localization Loss	Test Classification Loss	IoU	Classification Accuracy	+ve Objectness Accuracy	-ve Objectness Accuracy	Avg. # dets.	Recall	Recall @ 10 dets.	$\text{mAP}_{.5}$	$\text{mAP}_{.5:.95}$
Standard model	28.4M	2	1.94	3.36	0.76	2.60	**.805**	**.766**	**.455**	.960	40	.934	.888	**.652**	**.409**
Focal loss $\gamma = 1$	28.4M	2	1.42	2.72	0.74	1.98	**.808**	**.767**	.451	**.994**	55	.949	**.900**	**.654**	**.413**

Table 3.6 shows results of two models: one is trained with a standard loss, and another uses focal loss with a $\gamma = 1$. We didn't see any large differences when we change γ, so we only report results from a single parameter. The results show that there is hardly any positive effect of using focal loss. The only significant effect we see in the model's ability to classify negative anchors - which is expected because this model trains on 6.5 times more negative samples than the standard model. Due to no statistically significant increase in mAP, we choose to use the standard loss for all our other experiments. Our findings are in line with other recent methods, like [127].

3.8.5 Middle Frame Prediction from Multiple Frame Input

The exposition given in §3.4.1, the network is trained to predict objects spanning over all the input frames. One argument against this approach is that with the increase in complexity of the input (more frames are input to the model), the output dimensionality has also increased. We try to decrease the dimensionality of the output in this experiment by asking the network to detect objects in the middle frame, given a T frame input. For instance, the 5 frame model, which receives frames $\{0, 1, 2, 3, 4\}$, is trained to make predictions only on frame 2. In this experiment, the anchors are still constructed over all the frames, so the object needs to have a significant overlap with the anchor over all input frames. This form of prediction is equivalent to training the network to make just object localization predictions in the middle frame.

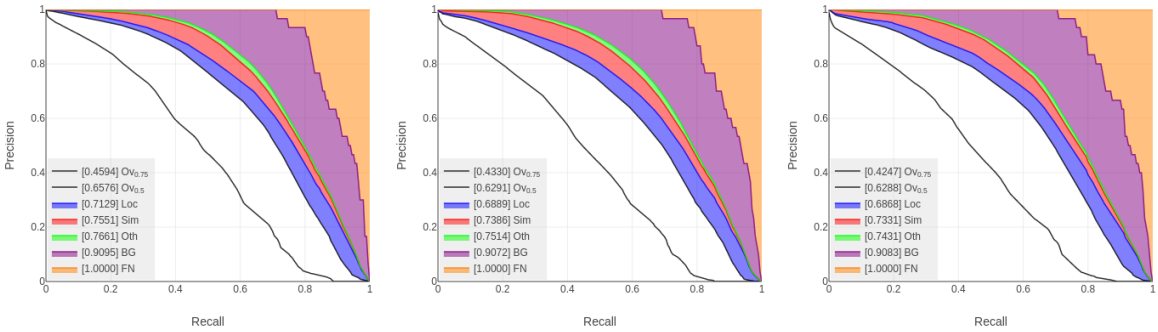
We train 3 different models which accept 3 frame inputs, 5 frame inputs, and a 7 frame inputs. All make prediction only at the middle frame. We compare the results directly to a single frame object detector in Table 3.7. As mentioned in the table, different models are trained with a slightly different setting for batch sizes due to GPU memory constraints. It is clear that the detection performance drops with models with increasing number of frames. This is expected since the input complexity has increased but the number of samples available has roughly remained the same. Only the 3 frame input model manages to do better

Table 3.7: Experiments on models which accept different number of input frames, and detect objects on the middle frame (hence mAP is computed over single frame predictions). The "single frame" is the baseline model trained to detect objects given a single image. All other models accept varying number of input frames and output detections on the middle frame. The other difference is the batch size used for training models. The single frame model was trained with a batch size of 32; both 3 frame and 5 frame input models were trained with a batch size of 16, whereas 7 frame input model was trained with a batch size of 12 samples. These choices were made considering the memory limitations of the 4 GPUs used for training.

Method	# Parameters	# 3D Conv. Ops.	# Training Samples	Train Loss	Test Loss	Test Localization Loss	Test Classification Loss	IoU	Classification Accuracy	+ve Objectness Accuracy	-ve Objectness Accuracy	Avg. # dets.	Recall	Recall @ 10 dets.	$mAP_{.5}$	$mAP_{.5:.95}$
Single frame	28.3M	0	1.26M	1.80	3.33	0.74	2.59	**.807**	.763	**.444**	**.964**	39	.937	**.893**	.650	.410
3 frame inp.	28.4M	2	1.78M	1.94	3.33	0.75	2.58	**.807**	**.768**	**.445**	**.964**	43	.939	**.891**	**.658**	**.414**
5 frame inp.	29.1M	4	1.77M	2.06	3.47	0.77	2.69	.803	.753	.423	**.964**	47	.934	.882	.629	.390
7 frame inp.	31.7M	6	1.76M	2.21	3.46	0.79	2.67	.800	.761	.411	**.965**	50	.939	.879	.629	.387

than the baseline single frame model. The drop in performance over increasing number of frames can be attributed to the a large drop in +ve objectness accuracy, even though negative objectness accuracy is maintained. This implies that the models with more input frames see a dramatic increase in false negatives, i.e. it confuses more objects as background. Fig. 3.4 shows the +ve objectness validation accuracy as training progresses over each model. Apart from the 3 frame model, it is clear that both the 5 frame and 7 frame model generates more false negatives over the training phase. This trend can also be seen directly in the comparative graph for +ve objectness validation accuracy. This indicates that models with more input frames increasingly fail to capture objects over any matching anchor. This problem is partially due to more objects not getting matched to an anchor with a higher overlap. This is a natural consequence of using space-time anchors which are static in time.

Fig. 3.3 shows the detection analysis of the 3 multi-frame detectors [128]. This gives an alternative view to our problem, and it shows that the increase in false negatives can be partially attributed to localization problems. This is corroborated with decrease in localization



(a) 3 frame input/1 frame output (b) 5 frame input/1 frame output (c) 7 frame input/1 frame output

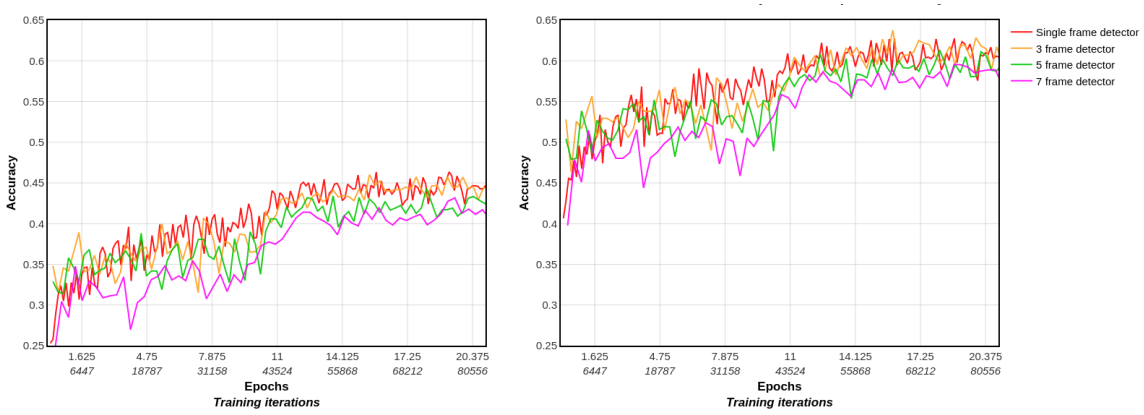
Figure 3.3: Detection analysis [128] for 3/5/7 frame input-middle frame detectors, evaluated in Table 3.7. These results were generated on input samples of lengths 3/5/7 frames respectively from ImageNet Vid validation set. Each model generated detection results over the middle frame given its multi-frame input. Each video was sampled with a stride of 2 frames which gave a total of 85K samples.

accuracy in Table 3.7.

3.8.6 Multiple Frame Prediction from Multiple Frame Input

This section shows the results for detection models trained with a multi-frame input, with localization output spanning all the frames. These experiments are similar to §3.8.5, except that the localization in these trained models was done over all input T frames. Table 3.8 compares results for models trained for 3, 5, and 7 frame inputs to a single frame detector baseline. Since all models predict localizations over T frames, $\text{mAP}_{.5}$ and $\text{mAP}_{.5:.95}$ values are not directly comparable across models. Hence, we also compute the single frame $\text{mAP}_{.5}$, which scores all detections on a per frame basis. Hence, a T frame model would be scored for detections made individually on each of the T frames.

Like in Table 3.7, it is apparent that the same set of problems are faced by detectors when predicting a multi-frame output. The false negatives produced by the network drastically increases (the +ve objectness accuracy drops by more than 10% between the 3 frame and 7 frame model). Moreover, the localization and classification accuracy drops more sharply compared to the experiments in §3.8.5. The rising train and test less is indicative



(a) +ve objectness accuracy during training (b) +ve objectness capture accuracy during training

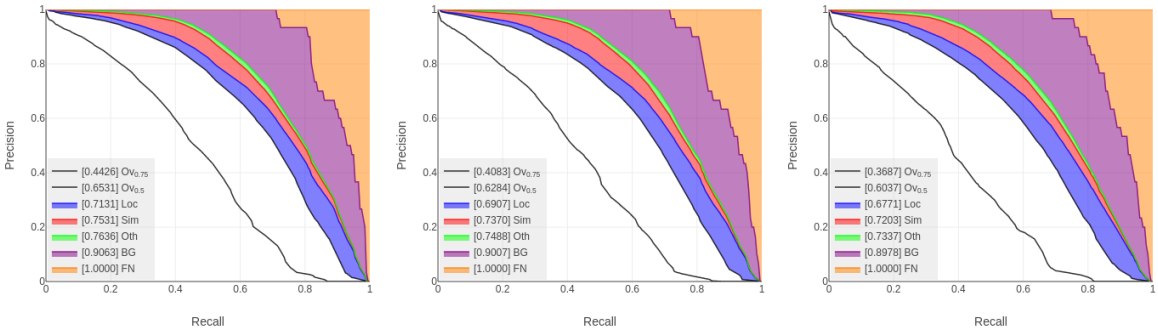
Figure 3.4: These two graphs show the validation +ve objectness accuracy computed every time after training on 15K samples. Both graphs show validation accuracy for all the models in Table 3.7, which are trained on multiple frame input, but make predictions over only the middle frame. The graph on the right gives the validation accuracy of the model, which is the percentage of +ve anchors (see §3.3.1) correctly classified as objects. Since many anchors can be associated to the same object, the graph on the right shows the ability of the model to classify at least one for each object correctly.

Table 3.8: Experiments on models which accept different number of input frames, and make video object detections (detections with space-time bounding boxes) in all the frames. The "single frame", like in Table 3.7, is the baseline model trained to detect objects given a single image. All other models accept varying number of input frames and output detections on the middle frame. The same batch sizes were used as the experiments in Table 3.7.

Method	# Parameters	# 3D Conv. Ops.	# Training Samples	Train Loss	Test Loss	Test Localization Loss	Test Classification Loss	IoU	Classification Accuracy	+ve Objectness Accuracy	-ve Objectness Accuracy	Avg. # dets.	Recall	Recall @ 10 dets.	$mAP_{.5}$	Single Frame $mAP_{.5}$	$mAP_{.5:.95}$
Single frame	28.3M	0	1.26M	1.80	3.33	0.74	2.59	**.807**	.763	.444	**.964**	39	.937	**.893**	-	.650	-
3 frame inp.	28.8M	2	1.78M	1.90	3.42	0.82	2.59	.794	**.771**	**.459**	**.961**	31	.916	.877	**.653**	**.654**	**.400**
5 frame inp.	29.7M	4	1.77M	2.17	3.52	0.87	2.65	.784	.758	.435	**.963**	35	.907	.865	.628	.629	.373
7 frame inp.	32.7M	6	1.76M	2.30	3.66	0.95	2.71	.772	.749	.403	**.965**	49	.894	.845	.604	.607	.347

of these metrics as well. Note that the average number of detections produced by the 7 frame detector increased by 58% as compared to a 3 frame detector, while the single frame detection performance dropped by 8% = .047 mAP points.

Fig. 3.5 shows that if we ignore for localization errors, class confusion, and background confusion, all models have very similar 'FN' performance (errors purely made due to de-



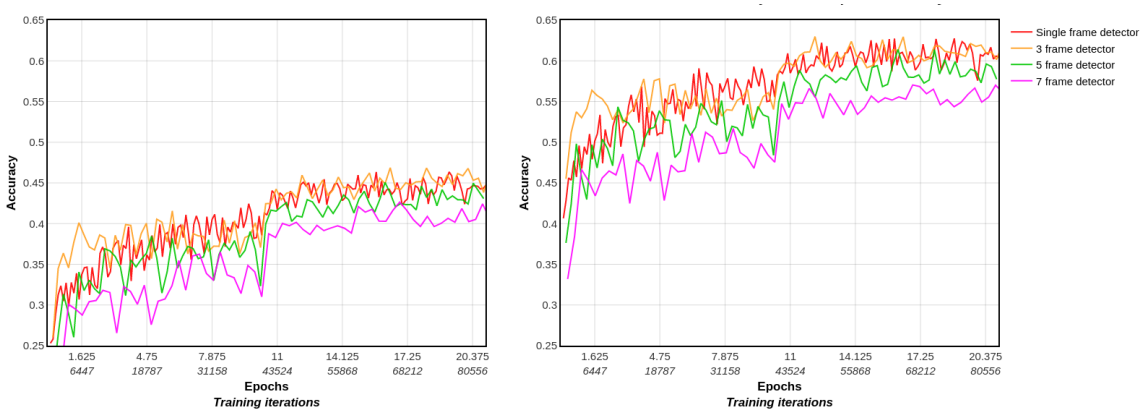
(a) 3 frame input/output (b) 5 frame input/output (c) 7 frame input/output

Figure 3.5: Detection analysis [128] for 3/5/7 frame detectors, evaluated in Table 3.8. These results were generated on samples of lengths 3/5/7 frames respectively from ImageNet Vid validation set. Each video was sampled with a stride of 2 frames which gave a total of 85K samples.

tector not activating on objects). This performance is also similar to the middle frame prediction models (see Fig. 3.3). But when we look at the mass of 'BG' it shows that the model confuse positive object classes to the background, i.e. the number false negatives increase even more dramatically than in the case of middle frame prediction models. Moreover, the error mass associated to localization errors also increases with increasing input size, as it also apparent from the IoU metric in Table 3.7. Interestingly, the 3 frame model's classification layer seems to be performing better than the single frame baseline.

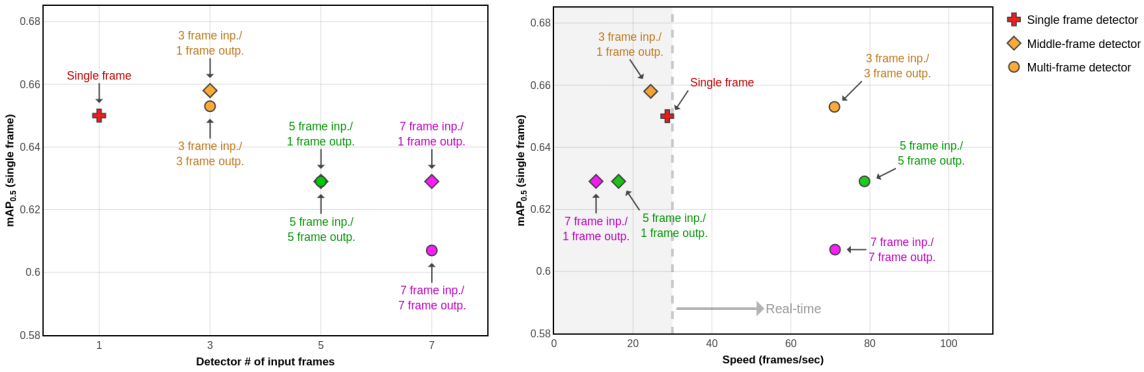
Fig. 3.6 shows the decrease in +ve objectness accuracy over validation samples as each model is trained. Only the 3 frame model works on par with the single frame baseline. Both 5 and 7 frame input models have a decrease in their ability to separate foreground objects from the background.

We compare all the different multi-frame models in this experiment, as well as §3.8.5, with the single frame baseline. Fig. 3.7 shows how the single frame $\text{mAP}_{0.5}$ changes with models trained with different number of input frames. It is clear that after a gradual rise for a 3 frame input, there is a steady downward trend for detector performance for both middle and multi-frame prediction. Interestingly, both 3 and 5 frame input multi-frame detectors manages to match the middle-frame detector counterpart. If you look at the performance



(a) +ve objectness accuracy during training (b) +ve objectness capture accuracy during training

Figure 3.6: These two graphs show the validation +ve objectness accuracy computed every time after training on 15K samples. Both graphs show validation accuracy for all the models in Table 3.8. The models were trained with multi-frame input, and predict detections spanning all input frames. For details see Fig. 3.4.



(a) Detector accuracy vs. # input frames (b) Detector accuracy vs. speed

Figure 3.7: These two graphs show how the accuracy of different models in Table 3.9 changes against the number of frames each model is trained for, or against the maximum number of frames a model can process in a second on a single GPU.

against the speed of each model, the 3 frame multi-frame detector provides a nice performance to speed trade-off. Its mAP performance is only a few points lower than the best model (3 frame middle-frame detector), but is about 3 times faster when run in batch mode over a single GPU. Table 3.9 collates all the performance metrics in a single table.

Table 3.9: This compares the methods in Table 3.7 and Table 3.8 from the perspective of both speed and accuracy. "Frames / sec / GPU" is the speed of each method when running on a single GPU (Maxwell Titan X - 12G) wit the maximum batch size possible for each method (the batch size is given under the column "Full GPU Batch Size"). To get a better sense of the speed of each model in an on-line setting, we limit inference to a single sample per batch - supposing a single stream of frames is available to the algorithm.

Method	# Parameters	# 3D Conv. Ops.	IoU	Classification Accuracy	+ve Objectness Accuracy	Single Frame $mAP_{.5}$	Frames / sec / GPU	Full GPU Batch Size	Frames / sec / GPU Batch Size 1	GPU Memory Footprint
Single frame	28.3M	0	**.807**	.763	.444	.650	28.69	24	12.49	**0.89G**
3 frame inp./1 frame outp.	28.4M	2	**.807**	**.768**	.445	**.658**	24.48	23	10.37	1.13G
3 frame inp./3 frame outp.	28.8M	2	.794	**.771**	**.459**	.653	71.04	23	28.50	1.13G
5 frame inp./1 frame outp.	29.1M	4	.803	.753	.423	.629	16.35	14	8.02	1.46G
5 frame inp./5 frame outp.	29.7M	4	.784	.758	.435	.629	**78.65**	14	**37.95**	1.47G
7 frame inp./1 frame outp.	31.7M	6	.800	.761	.411	.629	10.63	9	5.71	1.89G
7 frame inp./7 frame outp.	32.7M	6	.772	.749	.403	.607	71.19	9	35.98	1.89G

3.8.7 Detection + Tracking Results

This section uses the video detection models from §3.8.5 and §3.8.6, and gives results for a complete detection + tracking pipeline over the length of each validation video in ImageNet Vid dataset. Here we would define *window* as a set of frames over which one of the detection models can produce video detections over all the frames. For instance, for single frame models (given by a single frame baseline model or middle frame multi-frame model §3.8.5), all windows would be of 1 frame long. All detection models which produce results over multiple frames (§3.8.6) would use a window of T frames long. Detection + tracking results are generated in this section by producing detections for each non-overlapping window, and then stitching detection results over consecutive time windows using simple bipartite matching. For instance 7 frame input, 7 frame output detection model would be used to generate video detection results for windows frames 1-7, then 8-14, and so on. The process starts with creating *tracks* for all the detections in the first time window. When the process produces detections for the second time window, we attempt to

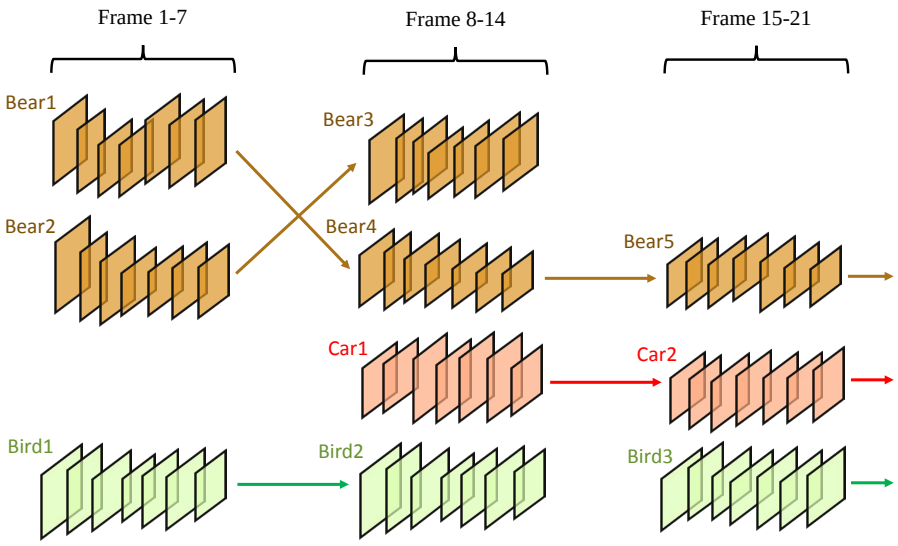
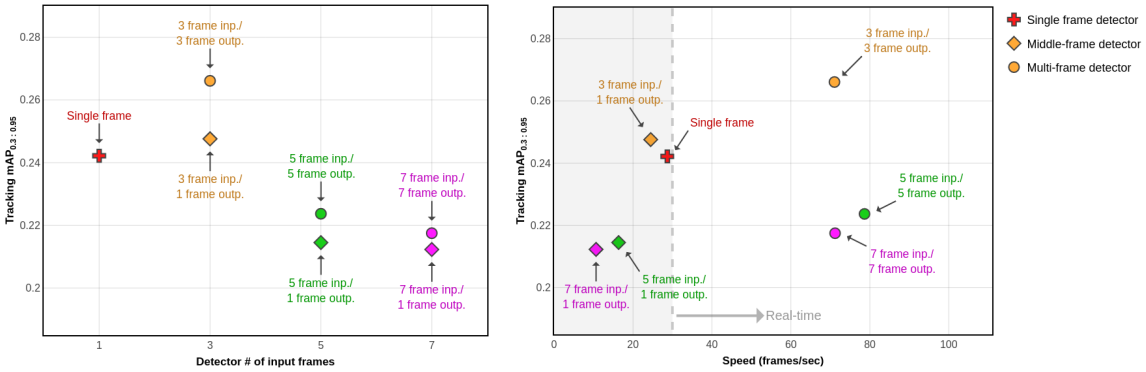


Figure 3.8: This example shows how tracks are generated from video object detections over different time windows. The video detection model produces detections over 7 frames. Note that since the model predicts object detections, each comes with object class predictions. During tracking, our process only matches tracks to detections if they belong to the same object category, i.e. a 'bear' track would not get matched to a 'bird' detection. The association between tracks and detections is done using bipartite matching where the cost of matching is 1-bounding box overlap. This process naturally creates new tracks (when a detection is not matched to a track - like 'Bird3'), and ends tracks (when a track is not matched to any detection - like 'Bear3') when necessary.

associate each detection to the current set of tracks. All tracks that do not get any detections die. All detections that are not associated to any track, create new tracks. The process continues till the end of the video.

The association between a track and the detection is done by computing the overlap between the bounding box in the last frame of the track and the first frame of the detection in the subsequent time window. To help get more accurate overlap scores, we use a Kalman Filter to predict the next bounding box for each track, which is used instead of the actual bounding box to compute overlaps with detections in the next time window. As illustrated in Fig. 3.8, matching between tracks and detections only takes place when both of them belong to the same object category.

During tracking we note that at times the detector would not produce a detection for a certain object. To avoid the case of loosing track of objects intermittently, we allow the



(a) Detection+tracking accuracy vs. # input frames

(b) Detection+tracking accuracy vs. speed

Figure 3.9: These two graphs show how the video object detection+tracking accuracy of different models in Table 3.10 changes against the number of frames each model is trained for, or against the maximum number of frames a model can process in a second on a single GPU. If the goal is to detect and track objects, these results give a sense of the number of frames a model should be trained for to get the best accuracy against inference speed.

track of an object to continue in absence of a detections on two consecutive time windows, i.e. a track is ended only if it is not associated with a detection over two successive time windows. The Kalman filter predicts the state of the bounding box to get better matches to any subsequent detections. Each track is only kept if the average confidence level predicted by the detector is above a certain threshold. This threshold is tuned empirically for each detection model to get the best accuracy. As a post processing step, all tracks are removed which have a length of less than 30 frames.

We measure the tracking accuracy in two ways in this section. We compute a *tracking mAP* score which is similar to the detection mAP - the difference between the two is in computing the overlap between the prediction and ground-truth. In detection mAP, as shown in §3.7.2 the overlap is computed between the detection prediction and the ground-truth which are both limited to the size of the time window T. If a ground-truth object does not overlap the current time window, it is not considered for a matching. Whereas, for computing a tracking mAP, all predictions over all the video frames are considered for matching to each ground-truth track. The volumetric IoU between the two is computed considering all the frames that both exist in. Note that a detection which predicts extra bounding boxes

Table 3.10: This compares the methods in Table 3.9 from the perspective of both speed and detection+tracking accuracy. The process to obtain full video object tracks across the length of the video from a set of detections is explained in §3.8.7. The 'Tracking mAP' metric is similar to a detection mAP metric (described in §3.7.2), except that the score computed between tracks predictions and GT tracks is the volumetric IoU over all the frames the prediction/GT exists. This allows us to naturally extend detection mAP to measure the localization/classification/tracking ability of an algorithm. Tracking $mAP_{.3:.95}$ is the mAP averaged over 14 different overlap thresholds. 'Recall' measures the number of true-positives from the 1,309 GT tracks in the 555 videos in ImageNet Vid validation set. The average track length of GT objects is 208.9 frames (~ 8 secs - see Table 3.1).

Method	# Parameters	# 3D Conv. Ops.	Single Frame $mAP_{.5}$	Frames / sec / GPU	Tracking $mAP_{.3}$	$Recall_{.3}$	Tracking $mAP_{.5}$	$Recall_{.5}$	Tracking $mAP_{.7}$	$Recall_{.7}$	Tracking $mAP_{.3:.95}$
Single frame	28.3M	0	.650	28.69	.435	.566	.324	.438	.223	.304	.242
3 frame inp./1 frame outp.	28.4M	2	**.658**	24.48	.446	.572	.329	.442	.231	**.309**	.248
3 frame inp./3 frame outp.	28.8M	2	.653	71.04	**.478**	**.584**	**.363**	**.448**	**.242**	.290	**.266**
5 frame inp./1 frame outp.	29.1M	4	.629	16.35	.427	.547	.293	.402	.180	.265	.215
5 frame inp./5 frame outp.	29.7M	4	.629	**78.65**	.445	.568	.298	.402	.193	.248	.224
7 frame inp./1 frame outp.	31.7M	6	.629	10.63	.415	.513	.289	.380	.185	.253	.212
7 frame inp./7 frame outp.	32.7M	6	.607	71.19	.423	.519	.301	.363	.197	.221	.218

in frames where the ground-truth object track doesn't exist would get heavily penalized for this spurious localization in additional frames. Once the overlap between all object detections and ground-truths has been computed, the computation of mAP follows as done in §3.7.2. This gives the tracking mAP metric - which can be computed at different overlap thresholds. We also compute the COCO style [125] mAP combined metric which in our case is averaged over overlap threshold 0.3 to 0.95, at 0.05 intervals. Given the overlap between predicted tracks the ground-truths, we also compute the recall percentage at different overlap thresholds.

Fig. 3.9 compares quantitative tracking results for all of the 6 video detection models used in the previous sections as well as the single frame detection baseline. It is important to note that unlike the single frame detection performance in Fig. 3.7, the tracking mAP accuracy of the multi-frame detectors is better than the middle-frame detector. This is largely

because the correspondence decisions made by the network are better than the bipartite matching across time windows. Also, it is worth noting that the 3 frame input/3 frame output model gives 10% better tracking performance than the single frame baseline, while being 2.5 times faster. Table 3.10 quantitatively shows the same tracking mAP metrics for all different models, as well as the recall scores at the overlap thresholds of 0.3, 0.5, and 0.7. The 3 frame input/ 3 frame output is a clear winner over most of these metrics.

3.9 Conclusion

In this chapter we explored a fully convolutional pipeline for the task of video object detection, which is trained to accept multiple frame inputs, and predict detections spanning over all the input frames. This network naturally produces detections which can be seen as tubelets, where the network learns to do correspondence over a short set of frames. This can be seen as an end-to-end learnable pipeline to generate tracklets over a short number of frames. We also propose a simple scheme to generate long term tracks from these detections by simple bipartite matching.

A conclusion we can draw from our experiments is that using more frames in 3D convolutional pipeline to detect objects is not guaranteed to improve the performance. We observe that a 3 frame model performs marginally better than a single frame detector, but then the performance gradually drops as models are trained with more input frames. A large part of these errors can be attributed to the model's inability to distinguish objects from background when the number of input frames increases. Part of the problem lies in the scale of ImageNet Vid dataset and the lack of diversity of videos. This leads to models being more discriminative if trained with an additional dataset.

We conclude that for performing detection in videos, it is beneficial to construct a 3D convolutional network and train it to produce predictions in form of short 3 frame tracklets. In addition marginal gain in mAP, one can expect the method to be 2.5 times faster than

traditional single frame detectors.

Chapter 4

Self-Directed Incremental Learning

Children are amazing learning machines. In the domain of word learning, for example, children acquire an average of 8 to 10 new words per day and reach a vocabulary of 60,000 words by adulthood [129]. Children accomplish much of their learning through self-directed play, and a remarkable property of their achievement is the minimal amount of explicit supervision they receive. An infant's play is *self-directed* in the sense that they pick up, examine, and put down toys of their own volition, and through this process they acquire extensive object knowledge. In contrast to the volume of their self-directed perceptual inputs, the moments in which a supervisory signal is available, for example when an adult names an object, are extremely rare. This stands in stark contrast to the dominant object recognition paradigm in computer vision, in which datasets of images are extensively annotated with object labels, requiring substantial labor and cost. In reaction to this reality, many recent works have developed partially—or fully—unsupervised learning methods meant to relieve the burden of annotation [130, 131, 132]. The goal of this chapter is to add to this dialog by framing self-directed play as an alternative paradigm for object learning, and by providing the necessary computational infrastructure and evaluation methodology to tackle it experimentally.

In addition to its self-directed nature, infant learning is inherently *incremental*. During

play, for example, an infant might pick up a toy duck and rotate it in depth while watching it. The resulting sequence of frames contains clues to the shape, appearance, and identity of the object. After losing interest, she might put down the duck and pick up a toy ball, subjecting it to a similar pattern of visual interrogation. Once the duck has been put down, however, its imagery is no longer available for learning. In contrast, modern deep learning architectures utilize a minibatch approach in which a random sample of frames that cover a significant subset of the label space are processed simultaneously. This ensures that gradient updates do not favor one object over another in moving collectively towards higher accuracy. However, if data is fed incrementally to standard deep learners the result is *catastrophic forgetting*, in which object representations developed early in training are forgotten at the expense of more recent examples [133, 134]. Recent incremental learning works have developed methods based on distillation loss [135] and exemplars [136] to address the catastrophic forgetting problem. However, these works assume that a ground truth label is provided for every frame. In contrast, children are exploiting an *inherently weaker* form of supervision during self-directed play. While the infant is examining the duck, she knows that it remains the same object during the entire time she is holding it, but she does not know its label. We refer to a consecutive "chunk" of frames containing the same object as a *learning exposure*. Figure 4.1 illustrates the scenario. Learning exposures contain single objects and are delineated by pick up and put down events whose occurrence times are known. When the infant picks up an object, she has to infer whether this is an object she has seen before (in which case she is receiving a repeated learning exposure from a familiar object, as in the case of the second exposure to object 7 in Figure 4.1), or whether it is a new object she is seeing for the first time. This inference must be performed without explicit supervision, as naming events occur too sparsely in time to be useful. This observation leads us to define a novel problem of *self-directed incremental learning*: given a sequence of learning exposures, the learner must incrementally construct object models, and must solve the problem of correctly associating learning exposures with objects in the

case of repeated exposures (when the same object occurs more than once).

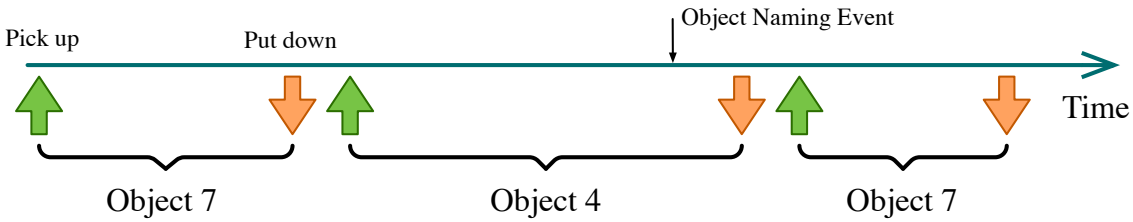


Figure 4.1: One of our main assumptions of child play—their visual experiences primarily consist of holding and examining objects. The labelling children receive is both very small and very sparse relative to the amount of visual information.

From the perspective of self-directed learning, an infant's first exposure to the world of objects comes through the repeated manipulation of a relatively small set of object *instances* in the form of toys, cups and other dishware, and assorted household objects [137]. These objects are often colorful, and their shapes can vary from simple to complex, including both articulated (e.g. ring of plastic keys or dumptruck) and deformable (e.g. various stuffed toys) objects [138]. We therefore focus in this chapter on the problem of recognizing object instances, and defer to future work the construction of category boundaries based on naming events and other cues. A key requirement for self-directed learning is a means for generating a sequence of learning exposures. While modern deep learners typically use randomly-selected subsets of unrelated images, we believe there is value in reproducing the smooth changes in viewpoint and zoom generated by infants. For example, there is evidence that infants are sensitive to motion cues and may use them in developing object models [139]. We choose to generate learning exposures through computer graphics rendering. Synthesized images are playing an increasingly important role in deep learning [140, 119, 141, 142, 143]. In contrast to finite datasets of real-world images, a computer graphics approach affords the generation of an unlimited number of object images with full control over viewpoint, zoom, and lighting changes. This approach allows us to address the problem of long-term learning with repeated exposures to objects that characterizes infancy. We have created a system for data generation called CRIB—Continual

Recognition Inspired by Babies, which can simulate the types of image sequences that infants produce during self-directed play. See examples in Figures 4.2 and 4.3. In contrast, recent works on incremental learning [136, 135, 144, 145] assume that every frame has a category label and work with existing datasets, such as MNIST [146] and CIFAR [147], which they organize to be processed incrementally. The goal of this chapter is to introduce a novel, developmentally-inspired incremental learning problem, and a challenging, easy-to-use system for data generation which can help to drive the research community in advancing this important topic.

This chapter makes the following contributions:

1. We introduce a new and developmentally plausible incremental learning paradigm, self-directed incremental learning, and evaluate the performance of incremental learning algorithms in this setting.

2. We perform extensive evaluation of the effects of catastrophic forgetting on incremental learning algorithms over hundreds of unique learning exposures.

3. Complementing previous work in incremental learning, we take a step back from the close focus on catastrophic forgetting and investigate the effect of repetition of previously learned concepts on recognition accuracy.

4. We develop the CRIB data generating system that uses a curated set of 200 3D models of toy-like objects.

4.1 Related Work

Most closely related to the current effort is the body of work in **incremental training** of deep models focusing on the problem of catastrophic forgetting, the main problem in the way of successful incremental learning algorithms. Authors have taken multiple approaches to incremental training of neural networks. Most relevant approaches exposes

learners to two or more categories at a time from a single fixed dataset [148, 136]. Alternatively, the problem is treated from a transfer learning perspective, initially training on one dataset then evaluating catastrophic forgetting when training on new data from either entire new datasets or parts of datasets [135, 144, 145]. Although previous work has had a significant effect of setting in motion the study of incremental learning, upon close review of these works certain straightforward questions arise that we investigate in our work: Are there possible ways to move away from the current assumption of using completely labelled data towards a less supervised paradigm? Abandoning full supervision is the most significant aspect of our self-directed incremental learning problem. To the best of our knowledge, no previous incremental learning work has investigated this topic. What are the effects of repeated exposure on catastrophic forgetting; can it mitigate forgetting in incremental algorithms? We investigate this by evaluating recognition accuracy of incremental learning algorithms on unique objects from our dataset, including a scenario with repeated exposures of each object. What is the effect of using pre-trained models for incremental learning rather than randomly initialized models? We empirically address this question in § 4.5.

Datasets of objects have been created in the past and fall into two main categories: image datasets of real objects, and of synthetic 3D models of objects. Known examples of datasets of fixed images datasets are NORB [146] and COIL [149], and more recently, CORe50 [150]—containing temporally continuous RGB-D videos of objects. Previous works on synthetic 3D object categories such as ShapeNet [141] and instances such as Sculptures [132] focus mainly on using 3D models to study aspects of 3D shape, leading to significant progress in the domain [151]. Distinguishing the CRIBdataset from this previous work is the variety among the objects, the data generation API and the fact that all objects are toy-like. The specific distinctions are outlined in Table 1.

Computer-graphics based methods, have been used to generate large amounts of train-

[1]CORe50's annotation is coarse; bounding boxes are not tight around the object as it is rotating.

Table 4.1: Characteristics of different datasets of objects that may be used for incremental learning compared to CRIB. Below the horizontal line are characteristics especially relevant to developmentally inspired incremental learning.

	COIL [149]	NORB [146]	CORe50 [150]	ShapeNet [141]	Sculptures [132]	CRIB
Category/Instance	I	C	C	C	I	I
Synthetic	✗	✗	✗	✓	✓	✓
Pose Labelling	✓	✓	✗	✓	✗	✓
Unlimited View Granularity	✗	✗	N/A	✗	✗	✓
Data Generation API	✗	✗	✗	✗	✗	✓
Temporally Continuous	✗	✗	✓	✗	✗	✓
Exposure Tracking	✗	✗	✗	✗	✗	✓
Bounding box annotation	✗	✗	✓[1]	✗	✗	✓

ing data for deep learning models. Examples include purpose built autonomous driving simulators such as TORCS [152] and CARLA [153]. For semantic segmentation, Koltun et al. [140] successfully showed that pixelwise labelled semantic segmentation data can be generated from an open-world commercial video game, and easily annotated without interaction with source code. For viewpoint estimation, Sheikh et al. [142] used 3D car assets to generate 800,000 images. These works have started to address the domain shift problem, with Koltun et al. showing comparable performance to models trained solely on real world data by augmenting the synthetic data with only 30% real world data. Sheikh et al. were within a small margin of error when tested on real data and got improved testing performance when augmenting real data with their synthetic data. These results indicate the potential impact of using synthetic data when obtaining and annotating real world data is difficult. Similar to these works, because of the synthetic nature our data, a domain shift exists between using CRIB and using real world data. In §4.5 we empirically show that CRIB data is still visually challenging for incremental learning algorithms.

There has been a long-standing interest in developmentally inspired approaches to robotics and learning [154]. Works, such as Gepperth et al. [155] and Kanan et al. [156], share our motivation for **biologically inspired** incremental learning. But, our work in this chapter is uniquely motivated by developmental psychology, and focuses on the evaluation of incremental learning algorithms in greater time horizons—and only allows for one

object in each learning exposure.

4.2 Approach

Our approach to the problem of self-directed incremental learning has three main elements. First, we develop a *data generator*, a computer graphics simulator which can generate an unlimited amount of the types of image sequences that children routinely produce and analyze during object play in early infancy. Second, we formally define the self-directed learning problem and present a solution which leverages existing approaches to incremental learning, and present the first empirical results for this challenging task. Third, we conduct a series of experiments that highlight key aspects of self-directed learning and its relationship to incremental learning methods based on deep networks. Our goal is to both motivate and enable the computer vision community to tackle the self-directed learning problem, which we will do by releasing our data generator and all supporting software on our project website.

A key element in our approach is the *data generator*, a computer graphics simulator which takes the place of the data loader in a deep network implementation and automatically generates image sequences of objects based on a model of the object exploration behaviors produced in early infancy. An example of the data generator output is illustrated in Figure 4.3. Our decision to utilize rendered images of toy objects as our primary data source is based on three considerations.

First, research has shown that infant's play consists of repeated bouts of object examination in which a toy is picked up, rotated (frequently in depth) and possibly mouthed, and then released. Infants can spend several hours per day in such object exploration activities, over the course of many months. In order to train models on the image sequences yielded by such play behavior, we need the ability to create images of objects over arbitrary rotation sequences and repeatedly generate a large number of such sequences for a

single object. There is consensus that kinematic information, that is is visual information provided through the motion of the object or the observer, is the basis for the perception of form [157]. A relevant study [139] suggests that four month old infants are able to distinguish between old and novel objects more so when the objects are undergoing rotation along an axis than when the object views are static.

Therefore we need to recreate the scenario in which an object might be picked up and manipulated multiple times within a single play session or in a multitude of sessions. First person data from play sessions has been gathered in developmental psychology studies [138]. However, in addition to being very small and expensive to obtain, this data is low resolution and with artifacts like overexposure and motion blur, resulting in the inaccurate capture of the visual stimuli infants experience, leading to our second consideration. Rendering enables the generation of an infinite quantity of high quality image sequences under varying illumination and provides fine-grained control of rotation and zoom, making it an extremely attractive solution for this scenario. In contrast, sequences of real-world object images captured with a turntable would be rapidly exhausted after only a few bouts of play, unless they were prohibitively large. Last, our decision is supported by the recent increase in interest and successful use of computer graphics based approaches to tackle problems in 3D shape recognition [132], semantic segmentation [140] and viewpoint estimation [142].

4.3 The CRIB Data Generating System

Having a diverse set of toy-like objects is central to building CRIB. We collected 200 unique toy object models from the freely available library of Blendswap [158] where they are available under CC licenses. The models we obtained are generally very distinct, with significant visual differences between some, and slight similarities between others. A representative set of these objects is shown in Figure 4.2. Our dataset includes synthetic replicas of 30 specific objects instances used in research studies with infants [138].

Figure 4.2: A rendering of approximately one third of the 3D models used by the CRIB data generator. These toy-like objects were curated from Blendswap, and adjusted in a way to appear visually diverse in shape and color.

CRIB is implemented in the open source and cross platform 3D graphics software Blender. The built-in Cycles rendering engine is used for the rendering part of the data generator. Blender shaders are composite functions indicating the texture and color characteristics of the data. Specific shaders were developed to give the objects used by CRIBa more toy-like, plastic appearance. These characteristics are evident in Figure 4.2. We implement additional functionality in CRIB that allows for precise bounding box generation and tracking of the amount of object surface shown in each rendered frame.

4.3.1 Incremental Learning Data Unit

For each learning exposure, the CRIB data generator creates a *data unit*—a temporally continuous sequence of data. We encourage the reader to refer to the left part of Figure 4.3 to follow the basic description of how a data unit is generated. For an object model, a lighting setting is chosen, and then the object rotated around its approximated center of mass, generating different views of that object along its viewing sphere. Different views are rendered by a camera placed above the object; resulting frames are depicted in the top row of Figure 4.3.

Following is a detailed description of the data generation process. Generating a data unit starts by rendering object frames with a zero alpha channel other than the object itself. To

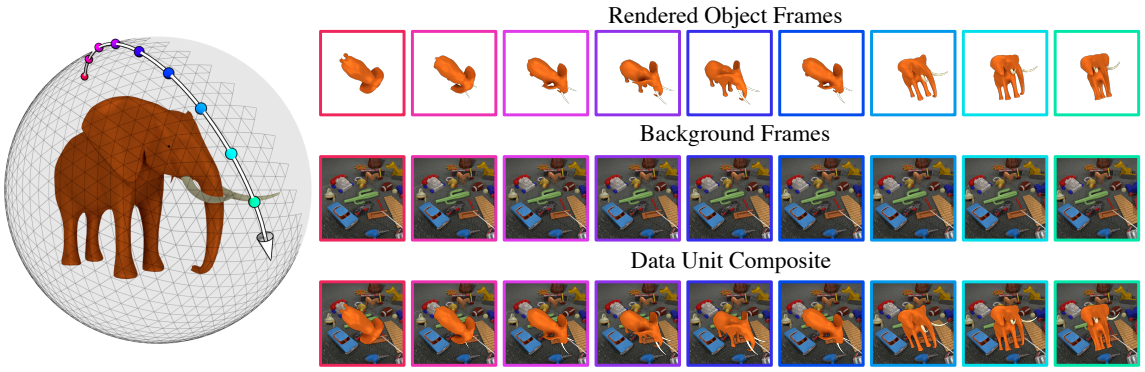


Figure 4.3: Illustrating the three steps of data generation: 1. object rendering, 2. background selection, 3. foreground and background compositing.

render object frames, a lighting setting of either four point or three rod light sources placed above the object is chosen. Each light is defined by a tuple (location, rotation, temperature, strength). Location and rotation are slightly jittered, random values within a range that produce a favorable visual result are chosen for strength, and within 4000-6000K (indoor lighting range) for temperature. To generate the rotation of the objects, first a user specified n random $(x, y, z, s) = (\text{azimuth}, \text{rotation}, \text{elevation}, \text{scale})$ points are generated, where s is within a specified range of the original object scale. Temporally continuous rotations are generated by linearly interpolating between the n points in arbitrary granularity, meaning that the position of the object can change by a lot or by a little between every two frames. The second step of generating a data unit consists of selecting a background image to composite the object on top of. There are 8 groups of background scenes consisting of 25 objects rendered laid out on a floor in a cluttered manner. These scenes are used to generate sequences of images with the camera above the objects moving left/right and then up/down over time to loosely emulate head motion, as evident in the second row of Figure 4.3. Our aim in doing this is to include a dynamic background environment, cluttered with different objects from the one in focus. We ensure that the objects in the background are distinct by choosing images from scenes where the object is absent. Finally, the background and foreground objects are composited, and a very small amount of pixel wise noise is added.

Testing Data:

The CRIB data generator is also required to generate testing data for object recognition. To this end, the data generator provides a set of images of the target object where each image has the object in rendered random (x, y, z, s). Each rendered image has a random lighting assignment, with the same ranges possible as when generating data units. Our goal in providing testing samples in this manner is to test the learning algorithm's ability to recognize the object in a manner invariant to a subset of the changes in infants' learning environments.

4.4 Self-Directed Incremental Learning

When trained incrementally, in every learning exposure learning algorithms receive labelled training data that is only available temporarily. The most important distinction of the self-directed problem is that no labelling information is provided other than indicating the start and end of a learning exposure. Allowing this kind minimum supervision is in line with our developmental motivations, specifically the fact that children are inherently aware of the start and end of the visual learning stimuli they receive. An additional important aspect is that there is no finite limit beforehand on the total number of learning exposures, in line with the vast amount of available to infants for learning. Our data generator allows for this unconstrained temporal dimension to learning.

Having briefly revisited assumptions with regard to learning during self-directed play in children in the introduction, we proceed to formally define the self-directed incremental learning problem.

At each learning exposure $k \geq 1$, the learning algorithm receives a new data unit from CRIB. CRIB provides no labelling information with the sequence of images, and the data unit may be of an object seen at learning exposure i such that $1 \leq i \leq k$. The algorithm is first required to successfully make a decision whether the current data unit is of a novel

object or of an object that has been previously experienced. Once that step is complete, if the decision is that a novel object is present, the algorithm's parameters get updated accordingly. If the decision is that an old object is present, an additional decision is required to correctly relate one the previously seen objects with the object in the current learning exposure. If this decision is made correctly, the algorithm can perform a parameter update, reinforcing its knowledge based on this repeated exposure.

The first evaluation metric of interest is object instance recognition accuracy, which is measured at the end of each learning exposure, in a way adapted from [136]. Let S_k denote the set of unique objects seen until instance k without regard to repetition. Let $a_k^{(i)}$ indicate the testing accuracy of an object $i \in S_k$. Object recognition accuracy after instance k is defined as

$$A_k = \frac{1}{|S_k|} \sum_{p \in S_k} a_k^{(p)}. \tag{4.1}$$

The reader can refer to Figure 4.4a for a visual example of how incremental accuracy evolves over multiple learning exposures.

Another evaluation criterion is required to keep track of the number of errors made in recognition of objects as old or new. High performance at this task is essential, since it is a crucial step towards achieving good testing accuracy. High testing accuracy numbers are impossible if the algorithm does not accurately track the unique objects it has seen. We define three kinds of errors that can occur in this task and keep track of them as learning progresses:

- Type 1: When exposed to a previously seen object, deciding it is new.
- Type 2: When exposed to a new object, identifying it as old.
- Type 3: When correctly deciding that an object is old, but failing to correctly relate which old object is in the current learning exposure.

4.5 Experiments

To clearly describe our baseline solution and position the self-directed incremental learning problem and the CRIB data generator relative to previous work, we first briefly refer to previous incremental learning solutions.

The Learning without Forgetting (LWF) [135] approach describes a strategy around a standard convolutional network to address catastrophic forgetting for image classification. First, for all new classes introduced in a new learning exposure, the fully connected layer of the CNN is expanded by adding output sigmoid units. Second, distillation loss is applied in to the old output units, keeping the network's parameters from significantly changing because of gradient backpropagation from new data.

The incremental Classifier Representation Learning (iCaRL) [136] approach builds upon LwF by including explicit memory in the form of an exemplar set that is appropriately managed as the algorithm encounters more learning exposures. iCaRL uses this exemplar set to perform nearest exemplar mean classification in feature space. Briefly put, the inference procedure consists of computing normalized exemplar mean features per class using the underlying CNN up to the fully connected layer as a feature map, and then classifying by determining the nearest exemplar mean from the normalized feature of each testing sample. Interestingly, training signal for backpropagation comes from the fully connected layer, that still gets trained as in LwF.

We evaluate reference performance using CRIB on the following existing incremental solutions, including slightly modified versions:

1. **LwF** - The standard Learning without Forgetting algorithm with distillation loss to account for forgetting, but with binary cross-entropy loss for classification.

2. **iCaRL-S-D** - The reference iCaRL algorithm as described in [136] and in the beginning of this section.

3. **iCaRL-S-ND** - The reference iCaRL algortithm but without distillation loss. Em-

pirically we found that when varying the number of different classes per learning instance, using only exemplars stored with ground truth labels rather than distillation improved accuracy.

4. **iCaRL-PT-ND** - The reference iCaRL algorithm, starting from a ResNet-34 [159] model pre-trained on iLSVRC-2014 classification dataset [160], trained without distillation loss.

4.5.1 Baseline Solution for Self-Directed Incremental Learning

As a baseline solution we implement two modifications to adapt iCaRL to the self-directed incremental learning problem. First, since iCaRL's exemplar management scheme operates under the assumption that every new learning exposure contains data for new concepts, modifications are required for repeated exposure. We modify the iCaRL algorithm to appropriately manage the exemplar set when exposed to previous concepts. See supplemental material for details.

Second, the iCaRL algorithm lacks a mechanism for learning without complete supervision. The algorithm operates under the assumption that in every new learning instance a new concept will be introduced, and performs the specific architectural and explicit memory changes. In iCaRL specifically, a decision is required as to whether to add a new output sigmoid unit—the case when a new concept is introduced, or to find which of the old sigmoid units in the fully connected layer corresponds to the concept in the current learning exposure and backward propagate gradients to train that unit—the case when an old concept is repeated. We introduce two ways in which iCaRL can make this decision using the incoming training data using a thresholding scheme:

- Using sigmoid unit values—first compute the mean sigmoid activation value for all images in the current data unit. Choose the sigmoid with the maximum mean output, and if it is above a certain threshold, infer that the current data is a repeated exposure.

If below, the data unit is of a new object.

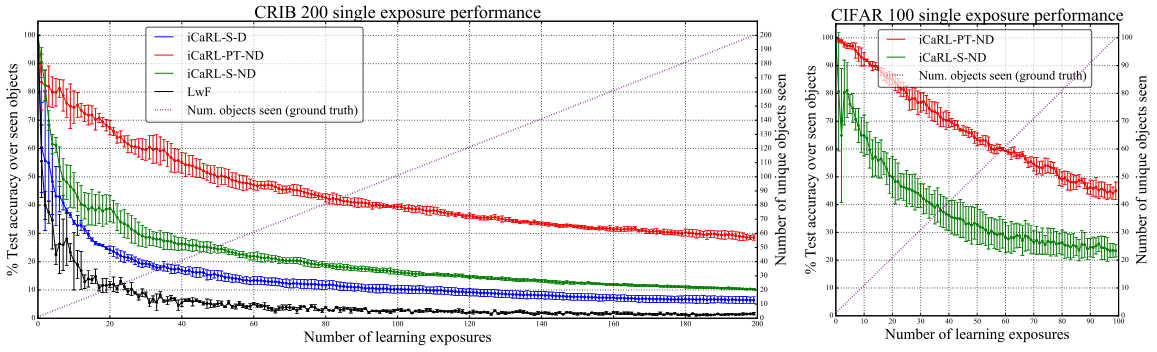
- Using exemplar means—for all images in the current data unit, collect normalized features using the penultimate layer of the network for all exemplar data, and compute an exemplar mean over those features for each learned concept as in [136]. Similarly collect normalized features for all train images in a data unit and compute the euclidean distances for each of these features from each of the exemplar means. Then compute the mean of these distances over all images in the data unit and pick the minimum mean distance value. If the value is below a certain threshold, infer that the new data unit is a repeated exposure for an object corresponding to the object instance whose exemplar mean was the closest, otherwise infer that the data unit contains a novel object.

Thresholds used in both schemes were found as the optimal operating point from a Precision-Recall analysis. The analysis was done on a binary classification problem of identifying incoming data unit as an object that was previously encountered (positive class) or a novel object (negative class) in a fully supervised setting with repeated exposure to previously seen object instances.

4.5.2 Evaluation and Analysis

We perform three different incremental learning experiments using CRIB. In the first setting, fully supervised single exposure we expose the learner to each object instance once—an experimental setting similar to past work [136]. Our second experiment, fully supervised repeated exposure, gives learners an object multiple times during learning, with the purpose of investigating catastrophic forgetting in presence of repetition. Finally, we evaluate the baseline performance of our solution to the self-directed incremental learning problem.

The algorithms used for establishing baseline performance on each incremental learning paradigm are iCaRL, iCaRL variants, and LwF where applicable. Both iCaRL and LwF



(a) (b)

Figure 4.4: (a) Performance of LwF and three variants of iCaRL when they are presented with a single exposure for each object instance from CRIB200. Accuracy at each learning instance is calculated using Eq. 4.1. The standard-deviation bars were computed over 3 runs for every experiment (except iCaRL-PT-ND on CIFAR 100, which is on 2 runs)—each with different random orderings of objects. (b) shows similar trends when using CIFAR-100 [147] for the two best variants of iCaRL.

and the self-directed incremental learning solution are implemented in PyTorch [161], using a ResNet-34 [159] as the backbone architecture. Specific hyper-parameters were tuned for each algorithm; the details are included in the supplementary materials. Learning exposures consist of data units that are 100 frames long and are generated by interpolating between three random points on the viewing sphere of an object, with scale smoothly varying from 0.3 to 1.1 of the original scale of the object. Testing is done on 100 frames of random object views, scale and lighting for each object that has been seen previously. This testing is performed after each learning instance. All classifiers are trained and tested on positive bounding box patches. During training random patches outside of the positive patch are included in the training data as negative samples to allow for discriminative learning.

Fully Supervised Single Exposure:

In this setting algorithms are trained on 200 learning exposures, each being one data unit of a unique object. iCaRL and its variations are allowed a memory of 2000 exemplars, which equals 10% of the total data that will be made available to the algorithm. In [136] when evaluating on CIFAR, the authors allow for 4% of the total data, meaning that our

experiment provides more favorable conditions in terms of explicit memory allowed.

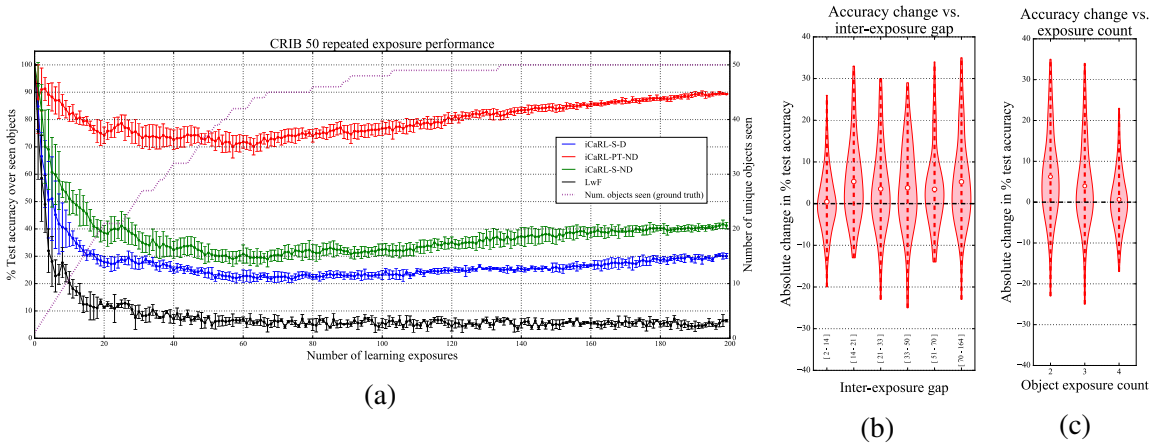
The results of this experiment are plotted in Figure 4.4a. All methods have a general downward trend, which is similar to observations made on other datasets [136, 135]. This trend can be partially attributed to catastrophic forgetting of classes which were seen early on in the experiment, as well as the algorithm having to absorb an increasingly large number of concepts over time. Results for the variants of iCaRL [136] show that exemplars can be effectively collected from data units produced by CRIB. This allows iCaRL-S-D's final accuracy to be 4 times better than LwF. The results for iCaRL-S-ND show that the use of distillation is not as effective as directly training with exemplar labels. Furthermore, the experiment with iCaRL-PT-ND shows that test accuracy can be easily improved by using a pre-trained model[2]. This finding is in line with other areas of computer vision, where model pre-training plays a key role in improving baseline performance of algorithms. To verify our findings, and confirm the difficulty of data generated by CRIB we repeat this experiment with CIFAR-100 [147] with iCaRL-S-ND and iCaRL-PT-ND (see Figure 4.4b).

Fully Supervised Repeated Exposure:

In this section, we evaluate whether the four learning algorithms can exploit repeated exposures to previously experienced objects. With this goal in mind, all algorithms are shown 50 unique objects over 200 data units, with each object appearing four times. For every experiment run, we generate a random sequence of object instances such that all methods see the same number of objects by any learning exposure. As mentioned in §4.5.1, the variants of iCaRL were modified to function in this experimental setting.

Figure 4.5a shows the results for each method in this experimental setting. All learning algorithms, except LwF, demonstrate some ability of using repeated exposures to improve the classifier's performance over time. The test accuracy of every iCaRL variant starts consistently improving after seeing 45 different objects over 70 learning exposures. Inter-

[2]ResNet-34 [159] model pre-trained on iLSVRC-2014 classification dataset [160]



(a) (b) (c)

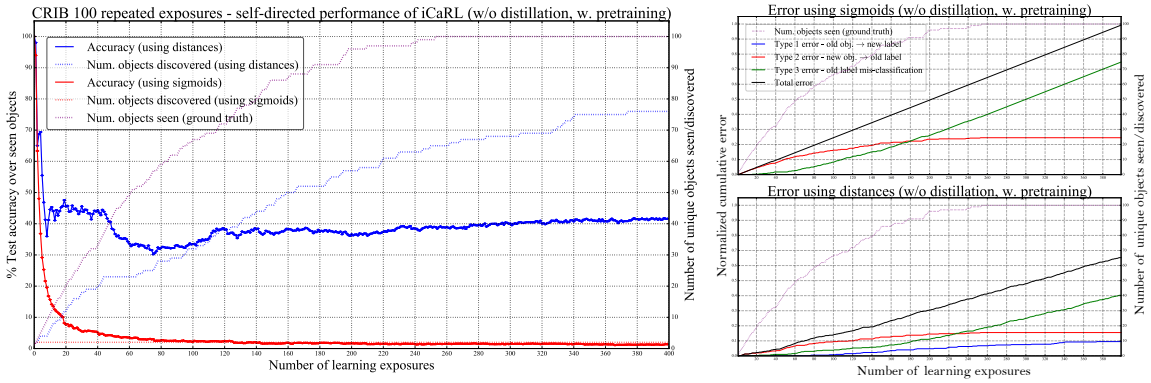
Figure 4.5: (a) Performance of the four baseline algorithms on CRIB50 with repeated exposures. Accuracy numbers and standard-deviation bars are computed similarly to the experiment in Figure 4.4. The violin plots show the distribution of changes in test accuracies across all objects with respect to the gap in exposure to the same instance (b), and the exposure count for a certain object (c). See the text for details.

estingly, the relative performance between different methods roughly remains the same as the single exposure experiment. This is possibly because the relative learning capacity of each method has not changed. Also noteworthy is iCaRL-PT-ND's performance, which is able to recover from an accuracy of 70% to reach a final value of 90%, which clearly demonstrates the importance of pre-training for an incremental learner. Like the single exposure experiment (see Figure 4.4), we expect a similar trend with pre-trained models on other datasets.

The figures follow what might be expected of a naturally forgetting learning algorithm. The trend in Figure 4.5b suggests that the learning algorithm benefits more from the repeated exposure of an object when it is seen after a long gap.

Self-directed Learning:

In this section we present baseline results on self-directed incremental learning using CRIB- with 100 objects. As previously explained in §4.4, in this setting the learner does not have access to the object label, and can only be sure about the permanence of the object in a single data unit. In addition to evaluating whether the algorithm can make use of repeated



(a) (b)

Figure 4.6: (a) Performance on the self-directed learning paradigm for CRIB100 using the approach described in §4.5.1. The results are shown for two variations on how a methods decides which exposure is a new object, or which class an exposure belongs to. The supplemental material describes how accuracy is computed at each learning exposure. The plots in (b) show how errors of different types (see §4.4) accumulate as the learning progresses. The top graph shows errors when using sigmoids (red line (a)), and the bottom graph is when using distances to exemplar means (blue line (a)).

exposures, we evaluate whether the algorithm can accurately build a set of unique classes to match the true labels of objects (see §4.4). The random ordering of object instances and number of exposures is the same as the repeated exposure experiment. In this experiment we evaluate the performance of our baseline solution, presented in §4.5.1, with a pre-trained initialization, since it is the best performing in our previous experiments.

Figure 4.6 shows the results for both variations of the self-directed learner explained in §4.5.1. The plot shows that using distances to exemplar means is far better than using sigmoids for making label decision for each incoming data-unit (in terms of final test accuracy). When we inspect errors, we observe that the sigmoid variant does not have enough capacity to discriminate, which introduces errors at the start of learning. This deteriorates the sigmoid based learner's classifier to a chance performance when the experiment concludes. On the other hand, the exemplar distance learner is able to gradually improve test accuracy after 90 learning exposures. This improvement is also reflected in the distance values, which is discriminative enough metric for the task of discovering new objects, and for assigning repeated exposures to the correct old labels. This finding is supported by

Figure 4.6b which shows that the distance learner adds very few type 1 and 2 errors as the learner settles on a good feature representation.

4.6 Conclusion

We introduce a new incremental learning problem, supported from knowledge in developmental psychology. To motivate work in this domain, we develop the CRIBdata generator system, which has the ability to generate arbitrary amount of image sequences. This process of collecting data is a clear departure from previous fixed datasets, which would help facilitate research on incremental learning. One of the key paradigms we introduce in this chapter is the problem of self-directed learning, which can be thought of as the unsupervised learning equivalent in a developmental context. We conclude from our current experiments that both repeated exposures to the same object instances and pre-training can increase the performance of an exemplar based incremental learner.

References

[1] E. S. Spelke, "Principles of object perception," *Cognitive science*, vol. 14, no. 1, pp. 29–56, 1990.

[2] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha, "Visual parsing after recovery from blindness," *Psychological Science*, vol. 20, no. 12, pp. 1484–1491, 2009.

[3] C. Von Hofsten and K. Rosander, "Development of smooth pursuit tracking in young infants," *Vision research*, vol. 37, no. 13, pp. 1799–1810, 1997.

[4] R. Girshick, "Fast r-cnn," in *ICCV*, Dec. 2015, pp. 1440–1448.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[6] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *ICCV*, Dec. 2013, pp. 2192–2199.

[7] G. Johansson, "Visual perception of biological motion and a model for its analysis," *Perception & Psychophysics*, vol. 14, no. 2, pp. 201–211, 1973.

[8] I. Bülthoff, H. Bülthoff, and P. Sinha, "Top-down influences on stereoscopic depth-perception," *Nature Neuroscience*, vol. 1, no. 3, pp. 254–257, 1998.

[9] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or, "Emerging images," *ACM Trans. Graph.*, vol. 28, no. 5, 163:1–163:8, Dec. 2009.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[11] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: integrated recognition, localization and detection using convolutional networks," in *ICLR*, Apr. 2014.

[12] *Imagenet large scale visual recognition challenge - video*, http://www.image-net.org/, Accessed: 2016-09-20.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: single shot multibox detector," in *ECCV*. Springer International Publishing, 2016, pp. 21–37.

[15] A. Humayun, F. Li, and J. M. Rehg, "The middle child problem: revisiting parametric min-cut and seeds for object proposals," in *ICCV*, Dec. 2015, pp. 1600–1608.

[16] J. D. Golomb and N. Kanwisher, "Higher level visual cortex represents retinotopic, not spatiotopic, object location," *Cerebral Cortex*, bhr357, 2011.

[17] R. F. Schwarzlose, J. D. Swisher, S. Dang, and N. Kanwisher, "The distribution of category and location information across object-selective regions in human visual cortex," *Proceedings of the National Academy of Sciences*, vol. 105, no. 11, pp. 4447–4452, 2008.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, Jun. 2014, pp. 580–587.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, vol. 8691, 2014, pp. 346–361.

[20] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler, "Segdeepm: exploiting segmentation and context in deep neural networks for object detection," in *CVPR*, IEEE, Jun. 2015, pp. 4703–4711.

[21] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *CoRR*, vol. abs/1412.1441 (v3), 2015.

[22] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *ECCV*, vol. 8695, 2014, pp. 297–312.

[23] S. Zagoruyko, A. Lerer, T. Lin, P. H. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár, "A multipath network for object detection," in *BMVC*, BMVA Press, 2016.

[24] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *ICCV*, Dec. 2015, pp. 1134–1142.

[25] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *CVPR*, IEEE, 2012, pp. 3378–3385.

[26] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Free-form region description with second-order pooling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 6, pp. 1177–1189, 2015.

[27] J. Carreira and C. Sminchisescu, "Cpmc: automatic object segmentation using constrained parametric min-cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1312–1328, Jul. 2012.

[28] I. Endres and D. Hoiem, "Category-independent object proposals with diverse ranking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 222–234, Feb. 2014.

[29] P. Krähenbühl and V. Koltun, "Learning to propose objects," in *CVPR*, IEEE, Jun. 2015, pp. 1574–1582.

[30] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[31] B. Bonev and A. Yuille, "A fast and simple algorithm for producing candidate regions," in *ECCV*, vol. 8691, 2014, pp. 535–549.

[32] J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marqués, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[33] P. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *NIPS*, 2015, pp. 1990–1998.

[34] T. Malisiewicz and A. A. Efros, "Improving spatial support for objects via multiple segmentations," in *BMVC*, BMVA Press, 2007, pp. 55.1–55.10.

[35] A. Humayun, F. Li, and J. M. Rehg, "Rigor: reusing inference in graph cuts for generating object regions," in *CVPR*, Jun. 2014, pp. 336–343.

[36] I. Endres and D. Hoiem, "Category independent object proposals," in *ECCV*, vol. 6315, 2010, pp. 575–588.

[37] P. Rantalankila, J. Kannala, and E. Rahtu, "Generating object segmentation proposals using global and local search," in *CVPR*, Jun. 2014, pp. 2417–2424.

[38] P. Krähenbühl and V. Koltun, "Geodesic object proposals," in *ECCV*, vol. 8693, 2014, pp. 725–739.

[39] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik, "Recognition using regions," in *CVPR*, 2009, pp. 1030–1037.

[40] J. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, T. Kröger, J. Lellmann, N. Komodakis, B. Savchynskyy, and C. Rother, "A comparative study of modern inference techniques for structured discrete energy minimization problems," *International Journal of Computer Vision*, vol. 115, no. 2, pp. 155–184, 2015.

[41] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 30–55, 1989.

[42] V. Kolmogorov, Y. Boykov, and C. Rother, "Applications of parametric maxflow in computer vision," in *ICCV*, Dec. 2007, pp. 1–8.

[43] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *ICCV*, Dec. 2013, pp. 1841–1848.

[44] T. Lin, M. Maire, S. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[45] J. H. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 814–830, Apr. 2016.

[46] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

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

[48] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, Nov. 2012.

[49] M. Everingham, S. Eslami, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: a retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[50] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, Jun. 2016, pp. 3213–3223.

[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, Jun. 2015, pp. 1–9.

[52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, Apr. 2015.

[53] P. H. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *ECCV*, 2016, pp. 75–91.

[54] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *ECCV*, 2016, pp. 534–549.

[55] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman, "Using multiple segmentations to discover objects and their extent in image collections," in *CVPR*, vol. 2, 2006, pp. 1605–1614.

[56] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151–172, 2007.

[57] C. Zitnick and P. Dollár, "Edge boxes: locating object proposals from edges," in *ECCV*, 2014, pp. 391–405.

[58] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: binarized normed gradients for objectness estimation at 300fps," in *CVPR*, Jun. 2014, pp. 3286–3293.

[59] J. Dong, Q. Chen, S. Yan, and A. Yuille, "Towards unified object detection and semantic segmentation," in *ECCV*, vol. 8693, 2014, pp. 299–314.

[60] V. Yanulevskaya, J. Uijlings, and N. Sebe, "Learning to group objects," in *CVPR*, Jun. 2014, pp. 3134–3141.

[61] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: a learned mid-level representation for contour and object detection," in *CVPR*, Jun. 2013, pp. 3158–3165.

[62] Y. Lim, K. Jung, and P. Kohli, "Efficient energy minimization for enforcing label statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 9, pp. 1893–1899, Sep. 2014.

[63] ——, "Energy minimization under constraints on label counts," in *ECCV*, 2010, pp. 535–551.

[64] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich, "Diverse m-best solutions in markov random fields," in *ECCV*, vol. 7576, 2012, pp. 1–16.

[65] J. Yang, Y.-H. Tsai, and M.-H. Yang, "Exemplar cut," in *ICCV*, Dec. 2013, pp. 857–864.

[66] P. Kohli and P. H. Torr, "Dynamic graph cuts for efficient inference in markov random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2079–2088, Dec. 2007.

[67] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[68] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[69] Y. J. Lee, J. Kim, and K. Grauman, "Key-segments for video object segmentation," in *ICCV*, Dec. 2011, pp. 1995–2002.

[70] D. Banica, A. Agape, A. Ion, and C. Sminchisescu, "Video object segmentation by salient segment chain composition," in *ICCV Workshops*, Dec. 2013, pp. 283–290.

[71] K. Fragkiadaki, P. Arbeláez, P. Felsen, and J. Malik, "Learning to segment moving objects in videos," in *CVPR*, IEEE, Jun. 2015, pp. 4083–4090.

[72] G. Sharir and T. Tuytelaars, "Video object proposals," in *CVPR Workshops*, Jun. 2012, pp. 9–14.

[73] Z. Wu, F. Li, R. Sukthankar, and J. M. Rehg, "Robust video segment proposals with painless occlusion handling," in *CVPR*, Jun. 2015, pp. 4194–4203.

[74] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, 2005, pp. 886–893.

[75] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *ECCV*, vol. 6315, 2010, pp. 282–295.

[76] F. Xiao and Y. J. Lee, "Track and segment: an iterative unsupervised approach for video object proposals," in *CVPR*, Jun. 2016, pp. 933–942.

[77] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *CVPR*, Jun. 2015, pp. 447–456.

[78] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool, "Online video seeds for temporal window objectness," in *ICCV*, Dec. 2013, pp. 377–384.

[79] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool, "Seeds: superpixels extracted via energy-driven sampling," *International Journal of Computer Vision*, vol. 111, no. 3, pp. 298–314, 2015.

[80] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid, "Spatio-temporal object detection proposals," in *ECCV*. 2014, pp. 737–752.

[81] S. Manen, M. Guillaumin, and L. Van Gool, "Prime object proposals with randomized prims algorithm," in *ICCV*, Dec. 2013, pp. 2536–2543.

[82] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," in *CVPR*, Jun. 2016, pp. 817–825.

[83] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *ICCV*, Dec. 2015, pp. 3119–3127.

[84] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang, "T-CNN: tubelets with convolutional neural networks for object detection from videos," *CoRR*, vol. abs/1604.02532, 2016.

[85] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang, "Object detection in videos with tubelet proposal networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[86] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," *CoRR*, vol. abs/1602.08465, 2016.

[87] B. Drayer and T. Brox, "Object detection, tracking, and motion segmentation for object-level video segmentation," *CoRR*, vol. abs/1608.03066, 2016.

[88] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 3038–3046.

[89] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems 29 (NIPS)*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 379–387.

[90] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 850–865.

[91] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 3074–3082.

[92] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[93] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," *CoRR*, vol. abs/1711.06368, 2018.

[94] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: efficient convolutional neural networks for mobile vision applications," vol. abs/1704.04861, 2017.

[95] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[96] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[97] T.-H. Vu, W. Choi, S. Schulter, and M. Chandraker, "Memory warps for learning long-term online video representations," *CoRR*, vol. abs/1803.10861, 2018.

[98] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[99] G. Bertasius, L. Torresani, and J. Shi, "Object detection in video with spatiotemporal sampling networks," *CoRR*, vol. abs/1803.05549, 2018.

[100] R. Hou, C. Chen, and M. Shah, "An end-to-end 3d convolutional neural network for action detection and segmentation in videos," *CoRR*, vol. abs/1712.01111, 2017.

[101] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran, "Detect-and-track: efficient pose estimation in videos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2018.

[102] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, 2014, pp. 818–833.

[103] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *ECCV*, 2010, pp. 140–153.

[104] B. Chen, J.-A. Ting, B. Marlin, and N. de Freitas, "Deep learning of invariant spatio-temporal features from video," in *NIPS Deep Learning and Unsupervised Feature Learning Workshop*, 2010.

[105] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, Jun. 2011, pp. 3361–3368.

[106] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," in *ICML*, Omnipress, Jun. 2010, pp. 495–502.

[107] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Human Behavior Understanding*, ser. Lecture Notes in Computer Science, vol. 7065, 2011, pp. 29–39.

[108] L. Sun, K. Jia, T.-H. Chan, Y. Fang, G. Wang, and S. Yan, "Dl-sfa: deeply-learned slow feature analysis for action recognition," in *CVPR*, Jun. 2014, pp. 2625–2632.

[109] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014, pp. 568–576.

[110] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Computer Vision (ICCV), IEEE International Conference on*, Dec. 2015, pp. 4489–4497.

[111] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014, pp. 1725–1732.

[112] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: learning optical flow with convolutional networks," in *ICCV*, Dec. 2015, pp. 2758–2766.

[113] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Deep end2end voxel2voxel prediction," *CoRR*, vol. abs/1511.06681, 2015.

[114] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *CoRR*, vol. abs/1708.05038, 2017.

[115] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 4724–4733.

[116] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár, "Exploring weak stabilization for motion feature extraction," in *CVPR*, Jun. 2013, pp. 2882–2889.

[117] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: a benchmark for multi-object tracking," *ArXiv:1603.00831 [cs]*, Mar. 2016, arXiv: 1603.00831.

[118] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: towards a benchmark for multi-target tracking," *ArXiv:1504.01942 [cs]*, Apr. 2015, arXiv: 1504.01942.

[119] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, Jun. 2016, pp. 4340–4349.

[120] J. Yuen, B. Russell, C. Liu, and A. Torralba, "Labelme video: building a video database with human annotations," in *ICCV*, Sep. 2009, pp. 1451–1458.

[121] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, Jun. 2012, pp. 3354–3361.

[122] S. Tripathi, S. Belongie, Y. Hwang, and T. Nguyen, "Detecting temporally consistent objects in videos through object class label propagation," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2016, pp. 1–9.

[123] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[124] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," *CoRR*, vol. abs/1602.08465, 2016.

[125] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, vol. 8693, 2014, pp. 740–755.

[126] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[127] J. Redmon and A. Farhadi, "Yolov3: an incremental improvement," *ArXiv*, 2018.

[128] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *European conference on computer vision*, Springer, 2012, pp. 340–353.

[129] B. McMurray, "Defusing the childhood vocabulary explosion," *Science*, vol. 317, no. 5838, pp. 631–631, 2007.

[130] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[131] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," May 2014.

[132] O. Wiles and A. Zisserman, "Silnet : single- and multi-view reconstruction by learning from silhouettes," in *BMVC*, BMVA Press, 2017.

[133] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[134] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks."

[135] Z. Li and D. Hoiem, "Learning without forgetting," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 614–629.

[136] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "Icarl: incremental classifier and representation learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[137] E. M. Clerkin, E. Hart, J. M. Rehg, C. Yu, and L. B. Smith, "Real-world visual statistics and infants' first-learned object names," *Phil. Trans. R. Soc. B*, vol. 372, no. 1711, p. 20 160 055, 2017.

[138] D. Yurovsky, L. B. Smith, and C. Yu, "Statistical word learning at scale: the baby's view is better," *Developmental Science*, vol. 16, no. 6, pp. 959–966, 2013.

[139] P. J. Kellman, "Perception of three-dimensional form by human infants," *Perception & Psychophysics*, vol. 36, no. 4, pp. 353–358, 1984.

[140] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision*, 2016, pp. 102–118.

[141] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *ArXiv preprint arXiv:1512.03012*, 2015.

[142] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?" In *European Conference on Computer Vision (ECCV)*, 2016, pp. 202–217.

[143] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, "Understanding real world indoor scenes with synthetic data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4077–4085.

[144] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, p. 201 611 835, 2017.

[145] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017, pp. 4652–4662.

[146] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2004, pp. II–104.

[147] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-100 dataset," *Online: Https://www.cs.toronto.edu* 2014.

[148] D. Lopez-Paz and M. A. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017, pp. 6467–6476.

[149] S Nayar, S Nene, and H. Murase, "Columbia object image library (coil 100)," *Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996.

[150] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[151] M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, *et al.*, "Shrec17 track large-scale 3d shape retrieval from shapenet core55," in *Eurographics Workshop on 3D Object Retrieval*, 2017.

[152] B. Wymann, "Torcs - the open racing car simulator,"

[153] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[154] Conference, *Icdl-epirob*, `http://www.icdl-epirob.org/`.

[155] A. Gepperth and C. Karaoguz, "A bio-inspired incremental learning architecture for applied perceptual problems," *Cognitive Computation*, vol. 8, no. 5, pp. 924–934, 2016.

[156] R. Kemker and C. Kanan, "Fearnet: brain-inspired model for incremental learning," in *International Conference on Learning Representations (ICLR)*, Apr. 2018.

[157] P. J. Kellman and M. E. Arterberry, *The cradle of knowledge: Development of perception in infancy*. MIT press, 2000.

[158] blendswap.com, `https://blendswap.com`.

[159] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[160] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[161] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop Autodiff*, 2017.