UNIVERSITY OF CALIFORNIA,
IRVINE

Training and Evaluating Visual Recognition Systems with Limited Annotations

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Phuc Xuan Nguyen

Dissertation Committee:
Professor Charless Fowlkes, Chair
Associate Professor Deva Ramanan
Chancellor's Professor Padhraic Smyth

2019

DEDICATION

To my parents who have always been there for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First and foremost, I'd like to send the biggest thanks to my advisors, Deva Ramanan and Charless Fowlkes, for their supports and guidance. Through my PhD, I have learned a tremendous amount from them: from technical research skills to personal communication skills. They taught me how to be a better researcher and a better person.

I also want to thank Professor Padhraic Smyth for his time and valuable feedback on my thesis.

I want to thank my labmates, Bailey, Shu, Minhaeng, James, Sam, Golnaz, Xiangxin, Zhe, and Daeyun, for accompanying me through this journey.

Lastly, I want to thank my parents for all the supports and packed meals delievered from San Diego.

CURRICULUM VITAE

Phuc Xuan Nguyen

EDUCATION

Doctor of Philosophy in Computer Science **2019**
University of California, Irvine *Irvine, CA*

Master of Science in Computer Sciences **2013**
University of California, San Diego *San Diego, CA*

Bachelor of Science in Electrial Engineering and Computer Sciences **2010**
University of California, Berkeley *Berkeley, CA*

REFEREED CONFERENCE PUBLICATIONS

The Open World of Micro-videos **July 2016**
CVPR - BigVision Workshop

Weakly Supervised Action Localization by Sparse Temporal Pooling Network **June 2018**
CVPR

Active Testing: An Efficient and Robust Framework for Estimating Accuracy **July 2018**
ICML

ABSTRACT OF THE DISSERTATION

Training and Evaluating Visual Recognition Systems with Limited Annotations

By

Phuc Xuan Nguyen

Doctor of Philosophy in Computer Science

University of California, Irvine, 2019

Professor Charless Fowlkes, Chair

In recent years, large-scale datasets with high-quality annotations have enabled many significant discoveries in computer vision and machine learning. However, the luxury of large-scale human annotations is not always affordable. For example, image semantic segmentation requires annotations for every pixel in an image. Similarly, action localization in untrimmed videos requires fine boundaries of action instances. The annotation effort for these tasks is prohibitively expensive to scale. What can we do in such scenarios? In this thesis, we look at different ways to circumvent this challenge. First, we examine micro-videos, a new and abundant source of visual data with user-generated metadata that could serve as web-supervision. Second, we explore weak supervision, in the context of action localization. While fine boundaries annotations of action instances are expensive and difficult to obtain, video-level labels can be cheaper and more accessible. We present two state-of-the-art systems that leveraged on this weaker form of supervision and compare them to their fully supervised counterparts. Lastly, we address the problems of accurately measuring the performance of computer vision systems with limited human annotations.

Chapter 1

Overview

Recent successes in computer vision have been largely due to large-scale, well-annotated datasets. High-quality human annotations in these datasets allows researchers to train supervised vision recognition systems with performance on par with human's. However, it can be prohibitively expensive to scale up the annotations. For example, the authors of NUS-WIDE [20], a multi-label image classification dataset, estimates 3000 man-hours to semi-manually annotate the complete ground-truth for a relative small set of 81 concepts across 270K images. The authors of Citiscapes [21], a popular image semantic segmentation dataset, reports that an average of 1.5 hours is required to annotate a single image. In the setting of limited human annotations, it is unclear how to train and test visual recognition systems. Could we scale up the annotations in some other ways? Is there some similar form of visual data that comes with freely-available annotations? Instead of requiring tedious human efforts to label all pixels in the image, how far do we get with a weaker form of supervision, e.g. image-level annotations that tell what parts are in the image instead of their exact boundaries? Measuring performance of a system also requires large-scale annotations. How is the test performance affected by limited high-quality annotations? What if we relaxed this constraint and report the test performance on a noisier type of annotations? In this

thesis, we explore these questions.

Chapter 2 investigates webly supervision. More specifically, we closely examine a new form of visual data, called micro-videos, that are abundant freely on social media platform. These micro-videos often comes with user-generated metadata, such as tags and timestamps. Figure 1.1 shows example frames with its tags organized into different categories. We compare micro-videos characteristics with traditional video classification datasets. We extend the analysis to a larger scale: what are the effects of using raw, uncurated tags as labels versus a curated set of tags?

Figure 1.1: Example micro-video frames. This new source of visual spans across different video contents: from atomic unbreakable actions to videos of a regular animal like cats.

Chapter 3 and 4 explore weak supervision in the context of action localization. Figure 1.2 summarize the task of action localization and the annotations format for full supervision and weak supervision. Chapter 3 introduces a convolution neural network with sparse attention-weighted temporal pooling. At inference time, we extract and score temporal proposals using temporal class activations and class-agnostic attention. Chapter 4 identifies this network's

flaws and proposes modifications that lead to significant improvements. The final model achieves state-of-the-art on weakly-supervised action localization and has comparable results with fully-supervised systems.

The lack of high-quality human annotations also affects the testing process. How much can you trust your performance metric when the testbed is too small? Can we trust a larger testbed with noisy labels? In Chapter 5, we introduce a framework to accurately estimate performance metrics of computer vision systems when using a cheaper, noisier labels. We re-formulate the problem as one of active testing, and examine strategies for efficiently querying a user to obtain accurate performance estimate with minimal human effort.

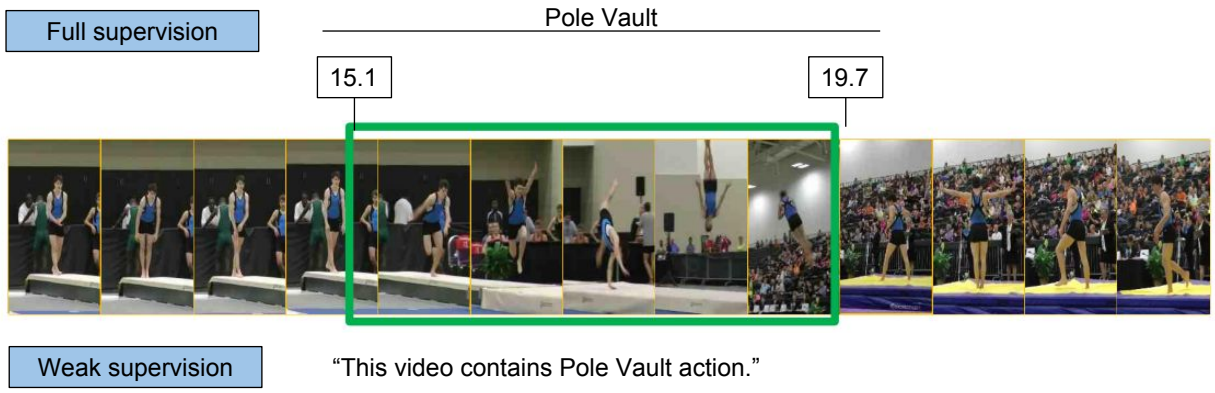


Figure 1.2: Given an untrimmed video, the task of temporal action localization is to find the boundaries (start and end timestamps) and the label for all the action instance. Fully-supervised models required the boundaries and action label annotations for each action instance in the video. A weaker, and often cheaper, form of supervision is video-level labels, indicating which action the video contains, but not how many instances or the locations of those instances.

Chapter 2

The Open World of Micro-Videos

Can user-generated tags serve as a substitute to human annotations? We examine a new source of visual data which comes with a cheaper, more-readily source of annotations. As an increasingly prevalent form of media, micro-videos are time-constrained (typically 5-10 second) video clips commonly appeared on social networking sites such as Instagram, Vine, and Snapchat. Micro-videos can be interpreted as the visual analog of a character-limited micro-blogs or "tweets" [60]. An estimated 12 million micro-videos are posted to Twitter each day. The number of micro-videos produced *surpasses the total inventory of YouTube every 3 months*. From an applied perspective, this flood of visual data is increasingly important and has unique characteristics that are not addressed by existing computer vision methodologies and benchmarks. We further argue that the microvideo format offers unique opportunities for basic research in building systems that address lifelong learning in *open-world* visual domains.

Ease of storage/processing: A particularly attractive aspect of micro-videos is the ease of large-scale collection, storage and processing. While large-scale video datasets (EventNet [131], YouTube8M [2], Sporst1M [52]) are available, none has become a gold-stardard benchmark for video understanding. One reason is that it is notoriously challenging to store, and process

a large diverse video collection because of resource constraints. To ease the collection process, existing video collections often contain multiple snippets cropped from a few longer videos [58, 66]. This limits the diversity of the dataset. It requires hundreds of terabytes to store the complete Youtube8M dataset. To mitigate this constraint, only the video URLs and the top-level features are provided. However, URLs are often an unreliable source as videos can come and go quickly. Micro-videos, on the hands, are easy to collect and compact to store at a large-scale. The storage size of 260K videos is only 220GB.

Micro-narratives: Another intriguing aspect of micro-videos is that they arrive "ready-for-analysis", due to the constraint that users are forced to trim content to fit within the six-second restriction. As a result, each frame of a micro-video typically has high information content compared to frames in a longer unconstrained video. This changes strategies for automated understanding and may even eliminate the need for common video preprocessing steps like keyframe or shot selection. The long format of traditional videos (such as YouTube) often forced models to operate on an sparse subsampled subset of frames and/or only use the first X seconds of the video. Whereas, there are only 180 frames in total for a microvideo. Hence, it is within reason for models to use all the available frames for analysis.

Viewpoint: A unique technical aspect of micro-videos is camera viewpoint (Fig. 2.4). Current action datasets in computer vision focus on *third-person* depictions of actions, where a person(s) performing an action or activity is framed in the view. In contrast, a significant fraction of socially-driven micro-videos include *egocentric* viewpoints, where the photographer is participating in the action. Another camera configuration is a *self-facing* viewpoint, or "selfie", where a single user is both the photographer and subject. This is particularly interesting in the study of interactions photographers and their subjects [68]. Such diverse camera viewpoints represent new modes of video acquisition that are not typically addressed in previous work and deserve a closer look from the computer vision community.

Figure 2.1: Micro-videos semantically rich micro-narratives while remaining tractable to collect, store and process. Mobile-videographers often interact with the scene and its subjects resulting in a wide range of camera viewpoints including **egocentric** views of activities and **self-facing** shots where a single individual is both the photographer and subject. Our dataset, **MV-58k**, includes common tags about actions and objects seen in other computer vision datasets as well as specific tags such as `#noseguitar` which are video-graphic styles unique to the micro-video sharing service Vine. Distribution of videos per tag is highly skewed.

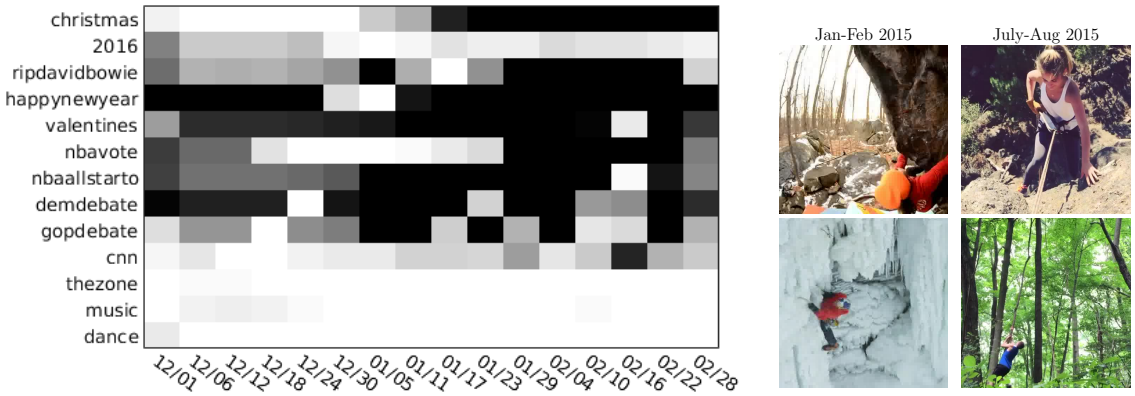
Open-world dynamics: Micro-videos come labeled with hashtags that enable search and play an important role in social communication. These tags provide a form of supervision, automatically labeling our diverse, multi-view, pre-trimmed dataset. In contrast to existing efforts to develop top-down ontologies for activities [28] or events [132], tags form an open-world vocabulary whose usage and semantics changes dynamically over time [22]. For example, the tag `#alternativefacts` did not exist until recently, while the visual meaning of the tag `#apple` expands whenever a new iPhone is released. Microvideos thus provide a unique opportunity to explore learning *in the open world*, where distributions of visual semantics follow long-tail statistics that change time over time. Taking a *bottom-up, data-driven* approach to video content semantics more accurately reflects naturally-occuring long-tail distributions that typically suppressed in hand-curated datasets. The dataset we present is thus large, has dynamic temporal variations and is continually growing is size. We analyze a snapshot as of Feb 2016 consisting of 264,327 videos with 58,243 tags.

2.1 Related Works

There is little existing work on micro-video analysis, as the medium itself is new. Redi *et al.* [82] explore the problem of finding creative micro-videos, inspired by similar studies of image quality assessment. Sano *et al.* [91] analyze the problem of detecting loop micro-videos (that are designed to be played in a continuous six-second loop). Here we focus on analyzing general properties of micro-videos, with the explicit goal of constructing a new, large-scale benchmark for temporally-evolving tag prediction.

Viewpoint modeling: A unique contribution of our dataset is the diversity of camera viewpoints. Existing video benchmarks for action recognition have focused on third-person viewpoints (e.g., HMDB [58], UCF101 [105], Hollywood-2 [69], UT-Interaction [87] and Olympic Sports [75]). Wearable cameras such as Google Glass and GoPro have spurred interest in analyzing egocentric views [29, 57, 77, 61]. Compared to existing action and egocentric datasets, our user-generated micro-videos contain a wider variety of categories (tags) and viewpoints (e.g., self-facing) with richer narrative content, even in a single clip. To understand and highlight these differences we train mixture-of-viewpoint models that specifically target viewpoint variations in dynamic micro-videos (Sec. 2.3) and carry out an extensive comparison with HMDB [58] (Sec. 5.3).

Closed vs open vocabularies: Traditionally, video datasets in computer vision have been labeled with a fixed ontology of activities or events [28, 132, 52]. An alternative perspective (popular in the multimedia community) is to formulate the problem as a multi-label tag or concept prediction task [110, 130, 7, 106, 109]. Our dataset falls into this later camp. In terms of size and diversity, the most relevant prior work appears to be Sports-1M [52] which contains 1M videos in 487 categories, and EventNet [132], which contains 95K videos labeled with 5K concepts. Our dataset already includes 2X more videos and 10X more concept tags. Unlike other video datasets, our data also includes timestamps which allow us to study



(a) tag usage dynamics (b) appearance dynamics

Figure 2.2: Unique temporal structure in microvideos. We visualize changes in tag label priors over time by plotting their popularity rank as a heatmap (brighter denotes higher popularity). Note that many tags exhibit large fluctuations; `#gopdebate` is ranked first during the weeks of the associated events, but dramatically decreases otherwise. The distribution of visual appearance associated with some tags also exhibit temporal dynamics. `#climbing` shows large temporal variations over summer and winter months due to changes in scenery (desert rocks versus ice) and equipment.

temporally-varying semantics, a relatively unexplored concept in vision, with the notable exception of [56]. Importantly, tag frequency distributions are highly imbalanced, following a natural long-tail distribution (Fig. 2.3). While highly imbalanced class distributions are somewhat uncommon in current vision datasets, they appear to be a fundamental aspect of life-long learning in the open-world [19]. With the advent of deep architectures that appear capable of transferring knowledge across imbalanced classes [10], we think the time is right to (re)consider learning in the open-world!

2.2 Dataset

In this section, we describe our ongoing data-collection process and analyze the statistics of micro-video tags and viewpoints that make our dataset distinct from existing video benchmarks.

Streaming dataset collection: We collect a stream of Vine videos by daily querying of Vine's API [1]. To ensure a diverse stream, we query the 300 most popular and 300 most recent videos across multiple community-curated channels (Comedy, Sports, Musics). We typically obtain 6000 videos daily. Each video is associated with a collection of hashtags that are added to our open vocabulary. On average, a video contains 1.56 hashtags, but this statistic is skewed by the fact that nearly half the videos do not contain any tags (56%).This indirectly motivates one practical application of our dataset - automatic prediction of hashtags. We visualize our overall distribution of tags in Fig. 2.1. We use **MV-58k** to refer to a snapshot of this datastream collected during the period Dec-2015 to Feb-2016.

Curation: To examine the amount of "noise" in the tag stream and perform diagnostic comparisons to existing activity datasets, we manually curated a subset of 40 tags and their associated 4000 videos. We selected 40 representative tags that span both common actions as well rare tags "in-the-tail". We "clean" this dataset by merging synonymous tags (e.g., `#horseback` and `#horsebackriding`), removing mistagged (spam) videos, and removing videos in which there was only circumstantial visual evidence for the tagged activity (see Fig. 2.3 for an example). We added additional annotations to each video including viewpoint and a dominant (tag) category. The latter allows us to recast tag prediction as a K-way classification problem, simplifying our diagnostic analysis.

We refer to this curated dataset as MV-40, and contrast this with (the fixed snapshot of) our uncurated, open-world dataset MV-58K. We organize MV-40 into broad categories of atomic actions, recreational activities, competitive sports, objects, and vine-specific. We visualize our two-level taxonomy and provide visual examples in Fig. 2.1.

Long-tail distributions: Our collection process reveals a salient property of open-world microvideos; they follow long-tail distributions of tags. This significantly complicates learning because there will be some tags for which we have little training data. While traditionally a notorious challenge for machine learning, our analysis suggests that hierarchical feature

learning (with CNNs) can learn to share, or transfer knowledge from the data-rich tags to the data-sparse tags (i.e., *one-shot learning*). For example, even if we have few examples of `#dunking`, mid-level features learned for a data-rich tag such as `#basketball` may still be useful for the former class.

Temporal dynamics: Our open-world dataset collection has another notable property - both the frequency of tag usage and the visual appearance semantics associated with a given tag evolve over time. In probabilistic terms, we can interpret such dynamics and changes in the *prior* of labels (Fig. 2.2) and the *likelihood* of image features conditioned on the tag label (Fig. 2.7). This suggests developing approaches for continually retraining models as new data becomes available, an idea we explore in Sec.5.3. An extreme case arises when a new tag first appears in the data stream there are no training examples available (e.g., `#trump2016` did not exist until recently). In our current experiments, we simply fail to predict such tags at test time. However, we point out that the temporal appearance of new tags provides a compelling natural example of *zero-shot learning* where side information about the semantic relation between tags could readily be exploited.

Viewpoint: To analyze the effect of viewpoint on recognition accuracy, we manually annotate MV-40 with the viewpoint of each video as egocentric, third-person, or self-facing. In some cases, the viewpoint changed between shots in the video in which case we record all the viewpoints present, as well as the dominant one. With this annotation we can treat viewpoint prediction as either a multi-label attribute prediction problem (where multiple viewpoints can be present in video) or a multi-class problem (where the goal is to predict the dominant viewpoint). We report the per-tag viewpoint statistics in Fig. 2.4. Interestingly, even at the shot-level, we sometimes find ambiguities in viewpoint. Some social activities (such as `#bicycling`) involve both the photographer and subjects in view, suggesting a simultaneous egocentric and third-person viewpoint.

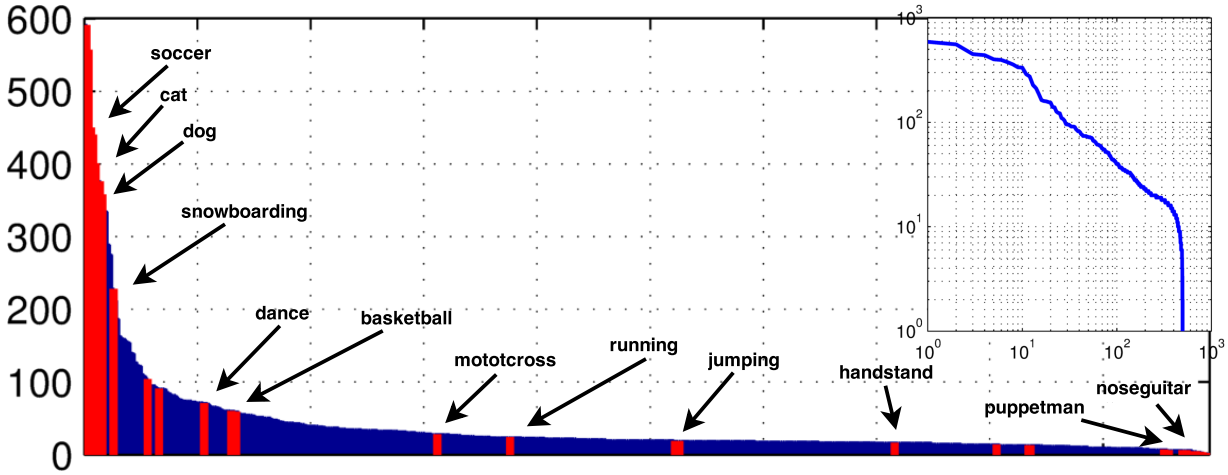


Figure 2.3: For diagnostic purposes we construct a hand-curated dataset (MV-40) containing only videos with direct visual evidence for a subset of 40 tags selected to span both common and rare tags, as shown in the long-tailed distribution. Inset shows the same distribution on a log-log scale.

2.3 Methods

In this section, we describe several baseline models for tag prediction and viewpoint classification. As the focus of our work is not on video feature extraction, we make use of standard feature sets. Recent results from the THUMOS evaluation benchmark [37] suggest that CNN spatial features (VGG [99]) combined with temporal motion features (IDT [115]) make for a reasonable video descriptor.

Appearance features: We found a good tradeoff in speed, storage, and accuracy with the following simple pipeline: given a video, (1) run off-the-shelf VGG-16 models [99] on 15 equally-spaced frames from a video, (2) for each frame, extract (6144-dimensional) multi-scale features across multiple layers [129], and (4) max-pool the resulting features across the 15 frames [88].

Motion features: While recent state-of-the-art results on video datasets have made use of optical flow features to represent motion features, Improved Dense Track (IDT) [115] still

offers strong signal for the learning process. This method is based on short-term tracks found from tracking interest points across frames. Various features aligned to these temporal tracks are extracted. We quantize these descriptors using K-means.

Cue-combination: We train tag classifiers that aggregate features by combining multiple kernels. We experimented extensively with various feature encodings and kernel combinations before settling on the following strategy.

We define the motion similarity with a χ^2-RBF kernel:

$$K_m(x_i, x_j) = exp\left(-\frac{1}{L}\sum_{c=1}^{L}\frac{d(x_i^c, x_j^c)}{A^c}\right), \quad x_i^c, x_j^c \in \mathcal{R}^{4000}$$

where A_c is the average χ^2 distance between all videos in the training data, L is the number of motion channels, $d(x_i^c, x_j^c)$ is the χ^2 distance between x_i and x_j with respect to the c-th channel. We measure the appearance similarity between two clips using a linear kernel: $K_a(x_i, x_j) = \sum_{f=1}^{N}(x_i^f \cdot x_j^f)$ summed over the $N = 15$ static feature channels extracted by the CNN. We compute the similarity between two video clips x_i and x_j by averaging their appearance and motion-feature similarities: $K(x_i, x_j) = \frac{1}{2}(K_m(x_i, x_j) + K_a(x_i, x_j))$. Given a training vocabulary of K tags, we train K binary (one-vs-all) kernelized SVMs using the LIBSVM package [17]. Finally, we calibrate each predictor using Platt scaling [78].

Viewpoint mixtures: Different viewpoints of the same tag can have significant differences in video content, as shown in Fig. 2.1. For example, a third-person and egocentric `#bicycling` video contain very different motions and appearances. To analyze such variations in the MV-40 diagnostic dataset, we train viewpoint-specific models for each tag. The final confidence associated with a tag prediction is the maximum score across the three viewpoint-specific models.

Temporally-adapted models: To explore the temporal evolution of tag semantics and

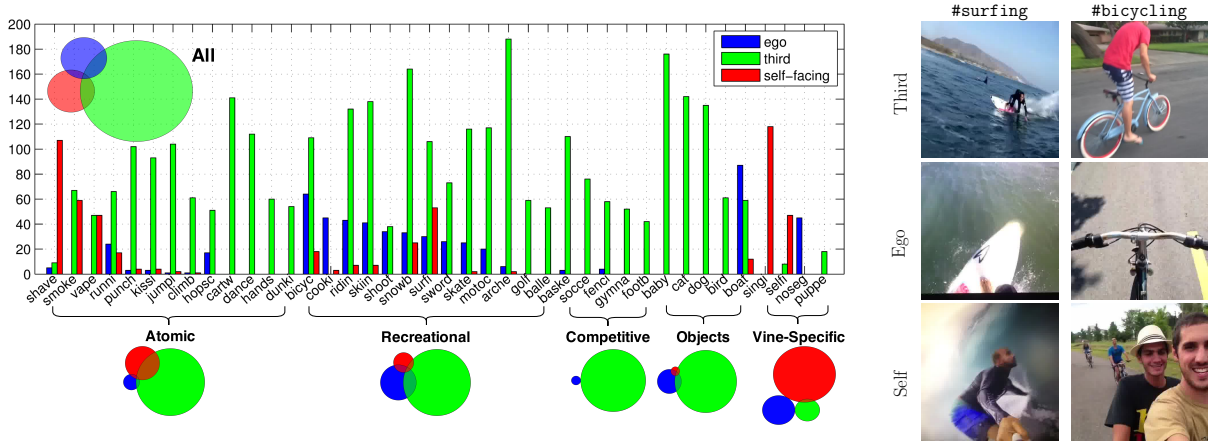


Figure 2.4: The unique viewpoints of microvideos. (**a**) shows the distribution of camera viewpoints annotated in MV-40. Most tags tend to be associated with third-person viewpoints. Some atomic actions such as `#shave` are captured with self-facing views. Recreational sports are often in egocentric views, but this is less likely for competitive sports (soccer). Many Vine-specific tags make use of self-facing viewpoints. The Venn diagram shows that there's a significant amount of videos that contain more than one viewpoint and some even has all 3 viewpoints. We show examples of frames with unique viewpoints on the right.

videos, we evaluate models trained with videos sampled over different temporal windows. Consider the task of predicting the label y for a video x collected at time t_0 using with models trained on a stream of timestamped training videos indexed by time. We consider models trained on three different subsets of training data:

$$\{x_t, y_t : \forall t\} \rightarrow \text{non-causal model} \tag{2.1}$$

$$\{x_t, y_t : t < t_0\} \rightarrow \text{causal model} \tag{2.2}$$

$$\{x_t, y_t : t_0 - \Delta < t < t_0\} \rightarrow \text{adaptive model} \tag{2.3}$$

where Δ is a specified window size. The above approach can be simplified by making some assumptions about the nature of temporal variation. For example, if we assume that the popularity of tags changes over time but their appearance models do not, one can model efficiently model temporal dynamics with a *statistical prior shift* [90]. Intuitively, dynamics can be captured with a fixed set of posterior class predictions that are reweighted by dynamically-varying tag priors. Unfortunately, this requires access to tag priors on test data

from the future, which violates causality. Instead, we assume that tag priors vary smoothly over time, and simply use a weighted estimate of recent tags' popularity. We found that the simple approach of applying temporally-weighted Platt rescaling (using a weighted dataset where recent videos are given more importance) outperformed an explicit prior model.

2.4 Experiments

In this section, we present an extensive set of experiments on our dataset. We focus on three sets of experiments: a diagnostic evaluation of features and viewpoints on our curated dataset (MV-40), its relation to popular benchmarks such as HMDB [58], and analysis of the open-world dynamics of MV-58K.

2.4.1 Diagnostics

First, we analyze various aspects of our dataset and recognition pipeline, focusing on the curated and annotated MV-40 subset.

Feature comparisons: We begin by comparing the performance of various combinations of our features in Fig. 2.5-(a). We observe CNN features outperform IDT in most category groups except 'Atomic'. Trajectory-based motion and appearance-based deep features are particularly effective when combined, indicating that they capture complementary cues.

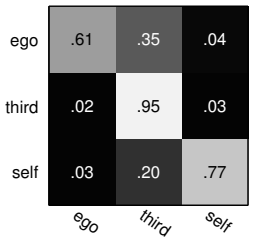
View-specific mixtures: We next evaluate the performance of view-specific tag classifiers and compare to results with a single classifier per tag. When training, one can treat clips from the same tag but different viewpoints as positive training examples (**Pos**), negative training examples (**Neg**), or such clips can be treated as neutral and ignored (**Neu**). Fig 2.5-(b) summarizes of the performance. When averaged over all tags, the performance increase

	IDT	CNN	IDT+CNN
All	53.90	62.13	68.82
Atomic	47.72	46.58	57.19
Recreational	60.81	69.09	76.06
Competitive	49.86	60.29	64.64
Objects	46.14	70.37	73.47
Vine-Specific	63.25	65.81	68.38
Ego	63.25	73.15	79.49
Third	51.45	60.00	67.14
Self	59.27	61.82	68.40

(a) Feature performance

Feature	Pos	Neu	Neg
IDT	62.65	64.98	64.34
CNN	58.28	60.82	60.47
IDT+CNN	68.57	70.61	70.40

(b) View-specific mixtures



(c) Viewpoint confusion

Figure 2.5: (a) Performance of different feature sets on MV-40 broken down by tag type and viewpoint. (b) plots the perfomance of view-point specific tag models, exploring different choices of positive and negative data. (c) visualizes the 3-way class confusion matrix for viewpoint prediction. See text for more details.

from view-specific mixtures is rather negligible (0.6% for our combined features). We also evaluate only those 11 classes for which additional views were trained. We see a small but definite improvement of 2.1%. Ignoring clips from other viewpoints (**Neu**) slightly increases performance.

Viewpoint prediction: We also investigate the task of viewpoint prediction: what is the viewpoint of a test video? Fig 2.5-(c) summarizes viewpoint confusions. Egocentric views are often confused with third-person and the accuracies for egocentric drops significantly. This is consistent with Fig. 2.4, which suggests many recreational activities involve both the photographer and subjects involved in the action. If we score viewpoint prediction as a multilabel problem (where each video could be labeled with more than one viewpoint), accuracy for `ego`, `third`, and `self-facing` jump to 92%, 90%, and 93%. This suggests the presence of any given viewpoint can be accurately predicted.

2.4.2 Cross-dataset Generalizability

We compare the overall difficulties and cross-dataset generalizability between our dataset and a standard action recognition benchmark, HMDB-51 [58]. Overall, the average accuracy on **vine-15** is lower than HMDB (65.99% vs. 71.27%), suggesting that micro-videos are

more challenging. To quantify how biased or general a dataset may be, we measure the 'performance drop' between different datasets [107]. The cross-dataset drop for Vine data (13%) is significantly lower than HMDB (26.75%), which suggests models trained on micro-videos are more able to generalize to other datasets.

Viewpoint: One might hypothesize that since HMDB contains mostly third-person views, it won't generalize to the other viewpoint in our data. To test this, we extract a smaller subset **vine-3rd** containing only third-person viewpoints. Fig 2.6-(g) shows that **vine-3rd** is more similar to HMDB, as models trained on HMDB data perform better on **vine-3** than **vine-15**. However, performance drop for models trained on **vine-3rd** is still significantly smaller than those trained on HMDB (12.88% vs. 24.4%). This suggests that even accounting for viewpoint, our videos still generalize better than HMDB.

Temporally iconic videos: The concept of "iconic views" in object recognition refers to "easy" images with a clear and distinctive depiction of an object, often close cropped or in an uncluttered setting without occlusion. We apply this notion to video, defining a *temporally iconic depiction* of a tag as one where temporal clutter has been removed by trimming down the video clip to focus on the core action. We manually segment each video in **vine-15** to derive an iconic version **vine-icon** and use **vine-3rd-icon** to denote the third-person subset. Fig. 2.6 reveals that **vine-icon** and **vine-3rd-icon** are more similar to HMDB (following our previous analysis). However, their cross-dataset performance drops (14.06% and 12.86% respectively) are still significantly smaller than the drop by models trained **HMDB** data (25.35% and 21.89%). *To summarize, even though third-person and temporal iconic-ness accounts for much of the difference between HMDB and our dataset, our micro-videos can still generalize better.*

Train	Test	avg
vine-15	vine-15	65.99
vine-15	hmdb	52.86
vine-3rd	vine-3rd	65.82
vine-3rd	hmdb	52.94

(a)

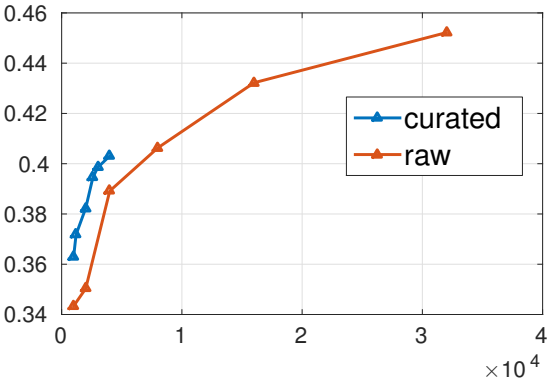
Train	Test	avg
vine-icon	vine-icon	68.03
vine-icon	hmdb	53.97
vine-3rd-icon	vine-3rd-icon	68.65
vine-3rd-icon	hmdb	55.79

(b)

Train	Test	avg
hmdb	hmdb	71.27
hmdb	vine-15	44.52
hmdb	vine-3rd	46.87
hmdb	vine-icon	45.92
hmdb	vine-3rd-icon	49.38

(c)

Figure 2.6: Cross-dataset Generalizability Comparisons.



(a) Effect of curated and noisy data.

Category	Non-causal	Causal	Adaptive
40 categories	44.55	38.05	43.60
`#climbing`	6.87	5.27	21.16

(b) Temporal dynamics of micro-videos

Figure 2.7: (a) Effect of data curation and (b) temporal dynamic of large-scale micro-videos. In (a), we compare the accuracy of models trained versus uncurated data when tested on MV-58k. Curated, clean data is often easier to train on. However, a modest amount of addtional raw training data (<2X) rivals the accuracy of a manually curated training set. In (b), we show the temporal dynamics of the micro-videos. Non-causal models perform the best but may be impractical because they must be trained on future videos. Adaptive models trained on recent videos perform better than a fixed training set because tag topics tend to temporally evolve.

2.4.3 Micro-videos at Large

In this section, we analyze properties of our open-world MV-58K dataset.

Curated vs raw data: When comparing models trained on curated versus open-world data (Fig. 2.7), given a fixed amount of training data, curated data performs better. However, the performance gap is closed quickly when trained with more ($< 2x$) raw data. This suggests that instead of spending efforts on cleaning the data, collecting additional raw, noisy data will achieve the similar results. This is promising as it is practically a lot easier to collect 2x more raw data than cleaning the current data.

Temporal dynamics: We now examine the time-varying properties of micro-video content and tags (Fig. 2.7). We find that the problem of causal prediction (where one only has access to data from the past) is much harder than the non-causal counterpart, suggesting that micro-videos are not "IID" over time. Current benchmarks are assuming the independence between samples over time and, hence, are likely to be over-estimating real-world performance on streaming data. Much of this temporal variation can be explained by fluctuations of tag popularity, whose degree of variation can vary dramatically across classes.

Open vocabularies: We now move beyond our 40 selected tags to evaluation of an open-world vocabulary. We treat this as a multi-label classification task (since videos can be naturally labeled with many tags). We report the per-tag mean average precision (mAP). Fig. 2.8 shows the per-tag mAP as a function of training-set size in. We tend to see different performance regimes. "Easy" tags perform well even with little training data, likely due to a characteristic appearance that is easy to learn from little training data (`#cavs`). "Challenging" tags appear to contain appearance variation, but are learnable with additional data (`#dogs`). "Unlearnable" tags remain near-zero AP even given lots of training data (`#revine`). We posit that these can be treated as stopwords that fail to capture much semantic meaning of the video.

2.5 Conclusions

We introduce a open-world dataset of micro-videos, which lie in a regime between single images and typical videos, allowing for easy capture, storage, and processing. They contain micro-narratives captured from viewpoints typically not studied in computer vision. Because they are naturally diverse, pre-trimmed, and user-annotated, they can be used a live testbed for open-world evaluation of video understanding systems.

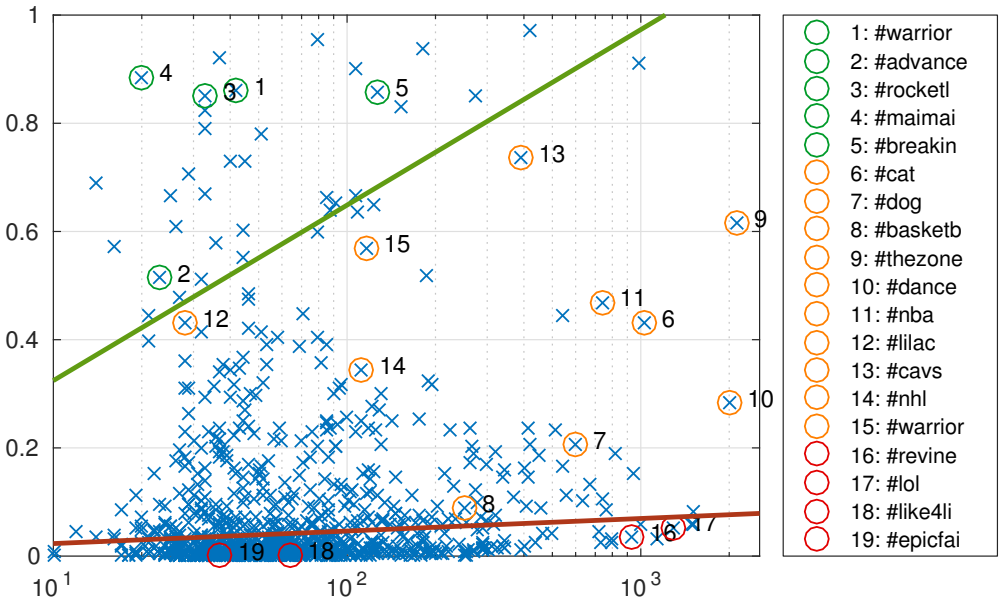


Figure 2.8: Open-world tag prediction. The scatter plot shows per-tag APs vs (log) number of training examples. We rank the "learnability" of a tag by the ratio of its AP to (log) number of training examples, and draw lines to loosely denote regions of easy, challenging, and unlearnable tags. Unlearnable tags appear to correspond to "stopwords" such as `#revine,lol` that do not capture video content. The per-tag mAP_T is 0.05.

Chapter 3

Weakly supervised Action Localization by Sparse Temporal Pooling

In the previous chapter, we explored the possibilities of using web supervision as a replacement for human annotations. In this chapter, we explore weak supervision in the context of action localization. Instead of requiring the effort-intensive annotation effort of labeling the boundaries for actions in videos, we investigate training with a weaker form of supervision: video-level labels. For this task, the objective is to find the boundaries and class of the action instances given an untrimmed video. Human annotation effort for constructing the training data for this task include watching uncut videos and identifying the action boundaries for each of the interested classes. The process of identifying action boundaries can be very noisy and time-consuming. An alternative approach is to train with weak supervision: the model can only observe video-level labels indicating whether the video contains certain actions. These types of annotation efforts are significantly less intensive than pinpointing the exact boundaries of every action instance within a video.

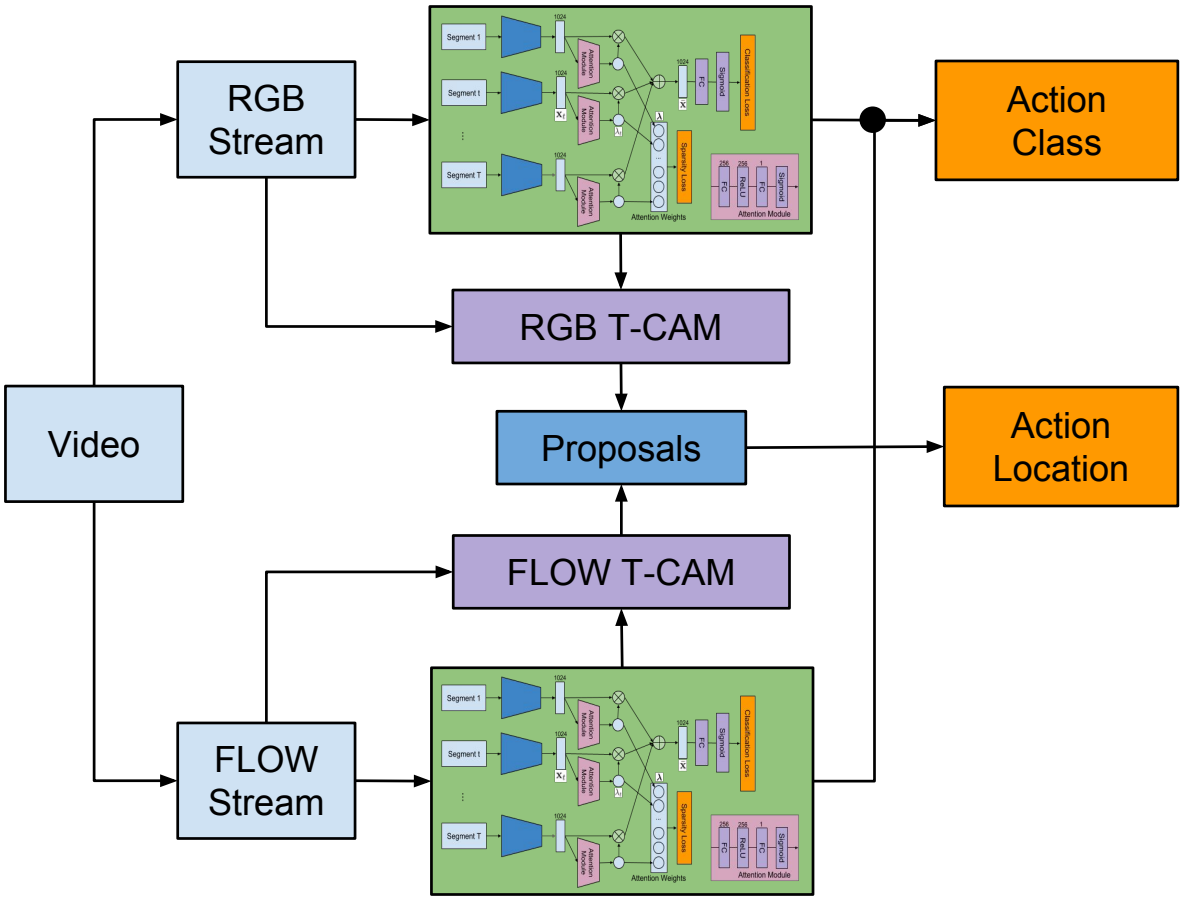


Figure 3.1: Overview of the proposed algorithm. Our algorithm takes a two-stream input—RGB frames and optical flow between frames—from a video, and performs action classification and localization concurrently. For localization, Temporal Class Activation Maps (T-CAMs) are computed from the two streams and employed to generate one dimensional temporal action proposals, from which the target actions are localized in the temporal domain.

3.1 Related Works

Action recognition aims to identify a single or multiple actions per video and is often formulated as a simple classification problem. Before deep learning started being actively used, the algorithm based on improved dense trajectories [116] presented outstanding performance. When it comes to the era of deep learning, convolutional neural networks have been widely used. Two-stream networks [98] and 3D convolutional neural networks (C3D) [108] are popular solutions to learning video representations and these techniques, including their variations, are extensively used for action recognition. Recently, a combination of two-stream

networks and 3D convolutions, referred to as I3D [16], was proposed as a generic video representation learning method. On the other hand, many algorithms develop techniques to recognize actions based on existing representation methods [119, 122, 30, 34, 31, 94].

Action localization is a different from the problem of action recognition, because it requires the detection of temporal or spatiotemporal volumes containing target actions. There are various existing methods based on deep learning including structured segment network [139], contextual relation learning [103], multi-stage CNNs [97], temporal association of frame-level action detections [35], and techniques using recurrent neural networks [133, 67]. Most of these approaches rely on supervised learning and employ temporal or spatio-temporal annotations to train the models. To facilitate action detection and localization, many algorithms use the action proposals [15, 25, 117] technique, which is an extension of object proposals for object detection in images.

There are only a few approaches based on weakly supervised learning that rely solely on video-level class labels to localize actions in temporal domain. UntrimmedNet [118] learns attention weights on precut video segments using a temporal softmax function and thresholds the attention weights to generate action proposals. The algorithm improves the video-level classification performance. However, only using class-agnostic attention weights for action proposal is clearly not optimal and the use of the softmax function across proposals may not be effective to detect multiple instances. Hide-and-seek [102] proposes a technique that randomly hides regions to force residual attention learning and thresholds class activation maps at inference time for weakly supervised spatial object detection and temporal action localization. While working well at spatial localization tasks, this method fails to show satisfactory performance in temporal action localization tasks. Both algorithms are motivated by the recent success of weakly supervised object localization in images. In particular, the formulation of UntrimmedNet for action localization heavily relies on the idea proposed in [11]. There are some other approaches [12, 46, 85] that learn to localize or segment actions in a

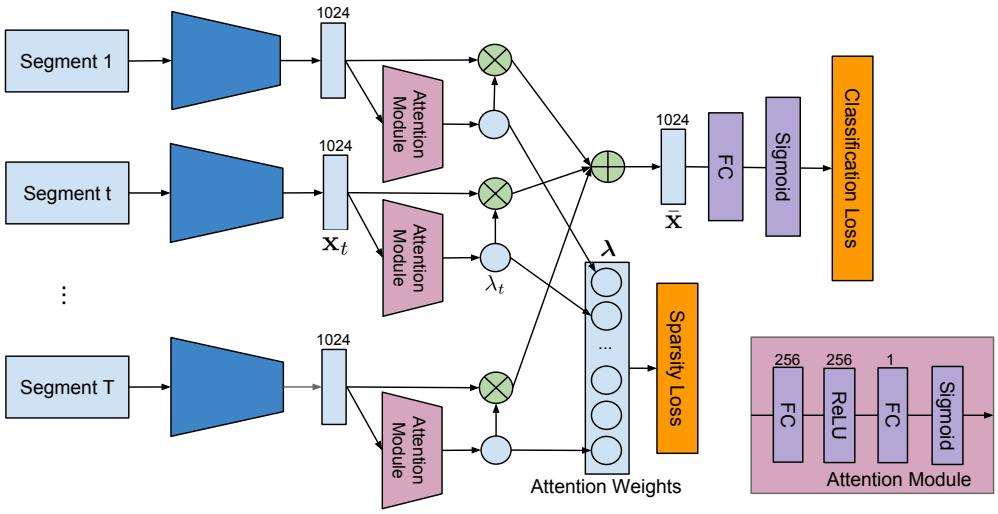


Figure 3.2: Network architecture for our weakly supervised temporal action localization model. We first extract feature representations for a set of uniformly sampled video segments using a pretrained network. The attention module computes class-agnostic attention weights for each segment, which are used to generate a video-level representation via weighted temporal average pooling. The representation is given to the classification module that can be trained with regular cross entropy loss with video-level labels. An ℓ_1 loss is placed on the attention weights to enforce sparse attentions.

weakly supervised setting by exploiting the temporal order of subactions during training. The main objective of these studies is to find the boundaries of sequentially presented subactions, while our approach aims to extract temporal intervals of full actions from input videos.

In terms of datasets, there are several publicly available datasets for action recognition including UCF101 [104], Sports-1M [53], HMDB51 [59], Kinetics [54] and AVA [38]. The videos in these datasets are trimmed so that the target actions appear throughout each clip. In contrast, THUMOS14 dataset [48] and ActivityNet [42] provide untrimmed videos that contain background frames and temporal annotations about which frames are relevant to the target actions. Note that each video in THUMOS14 and ActivityNet may have multiple actions happening in a single frame.

3.2 Proposed Algorithm

We claim that an action can be recognized from a video by identifying a set of key segments presenting important action components. So we design a neural network that learns to measure the importance of each segment in a video and automatically selects a sparse subset of representative segments to predict the video-level class labels. Only ground-truth video-level class labels are required for training the model. For action localization at inference time, we first identify relevant classes in each video and then generate temporal action proposals from temporal class activations and attentions to find the temporal locations of each relevant classes. The network architecture for our weakly supervised action recognition component is illustrated in Figure 3.2. We describe each step of our algorithm in the following sections.

3.2.1 Action Classification

To predict class labels in each video, we sample a set of segments and extract feature representations from each segment using pretrained convolutional neural networks. Each feature vector is then fed to an attention module that consists of two fully connected (FC) layers and a ReLU layer located between the two FC layers. The output of the second FC layer is given to a sigmoid function that ensures the generated attention weights are between 0 and 1. These class-agnostic attention weights are then used to modulate the temporal average pooling—a weighted sum of the feature vectors—to create a video-level representation. We pass this representation through an FC and sigmoid layers to obtain class scores.

Formally, let I_t represent a chunk of images (either RGB or flow) at centered timestamp t. We assume there exists a function Ψ that maps these frame or flow chunks into feature representation $x_t \in \mathbb{R}^d$. For the rest of this chapter, we assume a fixed Ψ during the training process, i.e. we freeze the weights in the layers implementation of Ψ. With a fixed

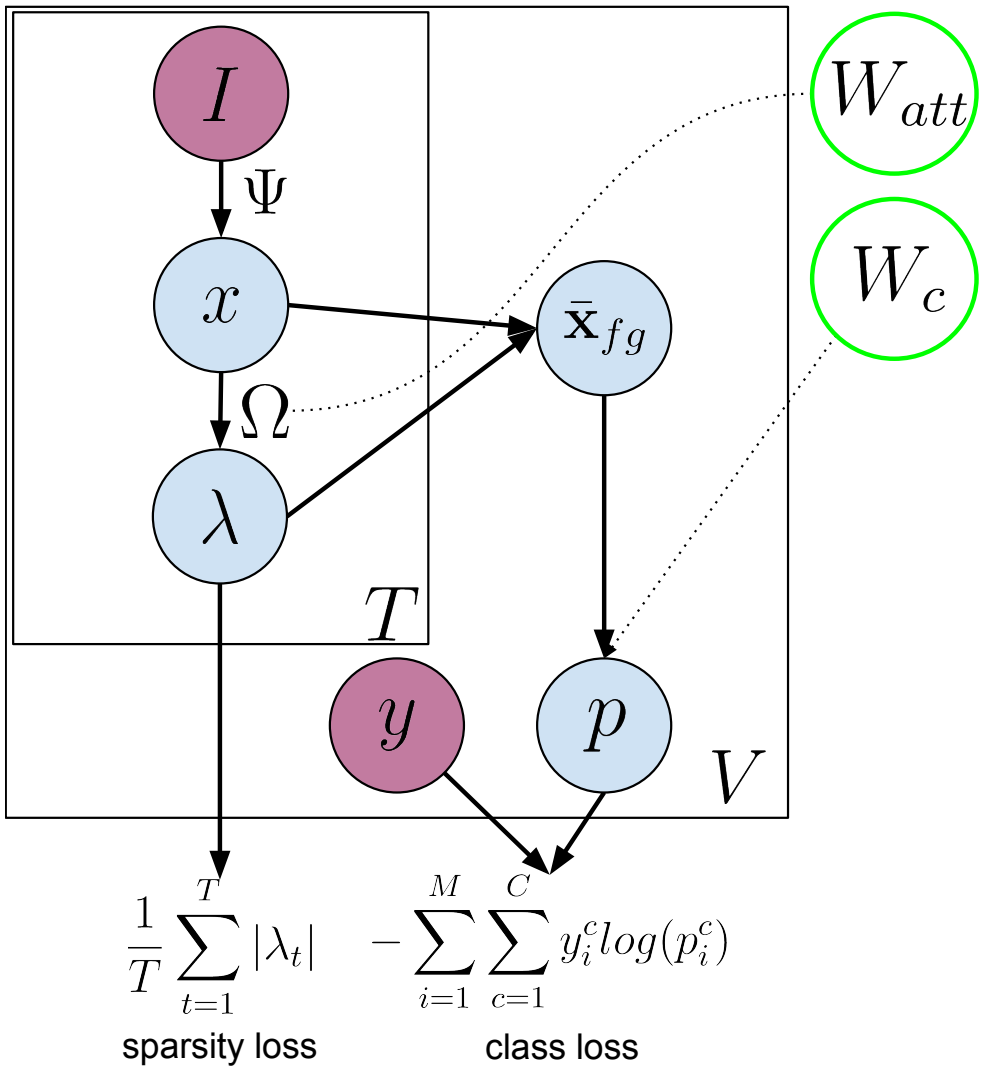


Figure 3.3: Our network is driven by two losses: a weakly supervised classification loss based on the video-level labels, y, and an unsupervised sparsity loss placed on the attention vectors, λ.

feature extractor, a dataset with C classes for weakly-supervised action localization is a set, $\left\{\{x_{i,t}\}_{t=1}^{T_i}, y_i\right\}_{i=1}^{M}$, containing M number of videos and each video contains T_i number of feature vectors $x_{i,t}$, and M number of video-level groundtruth label vectors, $y \in \{0,1\}^C$.

The attention function, Ω, converts this m-dimensional feature vector into a single real scalar, $\lambda_t = \Omega(x_t)$, between 0 and 1. This function is implemented using two fully-connected (FC) layers, ReLU layer located between the two FC layers, and a sigmoid function at the end. The video level representation, denoted by $\bar{\mathbf{x}}$, corresponds to an attention weighted temporal

average pooling, which is given by

$$\bar{\mathbf{x}} = \sum_{t=1}^{T} \lambda_t \mathbf{x}_t, \tag{3.1}$$

where $\lambda = (\lambda_1, \ldots, \lambda_T)^\top$ is a vector of scalar outputs from the attention module. The attention weight vector λ is defined in a class-agnostic way, which is useful to identify segments relevant to all the actions of interest for estimating the temporal intervals for action candidates later.

Feeding this foreground video-level feature representation to an FC layer, parameterized by $W_{class} \in \mathbb{R}^{C+1 \times D}$ and normalized to a probability using an activation function (sigmoid or softmax), we obtain

$$p = \sigma(W_{class}^T \bar{\mathbf{x}}), \tag{3.2}$$

where σ is an activation function of choice (softmax or sigmoid). The foreground classification loss is the defined via regular cross-entropy loss,

$$\mathcal{L}_{class} = -\sum_{i=1}^{M} \sum_{c}^{C} y_{i,c} \log(p_{i,c}^{fg}) \tag{3.3}$$

To enforce the fact that only a small fraction number of frames in the video should be active, we integrate a sparsity loss on attention weights,

$$\mathcal{L}_{sparsity} = \sum_{i}^{M} \frac{1}{T_i} \sum_{t}^{T_i} |\lambda_t|. \tag{3.4}$$

The loss function in the proposed network is composed of two terms, the classification loss and the sparsity loss, which is given by

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \beta \cdot \mathcal{L}_{\text{sparsity}}, \tag{3.5}$$

where β is a hyperparameter to control the trade-off between the two terms.

Because of the sigmoid function and the ℓ_1 loss, all the attention weights tend to have values close to either 0 or 1. Note that integrating the sparsity loss is aligned with our claim that an action can be recognized with a sparse subset of key segments in a video. Figure 3.3 summarizes our notations.

3.2.2 Temporal Class Activation Mapping

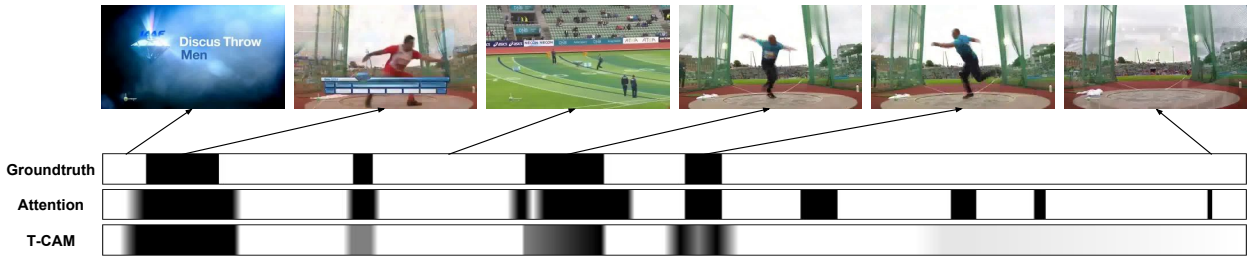


Figure 3.4: Illustration of the ground-truth temporal intervals for the *ThrowDiscus* class, the temporal attentions, and the T-CAM for an example video in the THUMOS14 dataset [48]. The horizontal axis in the plots denote the timestamps. In this example, the T-CAM values for *ThrowDiscus* provide accurate action localization information. Note that the temporal attention weights are large at several locations that do not correspond to the ground-truth annotations. This is because temporal attention weights are trained in a class-agnostic way.

To identify the time intervals corresponding to target actions, we extract a number of action interval candidates. Based on the idea in [141], we derive a one dimensional class-specific activation map in the temporal domain, referred to as the Temporal Class Activation Map (T-CAM). Let $\mathbf{w}^c(k)$ denote the k-th element in the weight parameter $\mathbf{w}^c$ of the final fully connected layer, where the superscript c represents the index of a particular class.

The input to the final sigmoid layer for class c is

$$\begin{aligned} s^c &= \sum_{k=1}^{m} \mathbf{w}^c(k)\bar{\mathbf{x}}(k) \\ &= \sum_{k=1}^{m} \mathbf{w}^c(k) \sum_{t=1}^{T} \lambda_t \mathbf{x}_t(k) \\ &= \sum_{t=1}^{T} \lambda_t \sum_{k=1}^{m} \mathbf{w}^c(k)\mathbf{x}_t(k). \end{aligned} \tag{3.6}$$

T-CAM, denoted by $\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^C)^\top$, indicates the relevance of the representations to each class at time step t, where each element a_t^c for class c ($c = 1, \ldots, C$) is given by

$$a_t^c = \sum_{k=1}^{m} \mathbf{w}^c(k)\mathbf{x}_t(k). \tag{3.7}$$

Figure 3.4 illustrates an example of the attention weights and the T-CAM outputs in a video given by the proposed algorithm. We can observe that the discriminative temporal regions are effectively highlighted by the attention weights and the T-CAMs. Also, some temporal intervals with large attention weights do not correspond to large T-CAM values because such intervals may represent other actions of interest. The attention weights measure the generic actionness of temporal video segments while the T-CAMs present class-specific information.

3.2.3 Two-stream CNN Models

We employ the recently proposed I3D model [16] to compute feature representations for the sampled video segments. Using multiple streams of information such as RGB and optical flow has become a standard practice in action recognition and detection [16, 32, 98] as it often provides a significant boost in performance. We also train two action recognition networks separately with identical settings as illustrated in Figure 3.2 for the RGB and the flow stream.

Note that our I3D networks are pretrained on the Kinetics dataset [54], and we only use it as a feature extraction machine without any fine-tuning on our target datasets. Our two-stream networks are then fused to localize actions in an input video. The procedure is discussed in the following subsection.

3.2.4 Temporal Action Localization

For an input video, we identify relevant class labels based on video-level classification scores (Section 3.2.1). For each relevant action, we generate temporal proposals, i.e. one-dimensional time intervals, with their class-specific confidence scores, corresponding to segments that potentially enclose the target actions.

To generate temporal proposals, we compute the T-CAMs for both the RGB and the flow streams, denoted by $a^c_{t,\text{RGB}}$ and $a^c_{t,\text{FLOW}}$ respectively, based on (3.7) and use them to derive the weighted T-CAMs, $\psi^c_{\text{t, RGB}}$ and $\psi^c_{\text{t, FLOW}}$ as

$$\psi^c_{t,\text{RGB}} = \lambda_{t,\text{RGB}} \cdot \text{sigmoid}(a^c_{t,\text{RGB}}) \tag{3.8}$$

$$\psi^c_{t,\text{FLOW}} = \lambda_{t,\text{FLOW}} \cdot \text{sigmoid}(a^c_{t,\text{FLOW}}). \tag{3.9}$$

Note that λ_t is an element of the sparse vector $\boldsymbol{\lambda}$, and multiplying λ_t can be interpreted as a soft selection of the values from the following sigmoid function. Similar to [141], we threshold the weighted T-CAMs, $\psi^c_{t,\text{RGB}}$ and $\psi^c_{t,\text{FLOW}}$ to segment these signals. The temporal proposals are then the one-dimensional connected components extracted from each stream. It is intuitive to generate action proposals using the weighted T-CAMs, instead of directly from the attention weights, because each proposal should contain a single kind of action. Optionally, we linearly interpolate the weighted T-CAM signals between sampled segments before thresholding to improve the temporal resolution of the proposals with minimal computation addition.

Unlike the original CAM-based bounding box proposals [141] where only the largest bounding box is retained, we keep all the connected components that pass the predefined threshold.

Each proposal $[t_{\text{start}}, t_{\text{end}}]$ is assigned a score for each class c, which is given by the weighted average T-CAM of all the frames within the proposal:

$$\sum_{t=t_{\text{start}}}^{t_{\text{end}}} \lambda_{t,*} \frac{\alpha \cdot a^c_{t,\text{RGB}} + (1-\alpha) \cdot a^c_{t,\text{FLOW}}}{t_{\text{end}} - t_{\text{start}} + 1}, \tag{3.10}$$

where $* \in \{\text{RGB}, \text{FLOW}\}$ and α is a parameter to control the magnitudes of the two modality signals. Finally, we perform non-maximum suppression among temporal proposals of each class independently to remove highly overlapped detections.

3.2.5 Discussion

Our algorithm attempts to localize actions in untrimmed videos temporally by estimating sparse attention weights and T-CAMs for generic and specific actions, respectively. The proposed method is principled and novel when compared to the existing UntrimmedNet [118] because of the following reasons.

- Our model has a unique architecture with classification and sparsity losses.
- Our action localization procedure is based on a completely different pipeline that leverages class-specific action proposals using T-CAMs.

Note that [118] follows a similar framework used in [11], where softmax functions are employed across both action classes and proposals; it has a critical limitation in handling multiple action classes and instances in a single video.

Similar to pretraining on the ImageNet dataset [23] for weakly supervised learning problems

in images, we utilize features from I3D models [16] pretrained on the Kinetics dataset [54] for video representation. Although the Kinetics dataset has considerable class overlap with our target datasets, its video clips are mostly short and contain only parts of actions, which makes their characteristics different from the ones in our untrimmed target datasets. We also do not fine-tune the I3D models and our network may not be optimized for the classes in the target tasks and datasets.

3.3 Experiments

This section describes the details of the benchmark datasets and the evaluation setup. Our algorithm, referred to as Sparse Temporal Pooling Network (STPN), is compared with other state-of-the-art techniques based on fully and weakly supervised learning. Finally, we analyze the contribution of individual components in our algorithm.

3.3.1 Datasets and Evaluation Method

We evaluate STPN on two popular action localization benchmark datasets, THUMOS14 [48] and ActivityNet1.3 [42]. Both datasets are untrimmed, meaning the videos include frames that contain no target actions and we do not exploit the temporal annotations for training. Note that there may exist multiple actions in a single video and even in a single frame in these datasets.

The THUMOS14 dataset has video-level annotations of 101 action classes in its training, validation, and testing sets, and temporal annotations for a subset of videos in the validation and testing sets for 20 classes. We train our model with the 20-class validation subset, which consists of 200 untrimmed videos, without using the temporal annotations. We evaluate our algorithm using the 212 videos in the 20-class testing subset with temporal annotations.

Table 3.1: Comparison of our algorithm with other recent techniques on the THUMOS14 testing set. We divide the algorithms into two groups depending on their levels of supervision. Each group is sorted chronologically, from older to newer ones. STPN, including the version using UntrimmedNet features, clearly presents state-of-the-art performance in the weakly supervised setting and is even competitive with many fully supervised approaches.

Supervision	Method	AP@IoU								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Fully supervised	Heilbron et al. [43]	–	–	–	–	13.5	–	–	–	–
	Richard et al. [84]	39.7	35.7	30.0	23.2	15.2	–	–	–	–
	Shou et al. [97]	47.7	43.5	36.3	28.7	19.0	10.3	5.3	–	–
	Yeung et al. [133]	48.9	44.0	36.0	26.4	17.1	–	–	–	–
	Yuan et al. [136]	51.4	42.6	33.6	26.1	18.8	–	–	–	–
	Escordia et al. [25]	–	–	–	–	13.9	–	–	–	–
	Shou et al. [95]	–	–	40.1	29.4	23.3	13.1	7.9	–	–
	Yuan et al.[137]	51.0	45.2	36.5	27.8	17.8	–	–	–	–
	Xu et al.[128]	54.5	51.5	44.8	35.6	28.9	–	–	–	–
	Zhao et al. [139]	66.0	59.4	51.9	41.0	29.8	–	–	–	–
	Alwasssel et al. [5]*	49.6	44.3	38.1	28.4	19.8	–	–	–	–
	Zhao et al. [140]	66.0	59.4	51.9	41.0	29.8	–	–	–	–
Weakly supervised	Wang et al [118]	44.4	37.7	28.2	21.1	13.7	–	–	–	–
	Singh & Lee [102]	36.4	27.8	19.5	12.7	6.8	–	–	–	–
	STPN	52.0	44.7	35.5	25.8	16.9	9.9	4.3	1.2	0.1
	STPN + UntrimmedNet features	45.3	38.8	31.1	23.5	16.2	9.8	5.1	2.0	0.3

This dataset is challenging as some videos are relatively long (up to 26 minutes) and contain multiple action instances. The length of an action varies significantly, from less than a second to minutes. The ActivityNet dataset is a recently introduced benchmark for action recognition and localization in untrimmed videos. We use ActivityNet1.3, which originally consisted of 10,024 videos for training, 4,926 for validation, and 5,044 for testing[1], with 200 activity classes. This dataset contains a large number of natural videos that involve various human activities under a semantic taxonomy.

We follow the standard evaluation protocol based on mean average precision (mAP) values at several different levels of intersection over union (IoU) thresholds. The evaluation of both the datasets is conducted using the benchmarking code for the temporal action localization task provided by ActivityNet[2]. The result on the ActivityNet1.3 testing set is obtained by

[1]In our experiments, there were 9740, 4791, and 4911 videos accessible from YouTube in the training, validation, and testing set respectively.

[2]`https://github.com/activitynet/ActivityNet/blob/master/Evaluation/`

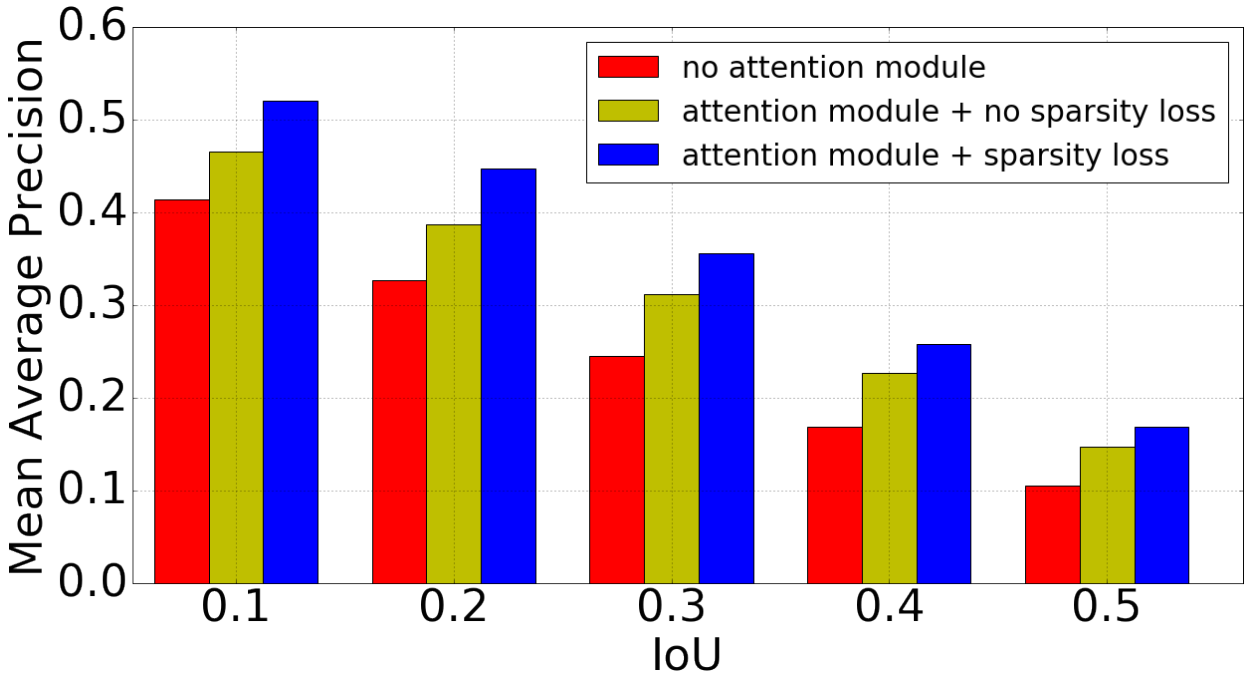


Figure 3.5: Performance with respect to architecture choices. The attention module is useful as it allows the model to explicitly focus on the important parts of input videos. Enforcing the sparsity in action recognition via ℓ_1 loss gives a significant boost in performance.

submitting results to the evaluation server.

3.3.2 Implementation Details

We use two-stream I3D networks [16] trained on the Kinetics dataset [54] to extract features for video segments. For the RGB stream, we rescale the smallest dimension of a frame to 256 and perform the center crop of size 224×224. For the flow stream, we apply the TV-$L1$ optical flow algorithm [123]. The inputs to the I3D models are stacks of 16 (RGB or flow) frames sampled at 10 frames per second.

We sample 400 segments at uniform interval from each video in both training and testing. During training, we perform stratified random perturbation on the segments sampled for data augmentation. The network is trained using Adam optimizer with learning rate 10^{-4}. At testing time, we first reject classes whose video-level probabilities are below 0.1, and then

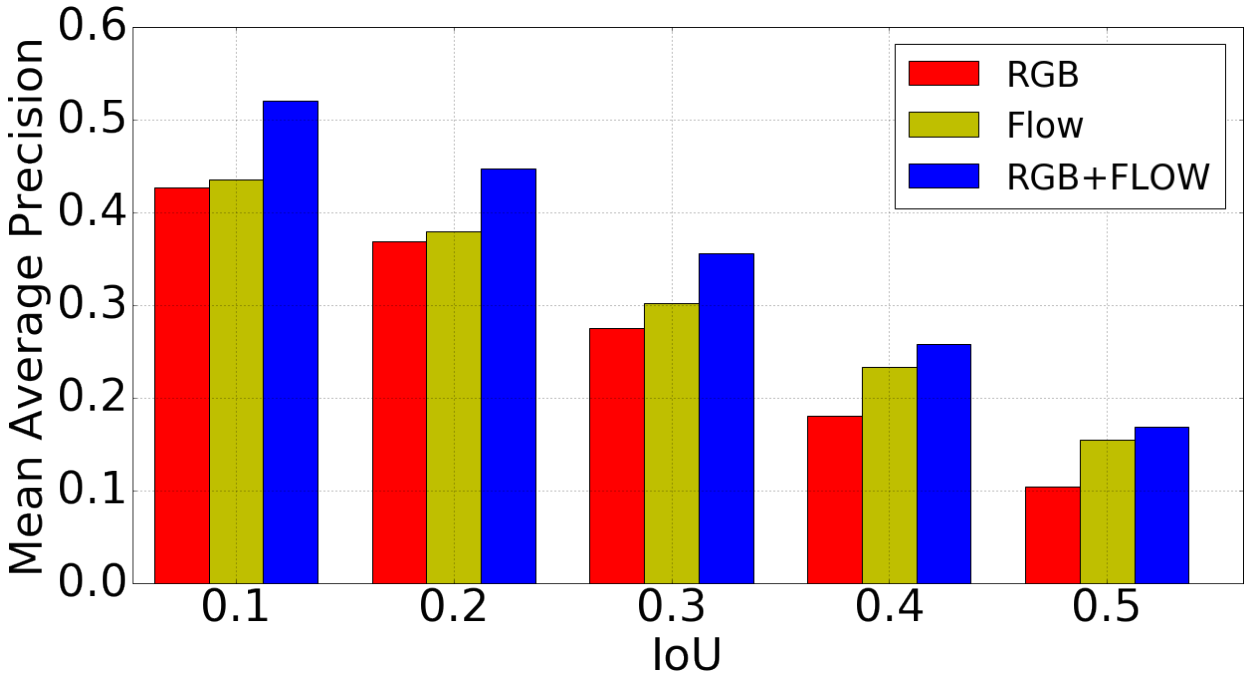


Figure 3.6: Performance with respect to feature choices. Optical flow offers stronger cues than RGB for action localization, and a combination of the two features leads to significant performance improvement.

retrieve one-dimensional temporal proposals for the remaining classes. We set the modality balance parameter α in (3.10) to 0.5. Our algorithm is implemented in TensorFlow.

3.3.3 Results

Table 3.1 summarizes the test results on THUMOS14 for action localization methods in the past two years. We included both fully and weakly supervised approaches in the table. Our algorithm outperforms the other two existing approaches based on weakly supervised learning [118, 102]. Even with significant difference in the level of supervision, our algorithm presents competitive performance to several recent fully supervised approaches. We also present performance of our model using the features extracted from the pretrained UntrimmedNet [118] two-stream models to evaluate the performance of our algorithm based on weakly supervised representation learning. For this experiment, we adjust α to 0.1 to

Table 3.2: Results on ActivityNet1.3 validation set. The methods with asterisk (*) report ActivityNet challenge results, may only be available in *arXiv* only, and are not comparable to our algorithm directly. Although [95] shows good accuracy, it is a post-processing result from [121], making comparison difficult.

	Method	AP@IoU		
		0.5	0.75	0.95
Fully supervised	Singh & Cuzzonlin [101]*	34.5	–	–
	Wang & Tao [121]*	45.1	4.1	0.0
	Shou et al. [95]*	45.3	26.0	0.2
	Xiong et al. [127]*	39.1	23.5	5.5
	Montes et al. [72]	22.5	–	–
	Xu et al. [128]	26.8	–	–
Weakly supervised	Ours	29.3	16.9	2.6

Table 3.3: Results on the ActivityNet1.3 testing set. The methods with asterisk (*) report ActivityNet challenge results only and are not comparable to our algorithm directly.

	Method	mAP
Fully supervised	Singh & Cuzzolin [101]*	17.83
	Wang & Tao [121]*	14.62
	Xiong et al. [127]*	26.05
	Singh et al. [100]	17.68
	Zhao et al. [139]	28.28
Weakly supervised	Ours	20.07

handle the heterogeneous signal magnitudes of the two modalities. From Table 3.1, we can see that STPN also outperforms the UntrimmedNet [118] and the Hide-and-Seek algorithm [102] in this setting.

We also present performance of our algorithm on the validation and the testing set of ActivityNet1.3 dataset in Table 3.2 and 3.3, respectively. We can see that our algorithm outperforms some fully supervised approaches on both the validation and the testing set. Note that most of the action localization results available on the leaderboard are specifically tuned for the ActivityNet Challenge, which may not be directly comparable with our algorithm. To our knowledge, this is the first attempt to evaluate weakly supervised action localization performance on this dataset, and we report the results as a baseline for future reference.

3.3.4 Ablation Study

We investigate the contribution of several components proposed in our weakly supervised architecture and implementation variations. All the experiments for our ablation study are performed on THUMOS14 dataset.

Choice of architectures Our premise is that an action can be recognized by a sparse subset of segments in a video. When we learn our action classification network, two loss terms—classification and sparsity losses—are employed. Our baseline is the architecture without the attention module and the sparsity loss, which share motivation with the architecture in [141]. We also test another baseline with the attention module but without sparsity loss. Figure 3.5 shows comparisons between our baselines and the full model. We observe that both the sparsity loss and the attention weighted pooling make substantial contributions to performance improvement.

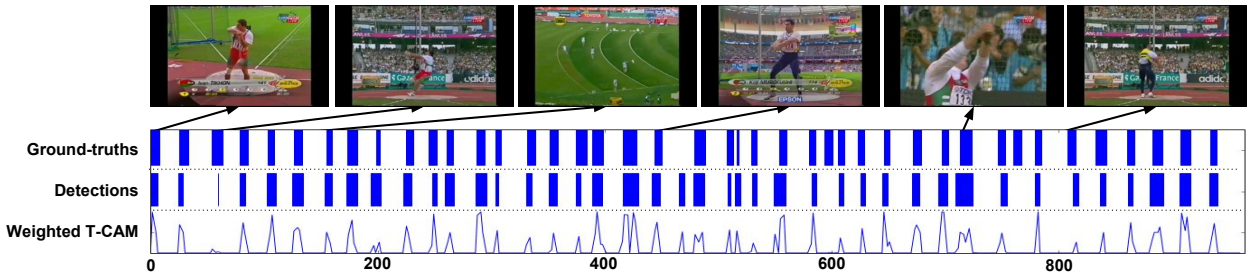
Choice of features The representation of each temporal segment is based on the two-stream I3D network, which employs two sources of information: one from the RGB image and the other from optical flow. Figure 3.6 illustrates the effectiveness of each modality and their combination. When comparing the individual performances of each modality, the flow stream offers stronger performance than the RGB steam. Similar to action recognition, the combination of these modalities provides significant performance improvement.

3.4 Conclusion

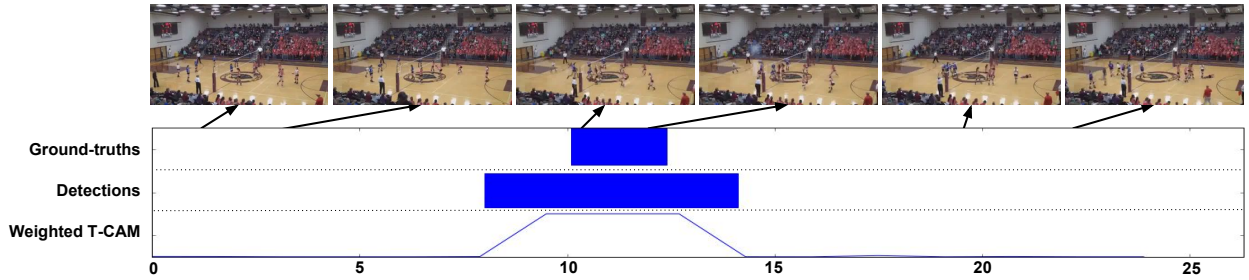
We presented a novel weakly supervised temporal action localization technique, which is based on deep neural networks with classification and sparsity losses. The classification is

performed by evaluating a video-level representation given by a sparsely weighted mean of segment-level features, where the sparse coefficients are learned with the sparsity loss in our deep neural network. For weakly supervised temporal action localization, one-dimensional action proposals are extracted, from which proposals relevant to target classes are selected to present time intervals of actions. The proposed approach achieved the state-of-the-art results in THUMOS14 dataset and, to the best of our knowledge, we are the first to report weakly supervised temporal action localization results on the ActivityNet1.3 dataset.

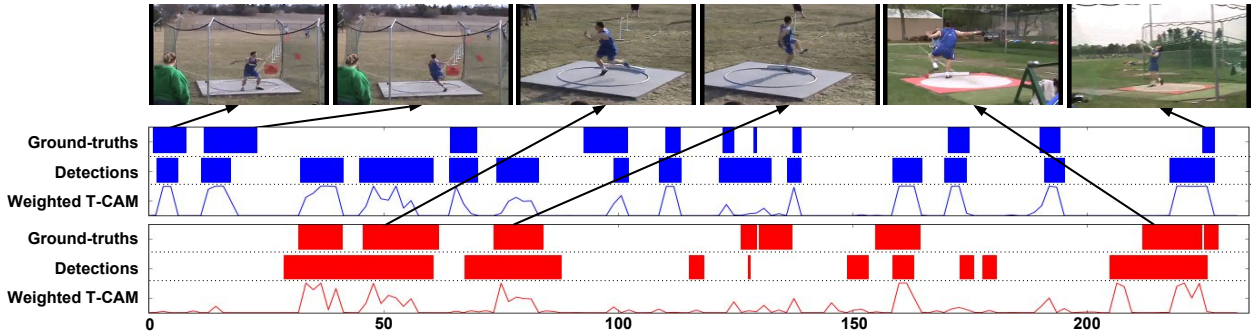
Acknowledgment We thank David Ross and Sudheendra Vijayanarasimhan at Google for providing the I3D features.



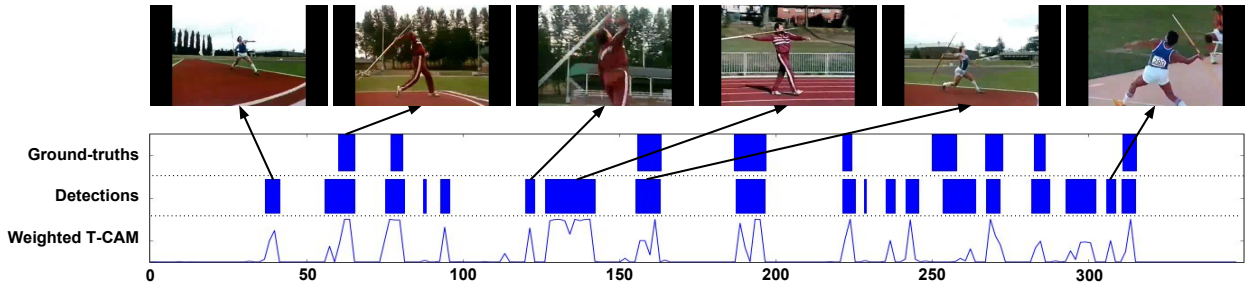
(a) An example of the *HammerThrow* action.



(b) An example of the *VolleyballSpiking* action.



(c) An example of the *ThrowDiscus* (blue) and *Shotput* (red) actions.



(d) An example of the *JavelinThrow* action.

s

Figure 3.7: Qualitative results on THUMOS14. The horizontal axis in the plots denote time index (in seconds). (a) There are many action instances in the input video and our algorithm shows good action localization performance. (b) The appearance of the video remains similar from the beginning to end. There is little motion between each frame. Our model is still be able to localize the small time window where the action actually happens. (c) Two different actions appear in a single video and their appearances along with motion patterns are similar. Even in the case, the proposed algorithm successfully identifies two actions accurately although there are some false alarms. (d) Our results have several false positives but they are often from missing ground-truth annotations. Another source of false alarms is the similarity of the observed actions to the target action.

Chapter 4

Improving Weakly Supervised Action Localization with Background Awareness

The previous chapter introduced an action localization system that leverages on weak supervision: only using video-level action labels instead of exact boundaries of action instances. While achieving state-of-the-art performance, the proposed system has three shortcomings. First, the system is unable to train until convergence due to the $l1$-loss on the attention vector. This loss in combination with video-level classification loss encourages the network to only keep a small number of frames that are very indicative of the video-level labels. As training progresses, the attention vectors will become overly sparse and, as a result, detection recalls worsen. This requires early stopping to maintain reasonable recalls. Second, there is an inherent lack of background modeling. A temporal localization model needs to not only differentiate between the foreground classes, but also between the backgrounds and foregrounds. However, the concept of background is never explicit in the architecture. Third, the temporal segment proposal process is done via the weighted Temporal Class Activation

Mappings (T-CAMs). T-CAMs can be quite spurious and incomplete because the trained network often focuses on the most discriminative parts of the action instead of the action as a whole. This phenomenon is also widely observed in the weakly supervised object detection literature [138].

This chapter extends on the existing system to improve the aforementioned shortcomings. We argue that in order to detect foreground actions better, we need to understand the background better. The concept of background modeling is not novel and has been an integral part of many fully-supervised object detection and action detection systems [83, 18]. However, it has been absent in weakly supervised detection systems. This may be due to lack of annotations for background class in the weak supervision settings. With full supervision, even though not explicitly annotated, background regions are easy to infer given the exact foreground boundaries. Without the foreground boundaries annotations, it is unclear how to model the background class with only video-level labels.

We incorporate background modeling via the attention vector, λ. The attention scalar, λ_t, is a class-agnostic weight designed to indicate how much the block of frames centered at timestamp t contributes to the video-level representation. Intuitively, regions with the high attention weight indicates high likelihood of actions, or foreground. The complements of attention weights, $1 - \lambda_t$, indicate frames that are uninteresting, i.e. the background. Pooling the frame features using background attention weights vector forms a background video-level representation. This augmented representation allows us to train a $C + 1$-class classifier, even without the explicit background annotations.

The second addition is a self-guided loss. Each T-CAM indicates the current's model belief on how much we should pay attention to a single frame feature for a single class. We can use the video-level labels to select T-CAMs and further condition the class-agnostic attention weights.

Thirdly, we further emphasize the difference between the background and foreground representation. We train a binary classifier that reinforces our current understanding of the foreground and background. This loss can also be loosely viewed as a clustering loss.

4.1 Algorithm

Figure 3.2 summarizes our algorithm. Let I_t represent a stack of consecutive frames (RGB or optical flow) centered at timestamp t. Let $\mathbf{x}_t = \Psi(I_t) \in \mathbb{R}^d$ be the d-dimensional feature representation extracted from I_t. Assume a fixed feature extraction, Ψ, a dataset for weakly supervised action localization is a set, $\left\{\{x_{i,t}\}_{t=1}^{T_i}, y_i\right\}_{i=1}^{M}$, containing M features-label pairs. Each pair contains T_i number of feature vector $x_{i,t}$, and a video-level groundtruth labels, $y \in \{0,1\}^{C+1}$. We use index 0 for the background class.

The attention function, Ω, converts this m-dimensional feature vector into a single scalar, $\lambda_t = \Omega(x_t)$, ranging between 0 and 1. This function is implemented using two fully-connected (FC) layers, ReLU layer located between the two FC layers, and a sigmoid function at the end. The video-level foreground representation, $\bar{\mathbf{x}}_{fg}$, corresponds to an attention-weighted temporal average pooling, which is given by

$$\bar{\mathbf{x}}_{fg} = \frac{1}{T}\sum_{t=1}^{T} \lambda_t \mathbf{x}_t. \tag{4.1}$$

Feeding this foreground video-level feature representation to an FC layer, parameterized by $W_{class} \in \mathbb{R}^{C+1\times D}$ and normalized to a probability using an activation function (sigmoid or softmax), we obtain

$$p_{fg} = \sigma(W_{class}^T \bar{\mathbf{x}}_{fg}). \tag{4.2}$$

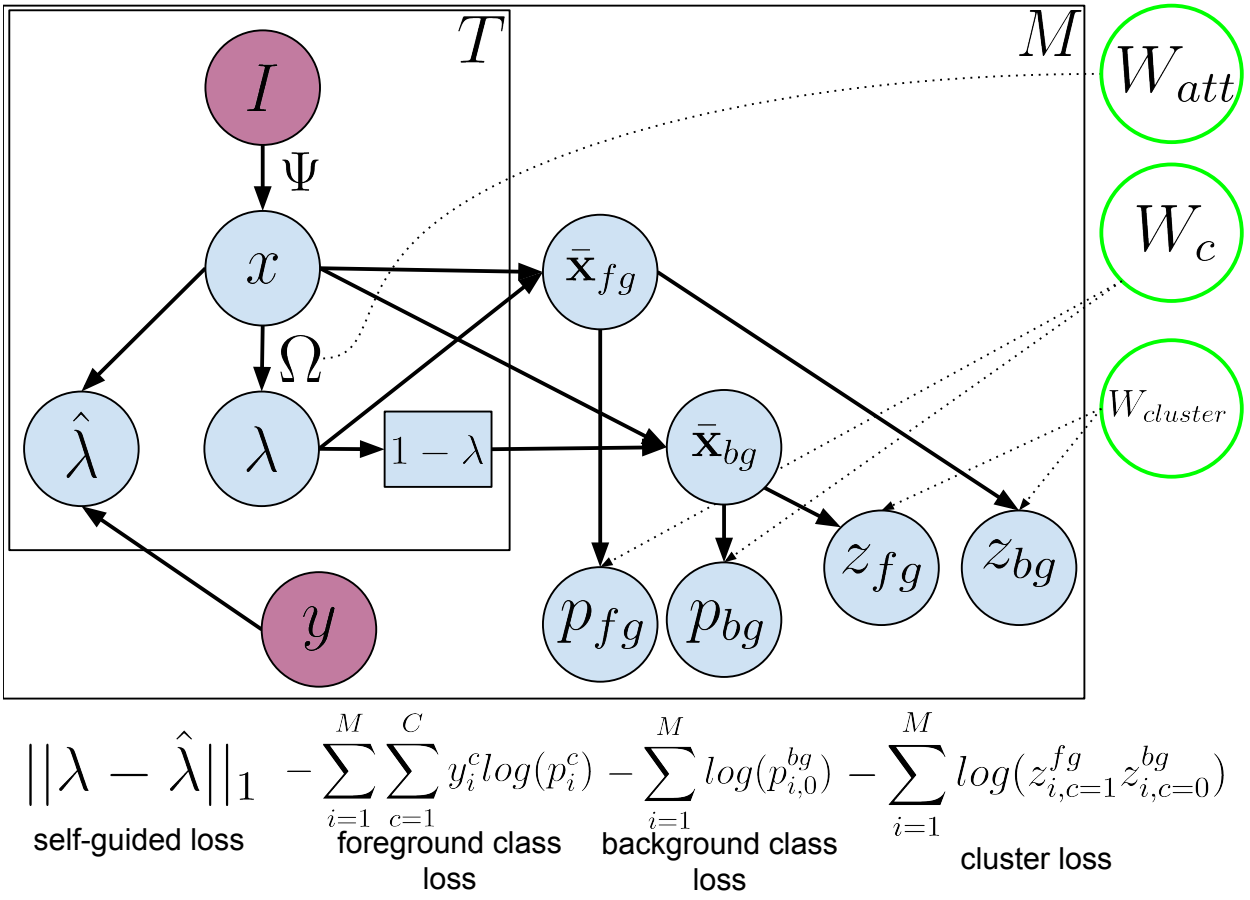


Figure 4.1: Using a pre-trained network, we extract the feature representation for a block of consecutive frames (RGB or optical flows), denoted by I. The attention module maps these segment-level features, x_t, into attention values, λ_t. These attention values can be used to pool the segment-level features into a single foreground video-level feature representation, $\bar{x}_{fg}$. The complements of the attention values, $1 - \lambda$, are used to pool the segment-level features into a single background video-level feature representation, $\bar{x}_{bg}$. The attention targets, $\hat{\lambda}$, are created from combining the T-CAMs responses. The foreground and background video-level representations are then trained with the usual cross entropy loss with video-level labels. The self-guided loss keeps the attentions closer to the attention targets.

The foreground classification loss is the defined via regular cross-entropy loss,

$$\mathcal{L}_{fg} = -\sum_{i=1}^{M}\sum_{c=1}^{C} y_{i,c} log(p_{i,c}^{fg}) \tag{4.3}$$

Background-awareness Loss The complement of the attention vector, $1-\lambda$, specifies timestamps where the current model believes that no action exists. Pooling frame features using these background attention weights creates a background video-level representation. Mathematically, this representation is defined as,

$$\bar{\mathbf{x}}_{bg} = \sum_{t=1}^{T}(1-\lambda_t)\mathbf{x}_t \tag{4.4}$$

$$p_{bg} = \sigma(W_{class}^{T}\bar{\mathbf{x}}_{bg}) \tag{4.5}$$

The background-awareness loss, L_{bg}, is,

$$\mathcal{L}_{bg} = -\sum_{i=1}^{M}\sum_{c=1}^{C} y_{i,c}^{bg} log(p_{i,c}^{bg}) \tag{4.6}$$

where $y_i^{bg} \in \{0,1\}^{C+1}$ is the background label vector with 1 at the background index and 0 everywhere else. This loss simplifies to,

$$\mathcal{L}_{bg} = -\sum_{i=1}^{M} log(p_{i,c=0}^{bg}) \tag{4.7}$$

Self-guided Loss When trained with only the foreground classification loss, attention weights tend to stay very close to 1. This suggests that the network prefers to use all the frames and perform the search for the best video-level solution by adjusting the weights of last fully-connected layers, W_{class}. The sparsity loss, proposed in STPN [73], forces the attention values to be sparse. This extra constraint causes the simultaneous search between W_{class}, and $W_{attention}$ to satisfy both losses. As a result, the attention weights are more diverse and

the attention module has to decide which frames are important. However, the sparsity loss in combination with video-level classification loss encourages the network to only keep a small number of frames that are very representative of the video-level labels. While this doesn't hurt the classification performance, detection performance worsens as recall becomes increasingly lower after certain iterations.

What if we keep the attention to some reasonable initial configuration and let the attention module explores the immediate space around this initial point? The T-CAMs, while are spurious and hard to generate proposals from, can still act as a reasonable initial points for attention configuration. We construct an attention target, or $\hat{\lambda}$, from the T-CAMs. Mathematically,

$$\hat{\lambda}(x_t) = G(\sigma) * \frac{1}{|V_c|} \sum_{c \in V_c} \sigma(w_c^T x_t)$$

where w_c is the weight of the last fully-connected layer corresponding to class c and V_c is the set of active classes in the current video. The summation is then convolved with a Gaussian kernel parameterized by a variance σ. The Gaussian smoothing combats the jaggedness of the Class Activation Mappings caused by the fact that weakly-supervised network often focuses on the most discriminative parts of the actions instead of the whole object/action. Intuitively, we are saying that if a frame has high probability of being an action, its neighboring frames should also high probability of containing an action. The self-guided loss, $\mathcal{L}_{guide}$, is defined as,

$$\mathcal{L}_{guide} = \frac{1}{T} \sum_t |\lambda_t - \hat{\lambda}_t| + |(1 - \lambda_t) - G(\sigma) * \sigma(w_0^T x_t)|.$$

The terms enforces differences between foreground and background attention correspondingly.

Foreground-background Clustering Loss We maintain another set of weights, $W_{cluster} \in R^{2 \times D}$, that maps a pooled video-level representation to two classes: foreground and back-

ground.

$$z^{fg} = \sigma(W^T_{cluster}\bar{\mathbf{x}}_{fg}) \tag{4.8}$$

$$z^{bg} = \sigma(W^T_{cluster}\bar{\mathbf{x}}_{bg}) \tag{4.9}$$

where $z^{fq}, z^{bq} \in R^2$ are two binary vectors.

The sum of two binary classification losses using the above probability vectors is,

$$\mathcal{L}_{cluster} = -\sum_{i=1}^{M} log(z^{fg}_{i,c=1} z^{bg}_{i,c=0}) \tag{4.10}$$

The total training loss is

$$\mathcal{L}_{total} = \mathcal{L}_{fg} + \alpha\mathcal{L}_{bg} + \beta\mathcal{L}_{guide} + \gamma\mathcal{L}_{cluster}.$$

with α, β and γ are the hyperparameters to control the corresponding weights between the losses. We find that these hyperparameters (α, β, γ) need to be small enough so that network is driven mostly by the foreground loss, $\mathcal{L}_{fg}$. We set these values, $\alpha = \beta = \gamma = 0.1$.

4.1.1 Action Localization

To generate action proposals and detections, we first identify relevant action classes based on video-level classification probabilities, p^{fg}. Segment proposals are then generated for each relevant class. These proposals are then scored with the corresponding weighted T-CAMs to obtain the final detections.

A proposal $[t_{start}, t_{end}, c]$, is scored as

$$\sum_{t_{\text{start}}}^{t_{\text{end}}} \frac{\theta \lambda_t^{RGB} w_c^T x_t^{RGB} + (1-\theta)\lambda_t^{FLOW} w_c^T x_t^{FLOW}}{t_{\text{end}} - t_{\text{start}} + 1} \tag{4.11}$$

where θ is a scalar denoting the relative importance between the modalities. In this work, we set $\theta = 0.5$.

Unlike previous chapter, we do not generate proposals using attention-weighted T-CAMs but from the attention weights vector, λ. We threshold the attention vector and perform one-dimensional connected components to generate proposals. Multiple thresholds are used to improve the boundary locations. We find that generating proposals from the averaged attention weights from different modalities leads to more reliable proposals.

Figure 4.2 shows an example of the inference process.

4.2 Experiments

We evaluate the proposed algorithm under the same settings as previous chapter: on THUMOS14 action localization dataset and mean Average Precision at diffnerent levels of intersection over union (IoU) thresholds.

4.2.1 Implementation Details

We use I3D networks trained on Kinetics [54] as a fixed feature extraction for a block of consecutive frames. I3D features were extracted using the publicly-available codes and model[1].

[1] https://github.com/deepmind/kinetics-i3d

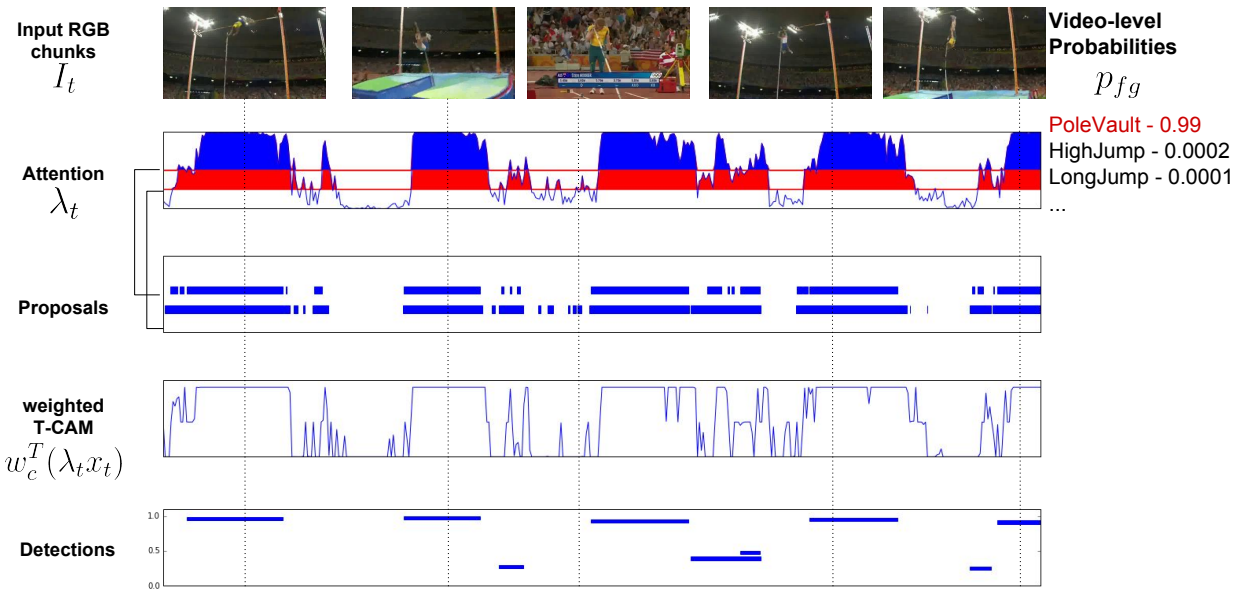


Figure 4.2: The detection process involves 3 steps: video-level classification probability thresholding, segment proposals and detections. First, relevant classes are selected using video-level probabilities. We threshold the attention vector and perform connected components to generate proposals. Multiple thresholds are used to obtain better segment boundaries. The proposals are then scored by the summation of the weighted-TCAM values fall into the interval. Non-maxima suppresion is done per-class to produce the final detection results. The y-axis in last figure indicates the detection score.

We follow the preprocessing steps for RGB and optical flows recommended by the software. For the flow stream, we use an OpenCV implementation to calculate the dense optical flow using the Gunnar Farneback's algorithm[2]. Rather than sampling a fixed number of segments per video and process multiple videos per batch, we load all available segments and process only one video per batch. The network is trained using the Adam optimizer with learning rate 10^{-4}. At test time, we reject classes with video-level probabilities below 0.1. If no foreground class has probability great than 0.1, we generate proposals and detections for the highest foreground class. We use values between 0.01 to 0.5 with 0.05 step for proposal thresholds. Our algorithm is implemented in Tensorflow.

[2]https://github.com/wanglimin/dense_flow

Table 4.1: The addition of each loss improve the localization performance. The improvements are also complement of each other as combining these losses achieves the best results. The second is copied from STPN [73] for reference.

					AP@IoU								
l_{fg}	l_{bg}	l_{guided}	$l_{cluster}$	l_{sparse}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
✓	–	–	–	–	46.6	38.7	31.2	22.6	14.7	–	–	–	–
✓	–	–	–	✓	52.0	44.7	35.5	25.8	16.9	9.9	4.3	1.2	0.2
✓	–	✓	–	–	53.8	46.4	38.2	29.0	19.2	10.6	4.4	1.3	0.1
✓	✓	–	–	–	53.6	47.6	39.1	30.2	20.5	12.2	5.4	1.7	0.2
✓	✓	–	✓	–	54.9	48.4	40.8	32.4	23.1	14.2	7.4	2.5	0.3
✓	–	✓	✓	–	60.1	54.1	45.6	34.0	23.2	13.6	6.2	1.4	0.1
✓	✓	✓	✓	–	60.7	56.0	46.0	36.6	27.1	17.6	9.0	3.3	0.5

4.3 Results

Ablation Studies Table 4.1 summarizes the ablation studies. The results demonstrate the effectiveness of each proposed loss. Each addition brings significant boost in localization results. Furthermore, the results also suggest that these losses are complement to each other as the combination gives compelling improvement.

Table 4.2 compare our action localization results on THUMOS14 against other weakly-supervised and fully-supervised systems in the last three years. We improve 10% mAP for IoUs less than 0.5 over the original STPN model. We also significantly outperform recent state-of-the-art weakly supervised action localization systems [96, 76]. Our models are also comparable to other fully-supervised systems, especially in the lower IoUs. At higher IoU, performance of our model tend to drop off quickly. To achieve strong results in high IoU, segment detections need to have high degree with ground truth. Here, knowledge of exact boundaries information allows models with full supervision to attain more accurate boundaries prediction.

Figure 4.3 compares the intermediate outputs (attentions, and weighted T-CAMs) and detections between STPN and our models. The attentions and weighted T-CAMs from our versions are better at discerning between the background and foreground classes, which leads

Table 4.2: Comparison of our algorithm with other recent techniques tested on THUMOS14. Our extensions gives 10% improvement over the original system [73]. We significantly outperform recent weakly supervised approaches [96, 76], 5% mAP@0.5. Our method is also comparable to fully-supervised methods, especially in lower IoU regimes. Good performance in higher IoU requires more accurate action boundary decisions, which is difficult to obtain without the actual boundary supervisions.

Supervision	Method	AP@IoU								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Fully supervised	Heilbron et al. [43]	–	–	–	–	13.5	–	–	–	–
	Richard et al. [84]	39.7	35.7	30.0	23.2	15.2	–	–	–	–
	Shou et al. [97]	47.7	43.5	36.3	28.7	19.0	10.3	5.3	–	–
	Yeung et al. [133]	48.9	44.0	36.0	26.4	17.1	–	–	–	–
	Yuan et al. [136]	51.4	42.6	33.6	26.1	18.8	–	–	–	–
	Escordia et al. [25]	–	–	–	–	13.9	–	–	–	–
	Shou et al. [95]	–	–	40.1	29.4	23.3	13.1	7.9	–	–
	Yuan et al.[137]	51.0	45.2	36.5	27.8	17.8	–	–	–	–
	Xu et al.[128]	54.5	51.5	44.8	35.6	28.9	–	–	–	–
	Zhao et al. [139]	66.0	59.4	51.9	41.0	29.8	–	–	–	–
	Alwasssel et al. [5]*	49.6	44.3	38.1	28.4	19.8	–	–	–	–
	Zhao et al. [140]	66.0	59.4	51.9	41.0	29.8	–	–	–	–
	Chao et al. [18]	59.8	57.1	53.2	48.5	42.8	33.8	20.8	–	–
	Alwassel et al. [4]	–	–	51.8	42.4	30.8	20.2	11.1	–	–
Weakly supervised	Wang et al [118]	44.4	37.7	28.2	21.1	13.7	–	–	–	–
	Singh & Lee [102]	36.4	27.8	19.5	12.7	6.8	–	–	–	–
	Nguyen et al.[73]	52.0	44.7	35.5	25.8	16.9	9.9	4.3	1.2	0.2
	Paul et al.[76]	55.2	49.6	40.1	31.1	22.8	–	7.6	–	–
	Shou et al.[96]	–	–	35.8	29.0	21.2	13.4	5.8	–	–
	Ours	61.0	56.0	46.6	37.5	27.2	17.6	9.0	3.3	0.5

to better detection results.

4.4 Conclusion

We extend on a state-of-the-art weakly supervised action localization system by incorporating background modeling. Based on the existing attention mechanism, we add three new losses: (1) a background classification loss based on the generated background representation, (2) a self-guided loss to guide the attention module using the combined outputs of the T-CAMS, and (3) a clustering loss that groups the representations into background or foreground class.

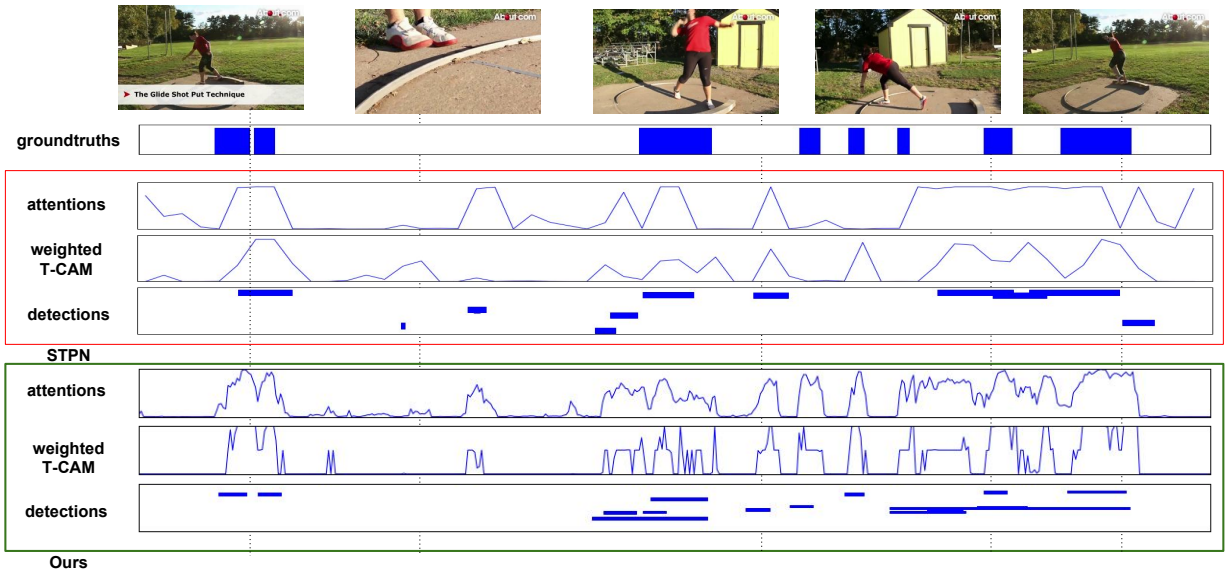


Figure 4.3: Compared to STPN [73], our model's attention responses are better able to pinpoint locations of background frames due to background modeling. The example is '*video_test*_001268'.

The new system outperforms the origial system by 10%mAP@IoU=0.5. We are also able to outperform more recent state-of-the-art weakly supervised action localization systems and maintain a comparable results with other fully-supervised systems.

Chapter 5

Active Testing: An Efficient and Robust Framework for Estimating Accuracy

Previous chapters address the lack of human annotations during training. But, what about testing? We also need a lot of annotated data for testing. How do we guarantee the performance of visual recognition systems with limited annotations? What happened when the testbed is too small? What about a larger, but noisier testbed? What about the "open world" scenario where visual appearance follows a long-tail distribution?

Visual recognition is undergoing a period of transformative progress, due in large part to the success of deep architectures trained on massive datasets with supervision. While visual data is in ready supply, high-quality supervised labels are not. One attractive solution is the exploration of unsupervised learning. However, regardless how they are trained, one still needs to *evaluate* accuracy of the resulting systems. Given the importance of rigorous, empirical benchmarking, it appears impossible to avoid the costs of assembling high-quality,

human-annotated test data for test evaluation.

Unfortunately, manually annotating ground-truth for large-scale test datasets is often prohibitively expensive, particularly for rich annotations required to evaluate object detection and segmentation. Even simple image tag annotations pose an incredible cost at scale [1]. In contrast, obtaining noisy or partial annotations is often far cheaper or even free. For example, numerous social media platforms produce image and video data that are dynamically annotated with user-provided tags (Flickr, Vine, Snapchat, Facebook, YouTube). While much work has explored the use of such massively-large "webly-supervised" data sources for learning [125, 135, 63, 112], we instead focus on them for evaluation.

How can we exploit such partial or noisy labels during testing? With a limited budget for vetting noisy ground-truth labels, one may be tempted to simply evaluate performance on a small set of clean data, or alternately just trust the cheap-but-noisy labels on the whole dataset. However, such approaches can easily give an inaccurate impression of system performance. We show in our experiments that these naive approaches can produce alarmingly-incorrect estimates of comparative model performance. Even with a significant fraction of vetted data, naive performance estimates can incorrectly rank two algorithms in 15% of trials, while our active testing approach significantly reduces this misranking error to 3%.

The problem of label noise even exists for "expertly" annotated datasets, whose construction involves manual selection of a test set which is deemed representative in combination with crowd-sourced labeling by multiple experts [81, 55]. Preserving annotation quality is an area of intense research within the HCI/crowdsourcing community [51, 93]. In practice, annotation errors are often corrected incrementally through multiple rounds of interactive error discovery and visual inspection of algorithm test results over the lifetime of the dataset. For example, in evaluating object detectors, the careful examination of detector errors on the test set [45]

[1] For example, NUS-WIDE, [20] estimated 3000 man-hours to semi-manually annotate a relatively small set of 81 concepts across 270K images

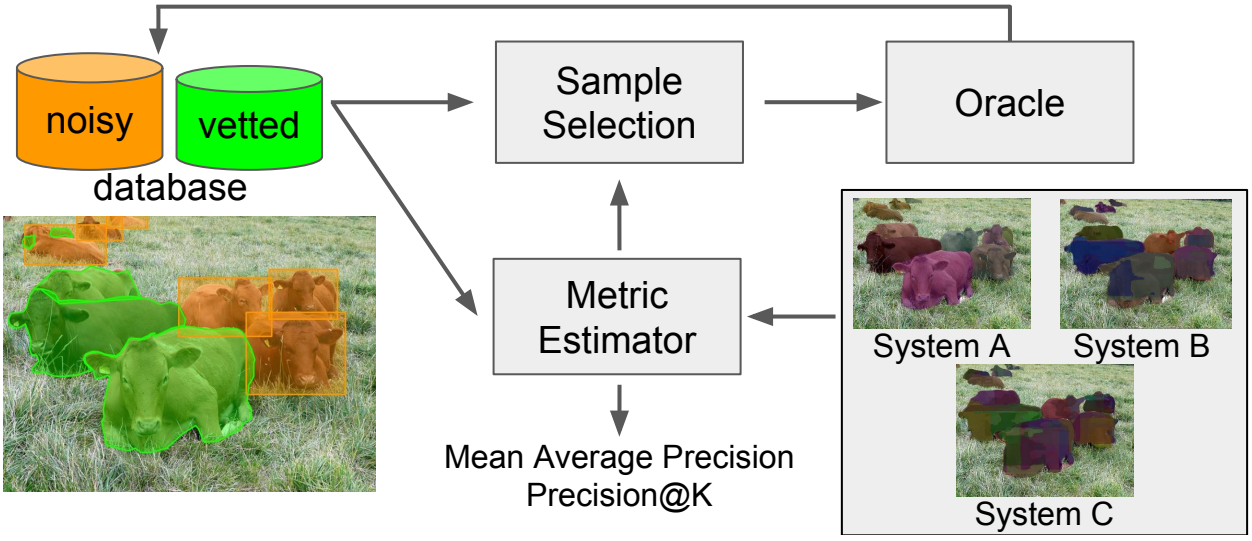


Figure 5.1: Classic methods for benchmarking algorithm performance require a test-set with high-quality labels. While it is often easy to obtain large-scale data with noisy labels, test evaluation is typically carried out on only a small fraction of the data that has been manually cleaned-up (or "vetted"). We show that one can obtain dramatically more accurate estimates of performance by using the vetted-set to train a statistical estimator that both (1) reports improved estimates and (2) actively selects the next batch of test data to vet. We demonstrate that such an "active-testing" process can efficiently benchmark performance and and rank visual recognition algorithms.

often reveals missing annotations in widely-used benchmarks [65, 26, 24] and may in turn invoke further iterations of manual corrections (e.g., [70]). In this work, we formalize such ad-hoc practices in a framework we term *active testing*, and show that significantly improved estimates of accuracy can be made through simple statistical models and active annotation strategies.

5.1 Related Works

Benchmarking: Empirical benchmarking is now widely considered to be an integral tool in the development of vision and learning algorithms. Rigorous evaluation, often in terms of challenge competitions [86, 27] on held-out data, serves to formally codify proxies for scientific or application goals and provides quantitative ways to characterize progress towards

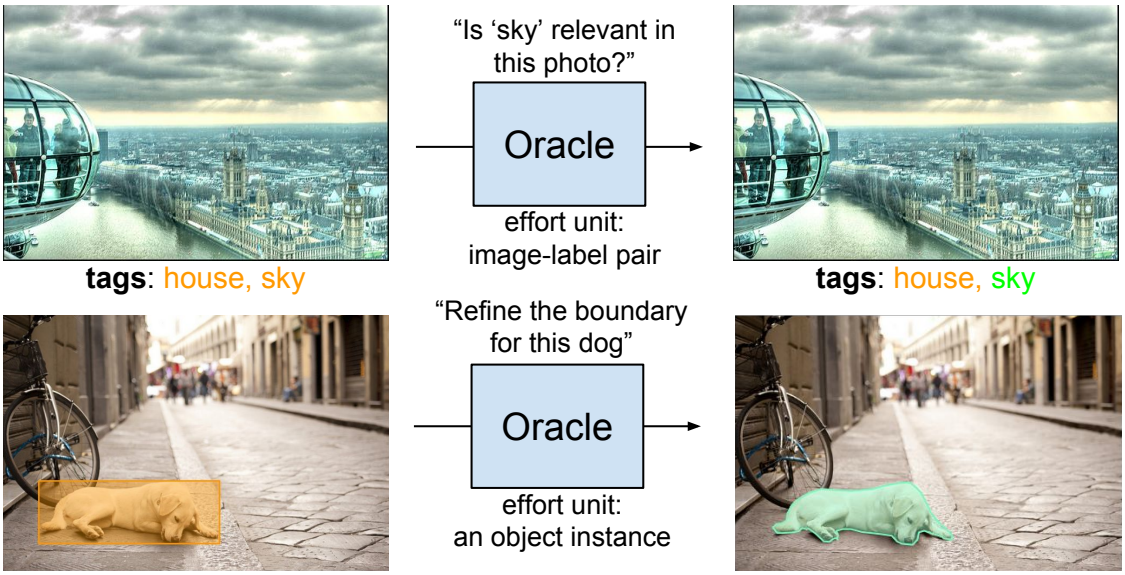


Figure 5.2: Vetting Procedure. The figure shows the vetting procedures for the multi-label classification (top) and instance segmentation (bottom) tasks. The annotations on the left are often incomplete and noisy, but significantly easier to obtain. These initial noisy annotations are "vetted" and corrected if necessary by a human. We quantify human effort in units of the number of image-label pairs corrected or object segment masks specified.

them. The importance and difficulties of test dataset construction and annotation are now readily appreciated [79, 107].

Benchmark evaluation can be framed in terms of the well-known *empirical risk* minimization approach to learning [111]. Benchmarking seeks to estimate the risk, defined as the expected loss of an algorithm under the true data distribution. Since the true distribution is unknown, the expected risk is estimated by computing loss a finite sized sample test set. Traditional losses (such as 0-1 error) decompose over test examples, but we are often interested in multivariate ranking-based metrics that do not decompose (such as Precision@K and Average Precision [49]). Defining and estimating expected risk for such metrics is more involved (e.g., Precision@K should be replaced by precision at a specified quantile [13]) but generalization bounds are known [3, 44]. For simplicity, we focus on the problem of estimating the empirical risk on a fixed, large but finite test set.

Semi-supervised testing: To our knowledge, there have only been a handful of works

specifically studying the problem of estimating recognition performance on partially labeled test data. Anirudh et al. [6] study the problem of 'test-driving' a detector to allow the users to get a quick sense of the generalizability of the system. Closer to our approach is that of Welinder et al. [124], who estimate the performance curves using a generative model for the classifier's confidence scores. Their approach leverages ideas from the semi-supervised learning literature while our approach builds on active learning.

The problem of estimating benchmark performance from sampled relevance labels has been explored more extensively in the information retrieval literature where complete annotation was acknowledged as infeasible. Initial work focused on deriving labeling strategies that produce low-variance and unbiased estimates [134, 8] and identifying performant retrieval systems [71]. [89] give error bounds for estimating PR and ROC curves by choosing samples to label based on the system output ranking. [33] estimate performance using an EM algorithm to integrate relevance judgements. [62] and [80] take a strategy similar to ours in actively selecting test items to label as well as estimating performance on remaining unlabeled data.

Active learning: Our proposed formulation of active testing is closely related to active learning. From a theoretical perspective, active learning can provide strong guarantees of efficiency under certain restrictions [9]. Human-in-the-loop active learning approaches have been well explored for addressing training data collection in visual recognition systems [14, 114, 113]. One can view *active testing* as a form of active learning where the actively-trained model is a statistical predictor of performance on a test set. Active learning is typically cast within the standard machine-learning paradigm, where the goal is to (interactively) learn a model that makes accurate per-example predictions on held-out *i.i.d* data. In this case, generalization is of paramount importance. On the other hand, active-testing interactively learns a model that makes *aggregate* statistical predictions over a *fixed* dataset. This means that models learned for active-testing (that say, predict average precision) need not generalize beyond the test set of interest. This suggests that one can be much more aggressive in

overfitting to the statistics of the data at hand.

5.2 Framework for Active Testing

In this section, we introduce the general framework for active testing. Figure 5.1 depicts the overall flow of our approach. Our evaluation database initially contains test examples with inaccurate (noisy) annotations. We select a batch of data items whose labels will be manually vetted by an oracle (e.g., in-house annotators or a crowd-sourced platform such as Mechanical Turk). Figure 5.2 shows examples of such noisy labels and queries to Oracle. The evaluation database is then updated with these vetted labels to improve estimates of test performance. Active testing consists of two key components: a *metric estimator* that estimates model performance from test data with a mix of noisy and vetted labels, and a *vetting strategy* which selects the subset of test data to be labeled in order to achieve the best possible estimate of the true performance.

5.2.1 Performance Metric Estimators

We first consider active testing for a simple binary prediction problem and then extend this idea to more complex benchmarking tasks such as multi-label tag prediction and instance segmentation. As a running example, assume that we are evaluating an system that classifies an image (e.g., as containing a cat or not). The system returns of confidence score for each test example $i \in \{1 \dots N\}$. Let y_i denote a "noisy" binary label for example i (specifying if a cat is present), where the noise could arise from labeling the test set using some weak-but-cheap annotation technique (e.g., user-provided tags, search engine results, or approximate annotations). Finally, let z_i be the true latent binary label whose value can be obtained by rigorous human inspection of the test data item.

Typical benchmark performance metrics can be written as a function of the true ground-truth labels and system confidences. We focus on metrics that only depend on the rank ordering of the confidence scores and denote such a metric generically as $Q(\{z_i\})$ where for simplicity we hide the dependence on s by assuming that the indices are always *sorted* according to s_i so that $s_1 \geq \cdots \geq s_N$. For example, commonly-used metrics for binary labeling include precision@K and average precision (AP):

$$Prec@K(\{z_1, \ldots, z_N\}) = \frac{1}{K} \sum_{i \leq K} z_i \tag{5.1}$$

$$AP(\{z_1, \ldots, z_N\}) = \frac{1}{N_p} \sum_{k} \frac{z_k}{k} \sum_{i \leq k} z_i \tag{5.2}$$

where N_p is the number of positives. We include derivations in supplmental material.

Estimation with partially vetted data: In practice, not all the data in our test set will be vetted. Let us divide the test set into two components, the unvetted set U for which we only know the approximate noisy labels y_i and the vetted set V, for which we know the ground-truth label. With a slight abuse of notation, we henceforth treat the true label z_i as a random variable, and denote its observed realization (on the vetted set) as $\tilde{z}_i$. The simplest strategy for estimating the true performance is to ignore unvetted data and only measure performance Q on the vetted subset:

$$Q(\{\tilde{z}_i : i \in V\}) \qquad \textbf{[Vetted Only]} \tag{5.3}$$

This represents the traditional approach to empirical evaluation in which we collect a single, vetted test dataset and ignore other available test data. This has the advantage that it is unbiased and converges to the true empirical performance as the whole dataset is vetted. The limitation is that it makes use of only fully-vetted data and the variance in the estimate can be quite large when the vetting budget is limited.

A natural alternative is to incorporate the unvetted examples by simply substituting y_i as a "best guess" of the true z_i. We specify this *naive* assumption in terms of a distribution over all labels $z = \{z_1, \ldots, z_N\}$:

$$p_{naive}(z) = \prod_{i \in U} \delta(z_i = y_i) \prod_{i \in V} \delta(z_i = \tilde{z}_i) \tag{5.4}$$

where $\tilde{z}_i$ is the label assigned during vetting. Under this assumption we can then compute an expected benchmark performance:

$$E_{p_{naive}(z)}\Big[Q(z)\Big] \qquad \textbf{[Naive Estimator]} \tag{5.5}$$

which amounts to simply substituting $\tilde{z}_i$ for vetted examples and y_i for unvetted examples.

Unfortunately, the above performance estimate may be greatly affected by noise in the nosiy labels y_i. For example, if there are systematic biases in the y_i, the performance estimate will similarly be biased. We also consider more general scenarios where side information such as features of the test items and distribution of scores of the classifier under test may also be informative. We thus propose computing the expected performance under a more sophisticated estimator:

$$p_{est}(z) = \prod_{i \in U} p(z_i | \mathcal{O}) \prod_{i \in V} \delta(z_i = \tilde{z}_i) \tag{5.6}$$

where $\mathcal{O}$ is the total set of all observations available to the benchmark system (e.g. noisy labels, vetted labels, classifier scores, data features). We make the plausible assumption that the distribution of unvetted labels factors conditioned on $\mathcal{O}$.

Our proposed active testing framework (see Alg 1) estimates this distribution $p_{est}(z)$ based on

Algorithm 1 Active Testing Algorithm

Input: unvetted set U, vetted set V, total budget T, vetting strategy VS, system scores $S = \{s_i\}$, estimator $p_{est}(z)$
while $T \geq 0$ **do**
  Select a batch $B \subseteq U$ according to vetting strategy VS.
  Query oracle to vet B and obtain true annotations $\tilde{z}$.
  $U = U \setminus B$, $V = V \cup B$
  $T = T - |B|$
  Fit estimator p_{est} using U, V, S.
end while
Estimate performance using $p_{est}(z)$

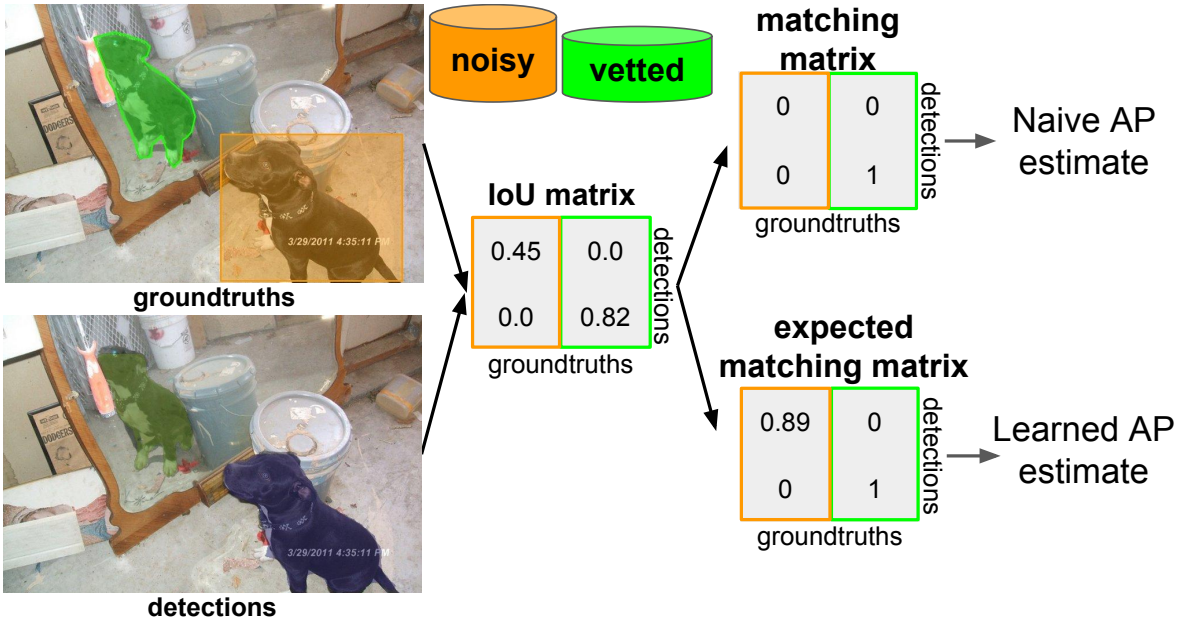


Figure 5.3: Standard instance segmentation benchmarks ignore unvetted data (top pathway) when computing Average Precision. Our proposed estimator for this task computes an expected probability of a match for coarse bounding box annotations when vetted instance masks aren't available.

available observations and predicts expected benchmark performance under this distribution:

$$E_{p_{est}(z)}\Big[Q(z)\Big] \qquad \textbf{[Learned Estimator]} \tag{5.7}$$

Computing expected performance: Given posterior estimates $p(z_i|\mathcal{O})$ we can always compute the expected performance metric Q by generating samples from these distributions, computing the metric for each joint sample, and average over samples. Here we introduce

two applications (studied in our experiments) where the metric is linear or quadratic in z, allowing us to compute the expected performance in closed-form.

Multi-label Tags: Multi-label tag prediction is a common task in video/image retrieval. Following recent work [50, 36, 47, 39], we measure accuracy with Precision@K - e.g., what fraction of the top K search results contain the tag of interest? In this setting, noisy labels y_i come from user provided tags which may contain errors and are typically incomplete. Conveniently, we can write expected performance Eq. 5.7 for Precision@K for a single tag in closed form:

$$E[Prec@K] = \frac{1}{K}\Big(\sum_{i \in V_K} \tilde{z}_i + \sum_{i \in U_K} p(z_i = 1|\mathcal{O}) \Big) \tag{5.8}$$

where we write V_K and U_K to denote the vetted and unvetted subsets of K highest-scoring examples in the total set $V \cup U$. Some benchmarks compute an aggregate mean precision over all tags under consideration, but since this average is linear, one again obtains a closed form estimate.

Instance segmentation: Instance segmentation is another natural task for which to apply active testing. It is well known that human annotation is prohibitively expensive – [21] reports that an average of more than 1.5 *hours* is required to annotate a single image. Widely used benchmarks such as [21] release small fraction of images annotated with high quality, along with a larger set of noisy or "coarse"-quality annotations. Other instance segmentation datasets such as COCO [65] are constructed stage-wise by first creating a detection dataset which only indicates rectangular bounding boxes around each object which are subsequently refined into a precise instance segmentations. Fig. 5.3 shows an example of a partially vetted image in which some instances are only indicated by a bounding box (noisy), while others have a detailed mask (vetted).

When computing Average Precision, a predicted instance segmentation is considered a true

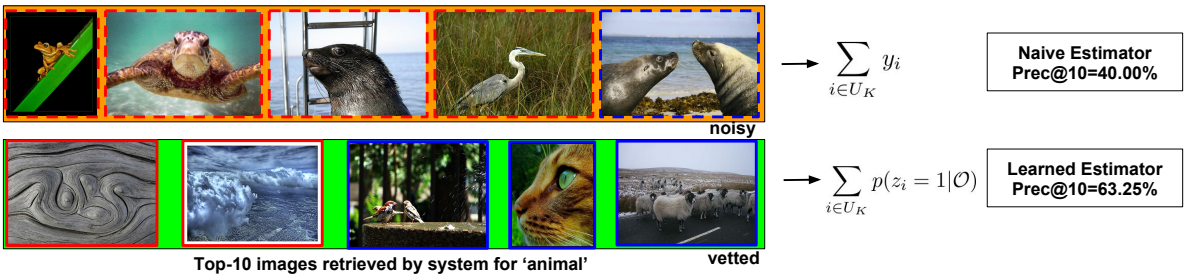


Figure 5.4: Estimating $Prec@K$. Images at left are the top K=10 entries returned by the system being evaluated. The image border denotes the current label and vetting status (solid blue/red = vetted positive/negative, and dotted blue/red = noisy positive/negative). Estimates of precision can be significantly improved by using a learned estimator trained on the statistics of examples that have already been vetted. Current approaches that evaluate on vetted-only or vetted+noisy labels (naive) produce poor estimates of precision (30% and 40% respectively). Our learned estimator is much closer to the true precision (63% vs 80% respectively).

positive if it has sufficient intersection-over-union (IoU) overlap with a ground-truth instance. In this setting, we let the variable z_i indicate that predicted instance i is matched to a ground-truth instance and has an above threshold overlap. Assuming independence of z_i's, the expected AP can be written as (see Appendix A.1 for proof):

$$E[AP] = \frac{1}{N_p}\Big(\sum_{k \in V} \tilde{z}_k E[Prec@k] + \sum_{k \in U} p(z_k = 1|\mathcal{O}) E[Prec@k]\Big) \tag{5.9}$$

In practice, standard instance segmentation benchmarks are somewhat more complicated. In particular, they enforce one-to-one matching between detections and ground-truth. For example, if two detections overlap a ground-truth instance, only one is counted as a true positive while the other is scored as a false positive. This also holds for multi-class detections - if a detection is labeled as a dog (by matching to a ground-truth dog), it can no longer be labeled as cat. While this interdependence can in principle be modeled by the conditioning variables $\mathcal{O}$ which could include information about which class detections overlap, in practice our estimators for $p(z_i = 1|\mathcal{O})$ do not take this into account. Nevertheless, we show that such estimators provide remarkably good estimates of performance.

Fitting estimators to partially vetted data: We alternate between vetting small batches

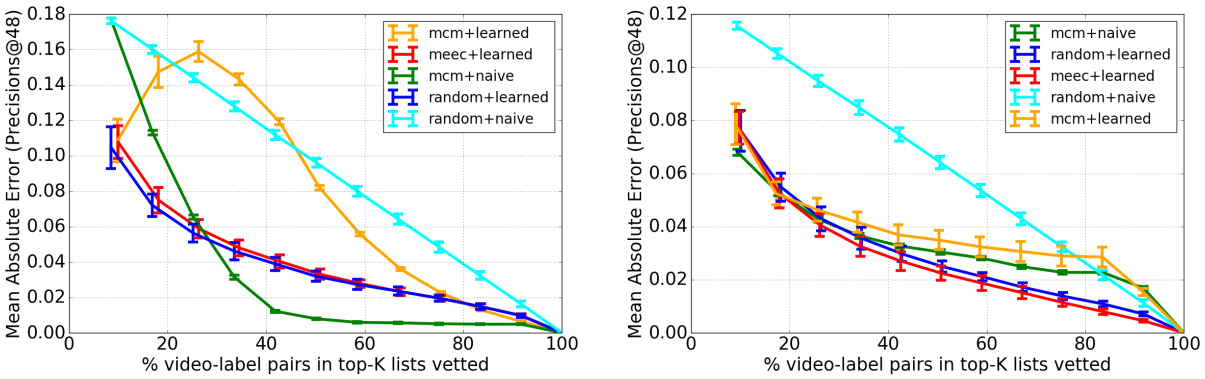
of data and refitting the estimator to the vetted set. For multi-label tagging, we update estimates for the prior probability that a noisy tag for a particular category will be flipped when vetted $p(\tilde{z}_i \neq y_i)$. For instance segmentation, we train a per-category classifier that uses sizes of the predicted and unvetted ground-truth bounding box to predict whether a detected instance will overlap the ground-truth. We discuss the specifics of fitting these particular estimators in the experimental results.

5.2.2 Vetting Strategies

The second component of the active testing system is a strategy for choosing the "next" data samples to vet. The goal of such a strategy is to produce accurate estimates of benchmark performance with fewest number of vettings. An alternate, but closely related goal, is to determine the benchmark rankings of a set of recognition systems being compared. The success of a given strategy depends on the distribution of the data, the chosen estimator, and the system(s) under test. We consider several selection strategies, motivated by existing data collection practice and modeled after active learning, which adapt to these statistics in order to improve efficiency.

Random Sampling: The simplest vetting strategy is to choose test examples to vet at random. The distribution of examples across categories often follows a long-tail distribution. To achieve faster uniform convergence of performance estimates across all categories, we use a hierarchical sampling approach in which we first sample a category and then select a sub-batch of test examples to vet from that category. This mirrors the way, e.g. image classification and detection datasets are manually curated to assure a minimum number of examples per category.

Most-Confident Mistake (MCM): This strategy selects unvetted examples for which the system under test reports a high-confidence detection/classification score, but which are



(a) Micro-Videos (b) NUS-WIDE

Figure 5.5: Results for multi-label classification task. The figures show the mean and standard deviation of the estimated Precision@48 at different amount of annotation efforts. Using a fairly simple estimator and vetting strategy, the proposed framework can estimate the performance very closely to the true values. For references, the precision@48 averaged across classes is 20.06% and 19.88% for Microvideos and NUS-WIDE respectively.

considered a mistake according to the current metric estimator. Specifically, we focus on the strategy of selecting *Most-confident Negative* which is applicable to image/video tagging where the set of user-provided tags are often incomplete. The intuition is that, if a high-performance system believes that the current sample is a positive with high probability, it's likely that the noisy label is at fault. This strategy is motivated by experience with object detection benchmarks where, e.g., visualizing high-confident false positive face detections often reveals missing annotations in the test set [70].

Maximum Expected Estimator Change (MEEC): In addition to utilizing the confidence scores produced by the system under test, it is natural to also consider the uncertainty in the learned estimator $p_{est}(z)$. Exploiting the analogy of active testing with active learning, it is natural to vet samples that are most confusing to the current estimator (e.g., with largest entropy), or ones that will likely generate a large update to the estimator (e.g., largest information gain).

Specifically, we explore a active selection strategy based on *maximum expected model change* [92], which in our case corresponds to selecting a sample that yields the largest

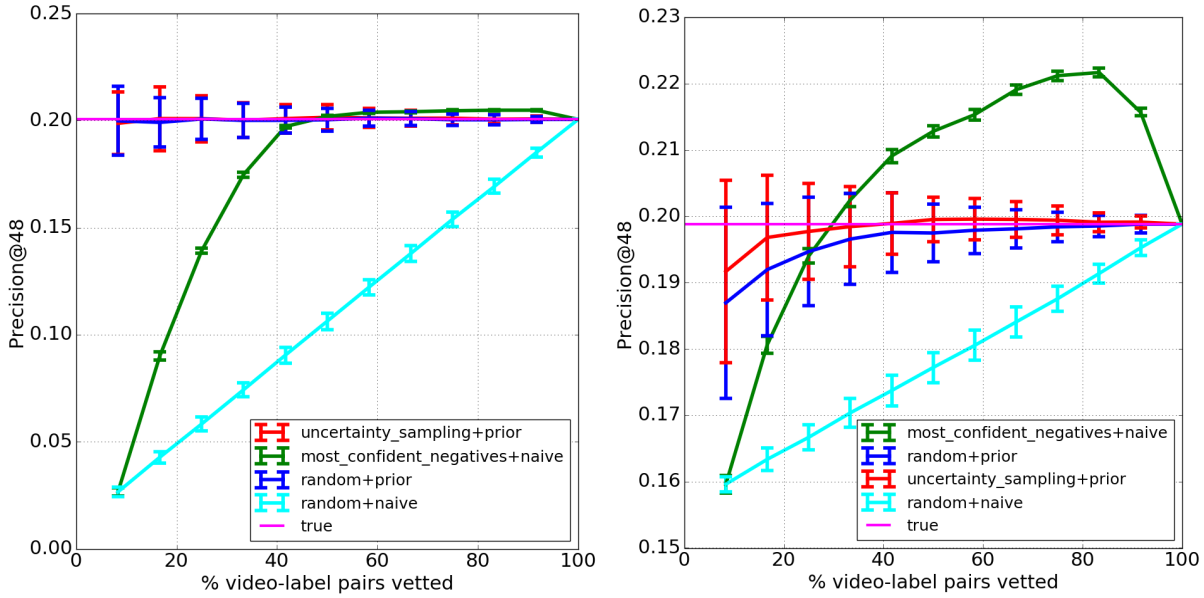


Figure 5.6: Results for multi-label classification task. The figures show the mean and standard deviation of the estimated Precision@K at different amount of annotation efforts. Using a fairly simple estimator and vetting strategy, the proposed framework can estimate the performance very closely to the true values.

expected change in our estimate of Q. Let $E_{p(z|V)}[Q(z)]$ be the expected performance based on the distribution $p(z|V)$ estimated from the current vetted set V. $E_{p(z|V,z_i)}[Q(z)]$ be the expected performance after vetting example i and updating the estimator based on the outcome. The *actual* change in the estimate of Q depends on the realization of the random variable z_i:

$$\Delta_i(z_i) = \left| E_{p(z|V,z_i)}[Q(z)] - E_{p(z|V)}[Q(z)] \right| \tag{5.10}$$

We can choose the example i with the largest *expected* change, using the current estimate of the distribution over $z_i \sim p(z_i|V)$ to compute the expected change $E_{p(z_i|V)}\left[\Delta_i(z_i)\right]$.

For $Prec@K$, this expected change is given by:

$$E_{p(z_i|V)}\left[\Delta_i(z_i)\right] = \frac{2}{K} p_i(1-p_i) \tag{5.11}$$

where we write $p_i = p(z_i = 1|\mathcal{O})$. Interestingly, selecting the sample yielded the maximum expected change in the estimator corresponds to a standard *maximum entropy* selection criteria for active learning. Similarly, in Appendix A.1 we show that for AP:

$$E_{p(z_i|V)}\left[\Delta_i(z_i)\right] = \frac{1}{N_p} r_i p_i (1 - p_i) \tag{5.12}$$

where r_i is the proportion of unvetted examples scoring higher than example i and N_p is the number of positive instances in the dataset. In this case, we select an example to vet which has high-entropy and for which there is a relatively small proportion of higher-scoring unvetted examples.

5.3 Experiments

We validate our active testing framework on two specific applications, multi-label classification and instance segmentation. For each of these applications, we describe the datasets and systems evaluated and the specifics of the estimators and vetting strategies used.

5.3.1 Active Testing for Multi-label Classification

NUS-WIDE: This dataset contains 269,648 Flickr images with 5018 unique tags. The authors also provide a 'semi-complete' ground-truth via manual annotations for 81 concepts. We removed images that are no longer available and images that don't contain one of the 81 tags. We are left with around 100K images spanning across 81 concepts. [47] analyzed the noisy and missing label statistics for this dataset. Given that the tag is relevant to the image, there is only 38% chance that it will appear in the noisy tag list. If the tag does not

apply, there's 1% chance that it appears anyway. They posited that the missing tags are either non-entry level categories (e.g., person) or they are not important in the scene (e.g., clouds and buildings).

Micro-videos: Micro-videos have recently become a prevalent form of media on many social platforms, such as Vine, Instagram, and Snapchat. [74] formulated a multi-label video-retrieval/annotation task for a large collection of Vine videos. They introduce a micro-video dataset, **MV-85k** containing 260K videos with 58K tags. This dataset, however, only provides exhaustive vetting for a small subset of tags on a small subset of videos. We vetted 26K video-tag pairs from this dataset, spanning 17503 videos and 875 tags. Since tags provided by users have little constraints, this dataset suffers from both under-tagging and over-tagging. Under-tagging comes from not-yet popular concepts, while over-tagging comes from the spamming of extra tags In our experiments we use a subset of 75 tags.

Recognition systems: To obtain the classification results, we implement two multi-label classification algorithms for images (NUSWIDE) and videos (Microvideos). For NUS-WIDE, we trained a multi-label logistic regression model built on the pretrained ResNet-50 [41] features. For Micro-videos, we follow the state-of-the-art video action recognition framework [120] modified for the multi-label setting to use multiple logistic cross-entropy losses.

Learned Estimators: We use Precision@48 as a evaluation metric. For tagging, we estimate the posterior over unvetted tags, $p(z_i|\mathcal{O})$, based on two pieces of observed information: the statistics of noisy labels y_i on vetted examples, and the system confidence score, s_i. This posterior probability can be derived as (see Appendix A.1 for proof):

$$p(z_i|s_i, y_i) = \frac{p(y_i|z_i)p(z_i|s_i)}{\sum_{v\in\{0,1\}} p(y_i|z_i = v)p(z_i = v|s_i)} \tag{5.13}$$

Given some vetted data, we fit the tag-flipping priors $p(y_i|z_i)$ by standard maximum likelihood estimation (counting frequencies). The posterior probabilities of the true label given the

classifier confidence score, $p(z_i|s_i)$, are fit using logistic regression.

5.3.2 Object Instance Detection and Segmentation

COCO Minival: For instance segmentation, we use 'minival2014' subset of the COCO dataset [65]. This subset contains 5k images spanning over 80 categories. We report the standard COCO metric: Average Precision (averaged over all IoU thresholds).

To systematically analyze the impact of evaluation on noise and vetting, we focus evaluation efforts on the high quality test set, but simulate noisy annotations by replacing actual instance segmentation masks by their tight-fitting bounding box (the unvetted "noisy" set). We then simulate active testing where certain instances are vetted, meaning the bounding-box is replaced by the true segmentation mask.

Detection Systems: We did not implement instance segmentation algorithms ourselves, but instead utilized three sets of detection mask results produced by the authors of Mask R-CNN [40]. These were produced by variants of the instance segmentation systems proposed in [126, 64, 40].

Learned Estimators: To compute the probability whether a detection will pass the IoU threshold with a bounding box unvetted ground-truth instance ($p(z_i|\mathcal{O})$ in Eq. 5.9), we train a χ^2-SVM using the vetted portion of the database. The features for an example includes the category id, the 'noisy' IoU estimate, the size of the bounding box containing the detection mask and the size of ground-truth bounding box. The training label is true whether the true IoU estimate, computed using the vetted ground-truth mask and the detection masks, is above a certain input IoU threshold.

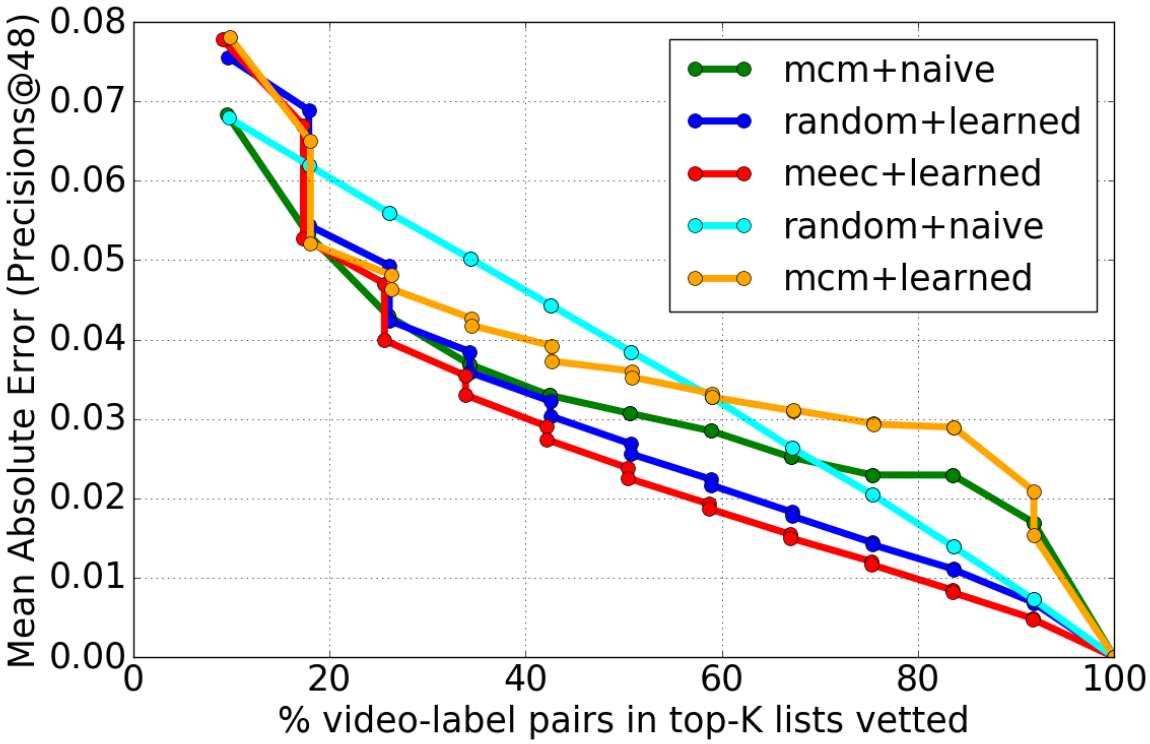


Figure 5.7: Decoupling the effect of model change and vetting effort for NUS-WIDE. This figure shows the reduction in estimation errors. The vertical drop at the same % vetted point indicates the reduction due to estimator quality. The slope between adjacent points indicates value of vetting examples. A steeper slope means the strategy is able to obtain a better set. In some sense, traditional active learning is concerned primarily with the vertical drop (i.e. a better model/predictor), while active testing also takes direct advantage of the slope (i.e. more vetted labels).

5.3.3 Efficiency of active testing estimates

We measure the estimation accuracy of different combination of vetting strategies and estimators at different amount of vetting efforts. We compute the absolute error between the estimated metric and the true (fully vetted) metric and average over all classes. Averaging the absolute estimation error across classes prevents over-estimation for one class canceling out under-estimation from another class. We plot the mean and the standard deviation over 50 simulation runs of each active testing approach.

Performance estimation: Figure 5.5 shows the results for estimating $Prec@48$ for NUSWIDE

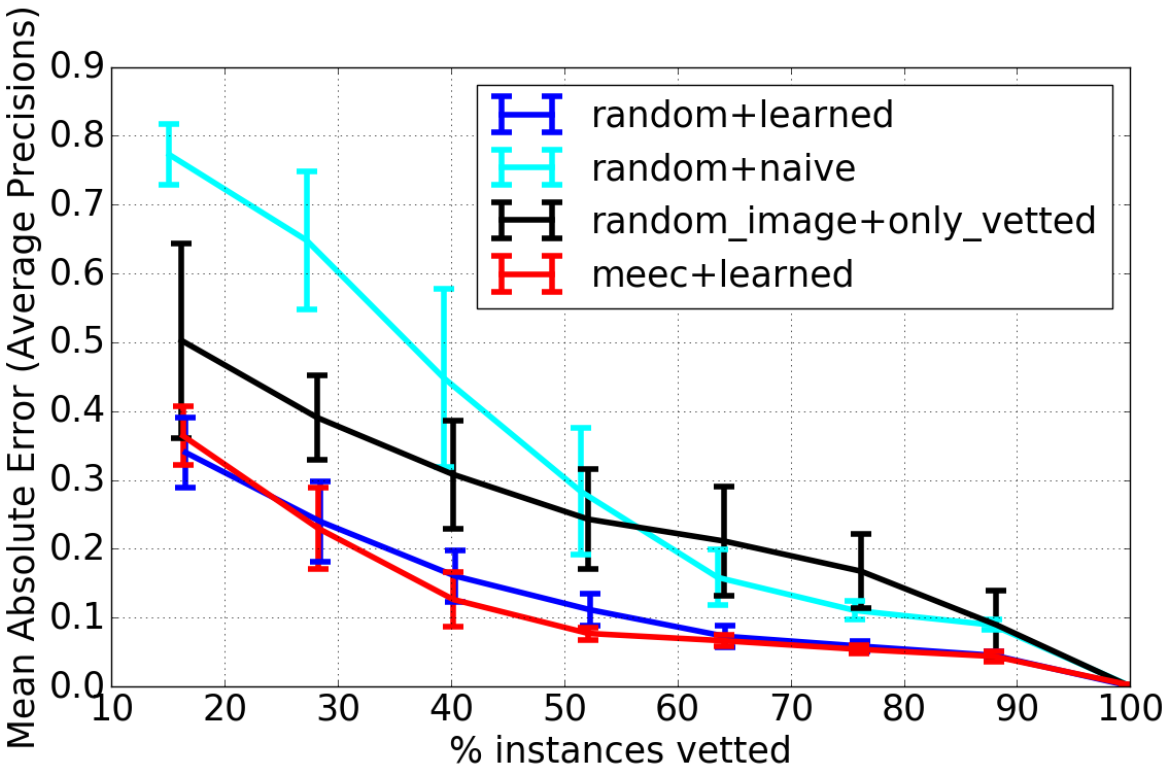


Figure 5.8: Results for instance segmentation. With 50% of instances vetted, our best model's estimation is 1% AP off from the true values with the standard deviation $\leq 1\%$. A smart estimator with a smarter querying strategy can make the approach more robust and efficient. Our approach has better approximation and is less prone to sample bias compared to the standard approach("random image"+ "only vetted").

and Microvideos. The x-axis indicates the percentage of the top-k lists that are vetted. For the *Prec@K* metric, it is only necessary to vet 100% of the top-k lists rather than 100% of the whole test set[2]. A 'random' strategy with a 'naive' estimator follows a linear trend since each batch of vetted examples contributes on average the same reduction in estimation error. The most confident mistake (mcm) heuristic works very well for Microvideos due to the substantial amount of under-tagging. However, in more reasonable balanced settings such as NUS-WIDE, this heuristic does not perform as well. The MCM vetting strategy does not pair well with a learned estimator due to its biased sampling which quickly results in priors that overestimate the number missing tags. In contrast, the random and active *MEEC*

[2]The "vetted only" estimator is not applicable in this domain until at least K examples in each short list have been vetted and hence doesn't appear in the plots.

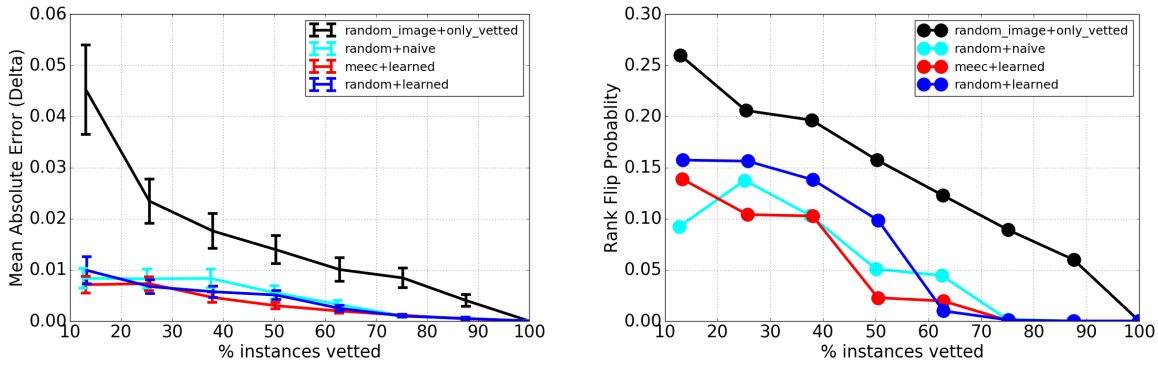


Figure 5.9: Relative performance differences and their relative ranking for multiple input systems. The left plot shows the mean squared errors between the current difference to the true difference. The right plot shows how often the ranking orders between two input algorithms are flipped. Both figures suggest that our active testing framework is a more robust and efficient approach toward comparing models. With 50% of the data vetted, standard approaches that evaluate on only vetted data (black curve) incorrectly rank algorithms 16% of the time, while our learned estimators with active vetting (red curve) reduce this error to 3% of the time.

vetting strategies offer good samples for learning a good estimator. At 50% vetting effort, *MEEC* sampling with a learned estimator on average can achieve within 2-3% of the real estimates.

Figure 5.6 shows the results of estimating absolute precision@48 for the multilabel classification tasks on both NUS-WIDE and Microvideos datasets. In contrast, plots in the main paper show the total absolute error from the true value.

Figure 5.7 highlights the relative value of establishing the true vetted label versus the value of vetted data in updating the estimator. In some sense, traditional active learning is concerned primarily with the vertical drop (i.e. a better model/estimator), while active testing also takes direct advantage of the slope (i.e. more vetted labels). The initial learned estimates have larger error due to small sample size, but the fitting during the first few vetting batches rapidly improves the estimator quality. Past 40% vetting effort, the estimator model parameters stabilize and remaining vetting serves to correct labels whose true value can't be predicted given the low-complexity of the estimator.

Figure 5.8 shows similar results for estimating the mAP for instance segmentation on COCO. The current 'gold standard' approach of estimating performance based only on the vetted subset of images leads to large errors in estimation accuracy and high variance from from small sample sizes. In the active testing framework, input algorithms are tested using the whole dataset (vetted and unvetted). Naive estimation is noticeably more accurate than vetted only and the learned estimator with uncertainty sampling further reduces both the absolute error and the variance.

Model ranking: The benefits of active testing are highlighted further when we consider the problem of ranking system performance. We are often interested not in the absolute performance number, but rather in the performance gap between different systems. We find that active testing is also valuable in this setting. Figure 5.9 shows the error in estimating the *performance gap* between two different instance segmentation systems as a function of the amount data vetted. This follows a similar trend as the single model performance estimation plot. Importantly, it highlights that only evaluating vetted data, though unbiased, typically produces a large error in in performance gap between models to high variance in the estimate of each individual models performance. In particular, if we use these estimates to rank two models, we will often make errors in model ranking even when relatively large amounts of the data have been vetted. Using stronger estimators, actively guided by *MEEC* sampling provide accurate rankings with substantially less vetting effort. With 50% of the data vetted, standard approaches that evaluate on only vetted data (black curve) incorrectly rank algorithms 15% of the time, while our learned estimators with active vetting (red curve) reduce this error to 3% of the time.

5.4 Conclusions

We have introduced a general framework for active testing that minimizes human vetting effort by actively selecting test examples to label and using performance estimators that adapt to the statistics of the test data and the systems under test. Simple implementations of this concept demonstrate the potential for radically decreasing the human labeling effort needed to evaluate system performance for standard computer vision tasks. We anticipate this will have substantial practical value in the ongoing construction of such benchmarks.

Bibliography

[1] https://github.com/starlock/vino/wiki/api-reference.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[3] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the roc curve. *Journal of Machine Learning Research*, 2005.

[4] H. Alwassel, F. Caba Heilbron, and B. Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *ECCV*, pages 253–269, 2018.

[5] H. Alwassel, F. C. Heilbron, and B. Ghanem. Action search: Learning to search for human activities in untrimmed videos. In *arXiv preprint arXiv:1706.04269*, 2017.

[6] R. Anirudh and P. Turaga. Interactively test driving an object detector: Estimating performance on unlabeled data. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 175–182. IEEE, 2014.

[7] H. Aradhye, G. Toderici, and J. Yagnik. Video2text: Learning to annotate video content. In *ICDMW'09*, pages 144–151. IEEE, 2009.

[8] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *ACM SIGIR*. ACM, 2006.

[9] M.-F. Balcan and R. Urner. Active learning–modern learning theory. *Encyclopedia of Algorithms*, pages 8–13, 2016.

[10] Y. Bengio and O. Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*, pages 18–36. Springer, 2011.

[11] H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016.

[12] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014.

[13] S. Boyd, C. Cortes, M. Mohri, and A. Radovanovic. Accuracy at the top. In *Advances in neural information processing systems*, pages 953–961, 2012.

[14] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.

[15] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles. SST: single-stream temporal action proposals. In *CVPR*, 2017.

[16] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[17] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[18] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, 2018.

[19] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *ICCV*, pages 1409–1416, 2013.

[20] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.

[21] 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*, 2016.

[22] E. Cunha, G. Magno, G. Comarela, V. Almeida, M. A. Gonçalves, and F. Benevenuto. Analyzing the dynamic evolution of hashtags on twitter: a language-based approach. In *Proceedings of the Workshop on Languages in Social Media*, pages 58–65. Association for Computational Linguistics, 2011.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: a large-scale hierarchical image database. In *CVPR*, 2009.

[24] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2012.

[25] V. Escorcia, F. C. Heilbron, J. C. Niebles, , and B. Ghanem. DAPs: deep action proposals for action understanding. In *ECCV*, 2016.

[26] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[27] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[28] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015.

[29] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV, 2011*.

[30] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016.

[31] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017.

[32] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[33] N. Gao, W. Webber, and D. W. Oard. Reducing reliance on relevance judgments for system comparison by using expectation-maximization. In *European Conference on Information Retrieval*. Springer, 2014.

[34] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[35] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015.

[36] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. *arXiv preprint arXiv:1312.4894*, 2013.

[37] A. Gorban, H. Idrees, Y. Jiang, A. R. Zamir, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2015.

[38] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *arXiv:1705.08421*, 2017.

[39] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 309–316. IEEE, 2009.

[40] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[42] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. ActivityNet: a large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[43] F. C. Heilbron, J. C. Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, 2016.

[44] S. I. Hill, H. Zaragoza, R. Herbrich, and P. J. Rayner. Average precision and the problem of generalisation. In *ACM SIGIR Workshop on Mathematical and Formal Methods in Information Retrieval*, 2002.

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

[46] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *ECCV*, 2016.

[47] H. Izadinia, B. C. Russell, A. Farhadi, M. D. Hoffman, and A. Hertzmann. Deep classifiers from image tags in the wild. In *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*, pages 13–18. ACM, 2015.

[48] Y.-G. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes, 2014.

[49] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

[50] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, pages 67–84. Springer, 2016.

[51] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

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

[53] 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.

[54] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[55] F. K. Khattak and A. Salleb-Aouissi. Quality control of crowd labeling through expert evaluation. In *Proceedings of the NIPS 2nd Workshop on Computational Social Science and the Wisdom of Crowds*, volume 2, 2011.

[56] G. Kim, E. P. Xing, and A. Torralba. Modeling and analysis of dynamic behaviors of web image collections. In *ECCV 2010*, pages 85–98. Springer, 2010.

[57] K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *CVPR 2011*, pages 3241–3248. IEEE, 2011.

[58] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV, 2011*.

[59] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[60] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW Conference*, pages 591–600. ACM, 2010.

[61] Y. J. Lee and K. Grauman. Predicting important objects for egocentric video summarization. *IJCV*, pages 1–18, 2014.

[62] D. Li and E. Kanoulas. Active sampling for large-scale information retrieval evaluation. In *ACM on Conference on Information and Knowledge Management*. ACM, 2017.

[63] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and J. Li. Learning from noisy labels with distillation. *arXiv preprint arXiv:1703.02391*, 2017.

[64] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[65] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[66] W. Liu and A. Berg. Imagenet video object detection. `http://image-net.org/challenges/LSVRC/2015/index`.

[67] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016.

[68] M. W. Marien. *Photography: A cultural history*. Laurence King London, 2002.

[69] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR, 2009*.

[70] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *European Conference on Computer Vision*, pages 720–735. Springer, 2014.

[71] A. Moffat, W. Webber, and J. Zobel. Strategic system comparisons via targeted relevance judgments. In *ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007.

[72] A. Montes, A. Salvador, S. Pascual, and X. Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. In *1st NIPS Workshop on Large Scale Computer Vision Systems (LSCVS)*, 2016.

[73] P. Nguyen, T. Liu, G. Prasad, and B. Han. Weakly supervised action localization by sparse temporal pooling network. *CVPR*, 2018.

[74] P. X. Nguyen, G. Rogez, C. Fowlkes, and D. Ramanan. The open world of micro-videos. *arXiv preprint arXiv:1603.09439*, 2016.

[75] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV, 2010*.

[76] S. Paul, S. Roy, and A. K. Roy-Chowdhury. W-talc: Weakly-supervised temporal activity localization and classification. *ECCV*, 2018.

[77] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR 2012*, pages 2847–2854. IEEE, 2012.

[78] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[79] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29–48. Springer, 2006.

[80] M. M. Rahman, M. Kutlu, T. Elsayed, and M. Lease. Efficient test collection construction via active learning. *arXiv preprint arXiv:1801.05605*, 2018.

[81] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics, 2010.

[82] M. Redi, N. OHare, R. Schifanella, M. Trevisiol, and A. Jaimes. 6 seconds of sound and vision: Creativity in micro-videos. In *CVPR, 2014*.

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

[84] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *CVPR*, 2016.

[85] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with RNN based fine-to-coarse modeling. In *CVPR*, 2017.

[86] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[87] M. Ryoo and J. Aggarwal. Ut-interaction dataset, icpr contest on semantic description of human activities (sdha), 2010.

[88] M. S. Ryoo, B. Rothrock, and L. Matthies. Pooled motion features for first-person videos. In *CVPR*. IEEE, 2015.

[89] A. Sabharwal and H. Sedghi. How good are my predictions? efficiently approximating precision-recall curves for massive datasets. In *UAI*, 2017.

[90] M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

[91] S. Sano, T. Yamasaki, and K. Aizawa. Degree of loop assessment in microvideo. In *ICIP 2014*, pages 5182–5186. IEEE, 2014.

[92] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[93] A. Sheshadri and M. Lease. Square: A benchmark for research on computing crowd consensus. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[94] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang. Learning long-term dependencies for action recognition with a biologically-inspired deep network. In *ICCV*, 2017.

[95] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. CDC: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. *CVPR*, 2017.

[96] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang. Autoloc: Weakly-supervised temporal action localization. *ECCV*, 2018.

[97] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016.

[98] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

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

[100] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, 2016.

[101] G. Singh and F. Cuzzolin. Untrimmed video classification for activity detection: submission to ActivityNet challenge. *arXiv preprint arXiv:1607.01979*, 2016.

[102] K. K. Singh and Y. J. Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017.

[103] K. Soomro, H. Idrees, and M. Shah. Action localization in videos through context walk. In *ICCV*, 2015.

[104] K. Soomro, A. R. Zamir, and M. Shah. UCF101: a dataset of 101 human action classes from videos in the wild. Technical Report CRCV-TR-12-01, University of Central Florida, 2012.

[105] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[106] G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, and J. Yagnik. Finding meaning on youtube: Tag recommendation and category discovery. In *CVPR 2010*, pages 3447–3454. IEEE, 2010.

[107] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[108] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015.

[109] A. Ulges, M. Koch, D. Borth, and T. M. Breuel. Tubetagger-youtube-based concept detection. In *ICDMW'09*, pages 190–195. IEEE, 2009.

[110] A. Vahdat and G. Mori. Handling uncertain tags in visual recognition. In *ICCV 2013*, pages 737–744. IEEE, 2013.

[111] V. Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[112] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. Learning from noisy large-scale datasets with minimal supervision. *arXiv preprint arXiv:1701.01619*, 2017.

[113] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal of Computer Vision*, 108(1-2):97–114, 2014.

[114] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2524–2531. IEEE, 2011.

[115] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV, 2013*.

[116] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[117] L. Wang, Y. Qiao, X. Tang, and L. V. Gool. Actionness estimation using hybrid fully convolutional networks. In *CVPR*, 2016.

[118] L. Wang, Y. Xiong, D. Lin, and L. van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 2017.

[119] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[120] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[121] R. Wang and D. Tao. UTS at Activitynet 2016. *AcitivityNet Large Scale Activity Recognition Challenge*, 2016.

[122] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal pyramid network for video action recognition. In *CVPR*, 2017.

[123] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. *An Improved Algorithm for TV-L^1 Optical Flow*. Statistical and geometrical approaches to visual motion analysis. Springer, 2009.

[124] P. Welinder, M. Welling, and P. Perona. A lazy man's approach to benchmarking: Semisupervised classifier evaluation and recalibration. In *CVPR*, 2013.

[125] B. Wu, S. Lyu, and B. Ghanem. Ml-mg: multi-label learning with missing labels using a mixed graph. In *Proceedings of the IEEE international conference on computer vision*, pages 4157–4165, 2015.

[126] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR, 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.

[127] Y. Xiong, Y. Zhao, L. Wang, D. Lin, and X. Tang. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716*, 2017.

[128] H. Xu, A. Das, and K. Saenko. R-C3D: region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.

[129] S. Yang and D. Ramanan. Multi-scale recognition with dag-cnns. *arXiv preprint arXiv:1505.05232*, 2015.

[130] W. Yang and G. Toderici. Discriminative tag learning on youtube videos with latent sub-tags. In *CVPR 2011*, pages 3217–3224. IEEE, 2011.

[131] G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 471–480. ACM, 2015.

[132] G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM MM*, 2015.

[133] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[134] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *ACM international conference on Information and knowledge management*. ACM, 2006.

[135] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, pages 593–601, 2014.

[136] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *CVPR*, 2016.

[137] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng. Temporal action localization by structured maximal sums. In *CVPR*, 2017.

[138] Y. Zhang, Y. Bai, M. Ding, Y. Li, and B. Ghanem. W2f: A weakly-supervised to fully-supervised framework for object detection. In *CVPR*, pages 928–936, 2018.

[139] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017.

[140] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. *ICCV*, 2017.

[141] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

Appendix A

Proofs for Active Testing

A.1 Expected Precision and Average Precision

Let Precision and Recall at K be defined as

$$P_k = \frac{1}{k} \sum_{i \leq k} z_i \tag{A.1}$$

and,

$$R_k = \frac{1}{N_p} \sum_{i \leq k} z_i \tag{A.2}$$

where N_p is the number of positive instances in the whole set. The average precision is given by integrating precision with respect to recall:

$$
\begin{aligned}
AP &= \sum_k (R_k - R_{k-1}) P_k \\
&= \sum_k \Big(\frac{1}{N_p} z_k\Big) P_k \\
&= \sum_k \Big(\frac{1}{N_p} z_k\Big)\Big(\frac{1}{k}\sum_{i \le k} z_i\Big) \\
&= \frac{1}{N_p}\sum_k \frac{z_k}{k}\sum_{i \le k} z_i
\end{aligned} \tag{A.3}
$$

We would like to compute expectations when some z_i are unobserved. For notational convenience, let $p_i = P(z_i = 1|\mathcal{O})$ when z_i is unobserved and $\tilde{z}_i$ be the observed value when the ground-truth associated with example i is vetted. We can then compute expected $Prec@K$ as:

$$
\begin{aligned}
E[Prec@K] &= \frac{1}{K}\sum_{i \le K} E[z_i] \\
&= \frac{1}{K}\left(\sum_{i \le K, i \in V} \tilde{z}_i + \sum_{i \le K, i \in U} p_i\right)
\end{aligned} \tag{A.4}
$$

And the expected change for this metric is given by:

$$
\begin{aligned}
E_{p(z_i|V)}\left[\Delta_i(z_i)\right] &= p_i \frac{1}{K}|1 - p_i| + (1 - p_i)\frac{1}{K}|0 - p_i| \\
&= \frac{2}{K} p_i (1 - p_i)
\end{aligned} \tag{A.5}
$$

where we write $p_i = p(z_i = 1|\mathcal{O})$.

Expected AP is more interesting because it includes products of of the z_i.

$$\begin{aligned} E[AP] &= \frac{1}{N_p} \sum_k \frac{1}{k} E[z_k \sum_{i \leq k} z_i] \\ &= \frac{1}{N_p} \sum_k \frac{1}{k} \sum_{i \leq k} E[z_k z_i] \end{aligned}$$

We note that in our application of evaluating instance segmentation, the quantity N_p is known prior to vetting. In other settings, it may also be a random variable that depends on the vetting outcomes. In the following derivation, we temporarily drop the constant $\frac{1}{N_p}$ to reduce notational clutter.

Assuming independence of z_i and z_k, we have:

$$\begin{aligned} E[z_i] &= p_i \\ E[z_i z_k] &= p_i p_k \end{aligned}$$

Expanding the vetted and unvetted terms we can compute:

$$\begin{aligned}
E[AP] =& \sum_{k} \frac{1}{k} \sum_{i \leq k} E[z_k z_i] \\
=& \sum_{k \in V} \frac{1}{k} (\sum_{i \leq k, i \in V} z_k z_i) + \sum_{k \in V} \frac{1}{k} (\sum_{i \leq k, i \in U} z_k p_i) \\
&+ \sum_{k \in U} \frac{1}{k} (\sum_{i \leq k, i \in V} p_k z_i) + \sum_{k \in U} \frac{1}{k} (\sum_{i \leq k, i \in U} p_k p_i)
\end{aligned}$$

which can be written a bit more compactly as:

$$E[AP] = \Big(\sum_{k \in V} (\frac{z_k}{k} E[Prec@k]) + \sum_{k \in U} (\frac{p_k}{k} E[Prec@k]) \Big)$$

We would like to compute the change in $E[AP]$ when we vet some example. Before vetting sample j, we have:

$$
\begin{aligned}
E[AP] = & \sum_{k \in V} \frac{1}{k} \left[\sum_{i \leq k, i \in V} z_k z_i \right] \\
& + \sum_{k \in V} \frac{1}{k} \left[\sum_{i \leq k, i \in U \setminus j} z_k p_i + z_k p_j \delta[j \leq k] \right] \\
& + \sum_{k \in U \setminus j} \frac{1}{k} \left[\sum_{i \leq k, i \in V} p_k z_i \right] + \frac{1}{j} \left[\sum_{i \leq j, i \in V} p_j z_i \right] \\
& + \sum_{k \in U \setminus j} \frac{1}{k} \left[\sum_{i \leq k, i \in U \setminus j} p_k p_i + p_k p_j \delta[j \leq k] \right] \\
& \quad + \frac{1}{j} \left[\sum_{i \in U \setminus j, i \leq j} p_j p_i + p_j p_j \right]
\end{aligned}
$$

After vetting the example j, we have:

$$
\begin{aligned}
E[AP|z_j] = & \sum_{k \in V} \frac{1}{k} \sum_{i \leq k, i \in V} z_k z_i \\
& + \sum_{k \in V} \frac{1}{k} \left[\sum_{i \leq k, i \in U \setminus j} z_k p_i + z_k z_j \delta[j \leq k] \right] \\
& + \sum_{k \in U \setminus j} \frac{1}{k} \left[\sum_{i \leq k, i \in V} p_k z_i \right] + \frac{1}{j} \left[\sum_{i \leq j, i \in V} z_j z_i \right] \\
& + \sum_{k \in U \setminus j} \frac{1}{k} \left[\sum_{i \leq k, i \in U \setminus j} p_k p_i + p_k z_j \delta[j \leq k] \right] \\
& \quad + \frac{1}{j} \left[\sum_{i \leq j, i \in U \setminus j} z_j p_i + z_j z_j \right]
\end{aligned}
$$

The difference between these estimates is,

$$\begin{aligned}
\Delta(z_j) &= E[AP|z_j] - E[AP] \\
&= \sum_{k \in V} \frac{1}{k} z_k (z_j - p_j) \delta[j \leq k] + \\
&\quad \frac{1}{j} \sum_{i \leq j, i \in V} (z_j - p_j) z_i + \\
&\quad \sum_{k \in U \setminus j} \frac{1}{k} \left[p_k (z_j - p_j) \delta[j \leq k] \right] + \\
&\quad \frac{1}{j} \left[\sum_{i \leq j, i \in U \setminus j} (z_j p_i + z_j z_j - p_j p_i - p_j p_j) \right]
\end{aligned}$$

The expected reduction given our estimator for z_j is

$$E[\Delta] = p_j \Delta(z_j = 1) + (1 - p_j) \Delta(z_j = 0)$$

where:

$$\begin{aligned}\Delta(z_j = 0) &= \sum_{k \in V} \frac{1}{k} z_k (-p_j) \delta[j \leq k] + \\ &\quad \frac{1}{j} \sum_{i \leq j, i \in V} (-p_j) z_i + \\ &\quad \sum_{k \in U \setminus j} \frac{1}{k} p_k (-p_j) \delta[j \leq k] + \\ &\quad \frac{1}{j} \left[\sum_{i \leq j, i \in U \setminus j} (-p_j p_i - p_j p_j) \right]\end{aligned}$$

$$\begin{aligned}\Delta(z_j = 1) &= \sum_{k \in V} \frac{1}{k} z_k (1 - p_j) \delta[j \leq k] + \\ &\quad \frac{1}{j} \sum_{i \leq j, i \in V} (1 - p_j) z_i + \\ &\quad \sum_{k \in U \setminus j} \frac{1}{k} p_k (1 - p_j) \delta[j \leq k] + \\ &\quad \frac{1}{j} \left[\sum_{i \leq j, i \in U \setminus j} (p_i + 1 - p_j p_i - p_j p_j) \right]\end{aligned}$$

Let's just look at the first pair of corresponding terms of $E[\Delta]$,

$$(1 - p_j) \sum_{k \in V} \frac{1}{k} z_k (-p_j) \delta[j \leq k] + p_j \sum_{k \in V} \frac{1}{k} z_k (1 - p_j) \delta[j \leq k]$$

It is clear to see that the above summation equals 0. This is also true for the second and

third terms,

$$(1-p_j)\frac{1}{j}\sum_{i\leq j, i\in V}(-p_j)z_i + p_j\frac{1}{j}\sum_{i\leq j, i\in V}(1-p_j)z_i$$

$$(1-p_j)\sum_{k\in U\setminus j}\frac{1}{k}p_k(-p_j)\delta[j\leq k] + p_j\sum_{k\in U\setminus j}\frac{1}{k}p_k(1-p_j)\delta[j\leq k]$$

Only the last pair of terms remains, and simplifies as:

$$\begin{aligned}
E[\Delta] &= -\frac{1}{j}\sum_{i\leq j, i\in U\setminus j} p_j^3 - p_j + p_j^2(1-p_j) \\
&= -\frac{1}{j}\sum_{i\leq j, i\in U\setminus j} p_j^3 - p_j + p_j^2 - p_j^3 \\
&= -\frac{1}{j}\sum_{i\leq j, i\in U\setminus j} -p_j + p_j^2 \\
&= \frac{1}{j}\sum_{i\leq j, i\in U\setminus j} p_j(1-p_j)
\end{aligned}$$

Let r_j be the proportion of unvetted examples scoring higher than example j:

$$\begin{aligned}
r_j &= \frac{|\{i\in U : i\leq j\}|}{|\{i\in U\cup V : i\leq j\}|} \\
&= \frac{1}{j}\sum_{i<j}\delta(i\in U)
\end{aligned}$$

Putting back in the constant scaling yields the expression,

$$\begin{aligned}
E[\Delta] &= \frac{1}{N_p}\frac{1}{j}\sum_{i\leq j, i\in U\setminus j} p_j(1-p_j) \\
&= \frac{1}{N_p}r_j p_j(1-p_j)
\end{aligned}$$

The term is largest when p_j is 0.5 and decrease as it approaches 0 or 1. The term also

decreases when there are many unvetted examples that score higher than j since they have relatively more impact on the AP.

A.2 Estimator for multilabel classification

Here we derive the basis for Equation 5.13.

$$\begin{aligned} p(z_i|y_i, s_i) &= \frac{p(z_i, y_i, s_i)}{\sum_{v\in\{0,1\}} p(z_i = v, y_i, s_i)} \\ &= \frac{p(y_i|z_i, s_i)p(z_i|s_i)}{\sum_{v\in\{0,1\}} p(y_i|z_i, s_i)p(z_i|s_i)} \end{aligned}$$

We assume that given the true label, z_i, the observed label y_i is conditionally independent of the classifier score, s_i. With $p(y_i|z_i, s_i) = p(y_i|z_i)$, the expression simplifies to,

$$p(z_i|y_i, s_i) = \frac{p(y_i|z_i)p(z_i|s_i)}{\sum_{v\in\{0,1\}} p(y_i|z_i)p(z_i|s_i)}$$