University of California
Santa Barbara

Attention Models for Activity Detection

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

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Oytun Ulutan

Committee in charge:

Professor B.S. Manjunath, Chair
Professor Kenneth Rose
Professor Shivkumar Chandrasekaran
Professor Ramtin Pedarsani
Lance Kaplan, PhD

December 2019

The Dissertation of Oytun Ulutan is approved.

Professor Kenneth Rose

Professor Shivkumar Chandrasekaran

Professor Ramtin Pedarsani

Lance Kaplan, PhD

Professor B.S. Manjunath, Committee Chair

November 2019

Attention Models for Activity Detection

To my parents.

Acknowledgements

First, I would like to thank my dad Levent and my mom Sevim. Their sacrifice brought me to where I am today. I thank them very much for everything they have done and supporting me in my education career. I am grateful for my partner Sophia for all of her patience and support during these years. I am also grateful for all of my friends who made Santa Barbara my home.

I want to especially thank Prof. Manjunath for his supervision during my graduate school. It has been a great privilege and I have learned a lot from him. He has taught me to think out of the box and how to do exciting research.

I would like to thank my thesis committee, Prof. Rose, Prof. Chandrasekaran, Prof. Pedarsani and Dr. Kaplan for their great feedback during my research.

I am grateful to Dr. Riggan from ARL for all of his help in my first years of graduate school which shaped my research significantly. I would also like to acknowledge Dr. Rallapalli and Dr. Srivatsa from IBM Research for giving me the great internship opportunities and their guidance which helped my research.

It has been a pleasure working in Vision Research Lab and interacting with many colleagues. I would like to thank my friend and collaborator Dr. Carlos Torres for his help. I am thankful for all of the VRL members, Iftekhar, Satish, Aditya, Angela, Amil, Amir, Archith, Austin, Devendra, Dmitry, Jiaxiang, Karthikeyan, Lakshman, Mike, Po-Yu, Rahul, Shailja, Thuyen, Utkarsh.

Research was supported in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S.

Curriculum Vitæ

Oytun Ulutan

Education

December 2019	**Doctor's of Philosophy (PhD)** Electrical and Computer Engineering, University of California, Santa Barbara.
December 2015	**Master's of Science (MS)** Electrical and Computer Engineering, University of California, Santa Barbara.
June 2014	**Bachelor's of Science (BS)** Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey.

Fields of Study

Computer Vision, Deep Learning, Machine Learning, Video Analytics, Multi-Modal Learning

Experience

6/2015 - 12/2019	**Vision Research Lab (VRL)** Graduate Student Researcher University of California, Santa Barbara, CA.
6/2018 - 9/2018	**IBM Research** Summer Research Intern Yorktown Heights, NY.
6/2017 - 9/2017	**IBM Research** Summer Research Intern Yorktown Heights, NY.
1/2017 - 3/2017	**US Army Research Lab (ARL)** Research Intern Adelphi, MD.
7/2016 - 8/2016	**US Army Research Lab (ARL)** Research Intern Adelphi, MD.

Publications

Oytun Ulutan, Swati Rallapalli, Mudhakar Srivatsa, Carlos Torres, B.S. Manjunath, "Actor Conditioned Attention Maps for Video Action Detection", Winter Conference on Applications of Computer Vision (WACV 2020).

Oytun Ulutan, ASM Iftekhar, B.S. Manjunath, "VSGNet: Spatial Attention Network for Detecting Human Object Interactions Using Graph Convolutions", Under Submission CVPR2020.

O. Ulutan, B.S. Riggan, N.M. Nasrabadi, B.S. Manjunath, "An Order Preserving Bilinear Model for Person Detection in Multi-Modal Data", Winter Conference on Applications of Computer Vision, (WACV 2018).

Oytun Ulutan, B. S. Manjunath. UCSB, ActivityNet-AVA Actions Challenge Solution Actor Conditioned Attention Maps for Video Action Detection. UCSB Submission to ActivityNet AVA - CVPRW 2019

Oytun Ulutan, Swati Rallapalli, Mudhakar Srivatsa, B.S. Manjunath. Joint Classification and Detection Using LSTMs UCSB & IBM Research Submission to ActivityNet Challenge - CVPRW 2017

Liu, Xiaochen; **Ulutan, Oytun**; Chan, Kevin; Manjunath, B.S.; Govindan, Ramesh; "Caesar: Cross-camera Complex Activity Detection", ACM SenSYS 19.

Celso M. de Melo, Brandon Rothrock, Prudhvi Gurram, **Oytun Ulutan**, B.S. Manjunath, Understanding the Value of Synthetic Data for Vision-Based Gesture Recognition Submitted to IJCV 2019

S. Kumar, C. Torres, **O. Ulutan**, A. Ayasse, D. Roberts, B.S. Manjunath. Deep Remote Sensing Methods for Methane Detection in Overhead Hyperspectral Imagery, under submission WACV 2020

ShreeRanjani SrirangamSridharan, **Oytun Ulutan**, Shehzad Noor Taus Priyo, Swati Rallapalli, Mudhakar Srivatsa, "Object Localization and Size Estimation from RGB-D Images", Arxiv, Pre-print.

Abstract

Attention Models for Activity Detection

by

Oytun Ulutan

Video action detection is an important part of video understanding and analysis. There are many possible applications such as smart home environments to recognizing user actions and acting, smart robotics such as autonomous cars, robot assistants, selfie drones following your gesture commands, automated security systems analyzing the environment and assessing events. This thesis focuses on introducing novel machine learning algorithms for video action detection. A central contribution of this research is in developing a context-aware attention model for atomic actions. An atomic action is a simple action which can be described with 1-3 words or atomic body movements such as walking, drinking, holding an object. While observing actions/activities, humans infer from the entire context and our perception depends on the surrounding objects, actors, and scene. Inspired by this, our Actor Conditioned Attention Maps(ACAM) model utilizes the surrounding scene for each actor and uses context for improving action/interaction detection. The modularity of the ACAM model allows us to detect, track and recognize actions over extended time periods. We further extend this framework to detect complex activities which are composed of sequences of atomic actions. We demonstrate the effectiveness of our proposed methods on aerial videos and videos from camera networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the proliferation of camera networks, automation of video analysis became a critical necessity for security systems. In the current security frameworks, videos are used in two ways. First way is to analyze the recordings of the videos after an incident. These recordings are helpful to understand and identify the incident, but this is not a preventative approach. Secondly, an human analyst watches the security videos real-time from a number of camera feeds and looks out for suspicious activities. This is a challenging task for humans as analyzing subtle activities which may span over multiple camera feeds in a network is a mentally demanding task. Additionally, noticing abnormal events just by watching the feeds over long periods of time is a difficult task. These issues can be addressed by automating the video analysis using machine learning frameworks trained for tasks such as object detection, tracking, action detection. This thesis introduces novel methods for the detection and recognition of such actions.

Action detection is an important part of video understanding and analysis. There are many possible applications such as smart home environments to recognizing user actions and acting, smart robotics (such as autonomous cars, robot assistants, selfie drones) following your gesture commands, automated surveillance systems analyzing the

environment and assessing threats. To achieve these goals, this thesis focuses on action detection from videos.

Compared to more traditional problems in computer vision such as object detection, re-identification, action detection task is less explored. This is due to the lack of available large-scale datasets which can generalize to different input settings. Most of the available action detection models focus more on video classification task and classify very context dependent classes (e.g., apply eye makeup vs basketball dunk).

Recently atomic action detection tasks has received much interest. An atomic action is a simple action which can be described with 1-3 words or atomic body movements (e.g., walking, holding an object, etc.). These actions describe every-day actions and can be used as a basis to generate more complicated actions.

Previous atomic action detection methods followed models from object detection and analyzed actors individually analyzing actions similar to object task. In object detection, the object bounding box contains the defining features of the object. However, in action detection the bounding box contains the actor and the action boundaries are not as well defined. Hence, this approach does not explicitly use the surrounding context which is essential in action understanding.

While observing actions/activities, humans infer from the entire context and our perception depends on the surrounding context such as objects, actors, and scene. In order to model the context, we propose to generate attention maps from each actor, which models that actor's interaction with the surrounding objects, actors and scene. Hence these attention maps represent the surrounding context while being conditioned on each actor individually. We call this method Actor Conditioned Attention Maps.

Combining the context-aware action detection model with actor detection and tracking, a video analysis framework can be obtained. These frameworks will allow wide range of applications including security video analysis and gesture commands.

A sequence of atomic actions can be combined into larger and more complicated activities/events. This is one of the advantages of atomic actions as they are defined to be simple and form a basis for more complicated actions. Frameworks which can detect and combine such actions into complex activities can play a significant role for security applications.

Object/actor detection is essential in action detection networks. The first step of action detection is to detect the actor. However, due to the differences in views, such as drone recorded videos, security cameras, autonomous car mounted cameras, this task gets challenging as well.

This thesis implements frameworks and novel methods to analyze actions from videos. These frameworks are implemented while keeping applications in mind and several application examples and implementations are also introduced.

1.1 Motivation

The usage of video cameras are growing significantly across the world. Several cities in China have millions of surveillance cameras and constitute the largest density of cameras per population. Table 1.1 shows the number of cameras in some of the most surveilled cities in the world.

City, Country	# Cameras	Population	Cameras/1000people
Chongqing, China	2.57M	15.3M	168
Shenzhen, China	1.93M	12.1M	159
London, UK	627K	9.1M	68
Atlanta, GA	7.8K	501K	15
Chicago,IL	35K	2.7M	13

Table 1.1: Number of security cameras in some of the most surveilled cities in the world. Data is taken from [17]

Current security cameras are used at traffic signals/main junctions, residential areas, for crowd security such as festivals and inside office buildings. Each one of these cameras have different use cases and different applications. Currently most of these systems are analyzed manually by a human analyst. Manual analysis have the following challenges.

- Manual analysis is a tedious and tiresome task for humans. Over long periods of time, the human analyst is expected to watch the video feeds and be alert of events.
- The events of interest can span across multiple cameras which adds another layer of difficulty for the human analyst.
- Analysis of crowds and detection/threat assessment of activities is challenging for humans as there will be multiple events happening simultaneously.
- It is difficult for analyst to detect subtle events. Manual analysis can only detect significantly obvious and visual events (e.g., fire, shooting, fights etc.)

These challenges cause the current security systems to be used for investigative purposes rather than preventative. A surveillance overview from Australian Government [21] concluded that in the current systems videos are of more value as evidence. One added benefit of security cameras it the psychological effect. People tend to commit fewer crimes knowing that they are watched.

Due to these reasons, current systems can benefit significantly from the automation of video analysis. Detecting actions is an essential step in video understanding. Tasks such as object detection, tracking builds up the ground work. However, action detection and activity understanding is the main step towards automating the surveillance applications.

Automated action detection can also be used to protect the privacy of the regular citizens. Videos provide massive amounts of personal information about citizens to monitoring departments. Automation of surveillance can improve these privacy concerns by

anonymizing (obscuring/blurring faces) the identities of people with normal activities. This requires an understanding of normal and criminal activities which can be detected by action detection frameworks.

1.2 Challenges and Contributions

1. Context-aware Atomic Action Detection: Atomic action detection is an important task for video analysis with many potential applications. As previous action datasets focused on video classification tasks such as sports videos (e.g.: basketball dunk vs golfing), previous research on this area have been limited and not transferable to many settings. Atomic action detection task addresses these issues as they can happen in any background and allows transferability. This adds another layer of challenge as basic clues such as presence of a basketball cannot be used for detecting atomic actions and models have to learn the temporal dynamics and contextual relations of these actions/interactions.

Initial models on atomic action detection followed the architectures from object detection and modeled actors individually. Unlike object detection, actors interact with their surroundings and action detection usually requires contextual information rather than just the actor. The contextual information can include the background, surrounding objects and other actors. In order to detect atomic actions effectively, models need to learn surrounding context explicitly.

Atomic action detection task also includes detection/localization of the actors. Previous neural network models detected the actors by adding a region proposal network to their overall model and learned the actor boxes during the action detection training. This approach limits the transferability of the final model. Action detection datasets do not have a large variance in actor views as they are mostly taken from a single type of videos

(e.g., movies). Additionally, embedding the actor detection in the framework prevents transferability to other settings. A modular system on the other hand, can replace the actor detector with a specialized detector during inference.

Chapter 3 addresses these challenges by presenting an attention model for atomic action detection. This chapter focuses on implementing an atomic action detection model which can model interactions of the actors effectively and can transfer to various input settings (e.g., surveillance videos, autonomous car videos, etc.). The contributions of this chapter are as follows:

- Actor Conditioned Attention Maps for context-aware action detection: Instead of following the object detection methods, we propose to use attention models to model the actor. The proposed attention model is conditioned on the actor and models the actor's relationship with the surrounding context.

- Object detectors as Region proposal networks: Actor detection task is essentially object detection with a single class. Instead of embedding a region proposal network to our framework, we use pre-trained object detectors and show that the modular system can transfer to other settings.

2. Real-time system for Action Detection : Action detection is a complex task which contains multiple steps. Understanding human actions in a video stream depends directly on the frameworks ability to detect objects and humans, understanding human dynamics and analyzing the scene. To analyze a human's action, the person initially needs to be detected and tracked. Previous approaches to action detection either simplified these steps by centering the videos on a single person performing a single action or used predefined person boxes to focus on action detection.

Previous action detection methods followed an end-to-end approach where the models had to detect the actor and recognize the action simultaneously. This is challenging as

the detection of actor locations and understanding the actions in videos are disparate tasks. Additionally, actor detection models learned on action datasets have limited transferability to other settings. Due to these reasons, a modular approach is more suited for these tasks where each module is specialized in their own tasks.

Previous methods for activity detection task have tried to exhaustively list high-level activities as action classes and trained one-vs-all classifiers. This approach is impractical in surveillance analysis setting as there could be combinatorially many activities. To address this, we break up complex activities as sequences of atomic actions. Utilizing our systems, atomic actions can be detected and combined for complex activities. Additionally, spatial movements and relations between objects provides important information about the actions in the scene. These rule-based spatial methods can be combined with the learned neural network models to detect actions effectively.

Chapter 4 addresses these issues by presenting a modular approach to action detection. This chapter combines the atomic action detector with object detection and tracking modules and analyzes the system. The system achieves real-time performance while combining multiple, computationally expensive modules. The system is tested on various application cases. Building on top of this system, this chapter completes the complex activity detection pipeline by combining the atomic action detection with a rule based model analyzing spatial relations between humans and objects. The contributions of this chapter are as follows:

- Modular real-time system for action detection: We implement a modular approach where each of the modules are trained specifically on their own task. Our system for action detection combines modules for 1) object detection models trained on object detection datasets, 2) trackers trained on human tracking problems and 3) action detection models trained on action detection datasets.

- Various applications for action detection: We implement this module on tasks of atomic action detection and gesture detection. Atomic action detection system provides a general use action detector which can be used in various settings such as surveillance, webcam, car-mounted cameras, etc.

- Complex Activity Detection for surveillance video analysis: We combine the atomic action detection models with a rule-based method which analyzes the spatial relations of objects and movements. These detections are combined over time to detect complex activities which completes the action detection pipeline.

3. Spatial Attention for Human-Object Interactions: Comprehensive visual understanding requires detection frameworks that can effectively learn and utilize object interactions while analyzing objects individually. This is the main objective in Human-Object Interaction (HOI) detection task. In particular, relative spatial reasoning and structural connections between objects are essential cues for analyzing interaction. For modeling interactions, in this chapter we study Human-Object Interaction detection from images task.

Human-object interaction is an important task for visual understanding. Datasets for video action detection usually don't have explicit interaction labels and focus on actors. Even though interaction classes exist in video datasets, they don't label the interacted object. In order to learn such interactions, we focus on detecting human-object interactions from images. Even though this input is missing the temporal dynamics, they are usually exhaustively annotated for objects and their interactions with humans. In this setting spatial configurations and structural relations provide essential information for detection.

Chapter 6 studies the Human-Object Interaction detection task on images. Building upon the rule-based spatial configurations from the previous chapter, we propose a novel

neural network model which can utilize the spatial configurations of the human-object pairs and their structural connections. The model uses spatial configurations to generate attention on each human-object pairs. The contributions of this chapter are as follows:

- VSGNet, Spatial Attention Network for Detecting Human Object Interactions Using Graph Convolutions: VSGNet for HOI detection refines the visual features using spatial relations of humans and objects. This approach amplifies the visual features of spatially relevant pairs while damping the others. Additionally, this model uses graph convolutional networks to model the interactions between humans and objects.

4. Low-Resolution Long Range Person Detection: Object detection is a frequently studied problem in the computer vision with well annotated, higher resolution object detection datasets. These models work well on many scenarios and can be used "out of the box" for inputs such as webcam recordings, mobile phone photos, online videos etc. However, in surveillance applications visual conditions are usually challenging and such object detection methods don't transfer well due to lower resolution, distance from the cameras and weather conditions. In such conditions, specialized models are needed to be trained for detecting objects of interest.

In security applications such as border surveillance, a very large area (1000s of miles) need to be watched over very long periods of time. In such applications, due to bandwidth and power limitations, the surveillance cameras need to be placed close to computational resources and on strategic locations, such as on top of the hills. These visual conditions prevent robust detection from visual sensors only as the targets will have a size of couple of pixels (approximately 40 pixels). In such conditions, additional modalities can help vision for detecting the targets.

Chapter 7 focuses on the surveillance task of detecting humans in a very large area

by leveraging multi-modal sensor data. Low resolution visual videos are fused with a low-cost seismic sensor recordings to detect humans walking in the field. Low-resolution videos are recorded from a high resolution camera with a wide field of view (FOV) which is placed close to a power source but far away from the field with targets. This requires visual detection frameworks to search for small (few pixels) objects on a large field. Seismic sensors on the other hand can provide reliable information about their close surroundings and can easily be distributed on a large field. This allows the data from a seismic sensor to improve the detection of cameras in regions where camera view and sensor range intersects. The contributions of this chapter are as follows:

- A bilinear model for multi-modal fusion: We propose a novel fusion method for multi-modal setting. Our model extracts convolutional features from each modalities and combines the features using matrix outer product while preserving the spatio-temporal order.
- Multi-modal Person detection from low-resolution long range cameras and seismic sensor: We evaluate our multi-modal fusion method on a novel application, combining low-resolution cameras with seismic sensor for person detection.

1.3 Organization

This dissertation is organized as follows:

- **Chapter 2, Overview of Methodologies** provides an overview of detection tasks such as object detection and action detection, summarizes available datasets and models and motivates the need for the work discussed in this dissertation.
- **Chapter 3, Context-Aware Attention Model for Action Detection** summarizes action datasets, introduces the model for atomic action detection which

uses context-aware attention and demonstrates its performance.

- **Chapter 4, Real-time System for Action Detection** combines the atomic action detection module with object detectors and trackers and implements a real-time framework for atomic action detection. Additionally, action detection from this framework is combined temporally to detect complex activities which are sequences of atomic actions.

- **Chapter 5, Spatial Attention for Human-Object Interactions** introduces the spatial attention model which utilizes the spatial configurations of the human-object pair to refine the visual features.

- **Chapter 6, Person Detection in multi-modal setting** follows the object detection in challenging settings task and introduces a person detection model in multi-modal setting from long range, low-resolution cameras and seismic sensors.

- **Chapter 7, Discussions** re-iterates the main concepts of this dissertation, summarizes its contributions, and provides an overview of potential future directions.

Chapter 2

Background on Action Detection

2.1 Introduction

This chapter summarizes the previous action detection methods, datasets and details about relevant tasks such as image recognition and object detection. There have been two major factors that significantly improved the research in computer vision. These factors are 1) the availability of large-scale datasets collected from/annotated by online communities, and 2) parallelization of computations by using Graphical Processing Units (GPUs) and deep neural networks.

Early datasets were usually gathered by research labs individually and was limited in size with a few samples (10-100). The major challenge in this setting is collecting and annotating the data. As these were usually smaller labs, researchers tried acting out and collecting the data themselves which was time consuming and impractical for large scale datasets. Most of these issues were addressed when both of these tasks are crowd-sourced on online communities. With the large image websites such as Flikr, Instagram getting more popular, larger scale image datasets were able to be collected, with the number of collected images exceeding several millions. Additionally, annotation tools such as

Amazon Mechanical Turk allowed researchers an easy way to get annotators and label the large-scale data quickly.

Early models for computer vision tasks depended on hand-crafted features and simple machine learning methods. Hand-crafted features such as SIFT [22], edges, blobs required a higher understanding of the physics behind the input data and was difficult to generalize to other tasks. Additionally, simple machine learning methods would struggle to model basic image characteristics such as spatial coherence. With the introduction of GPUs and parallelization of the machine learning methods, such limitations were addressed and computer vision methods started achieving human-level labeling accuracy. One of the first models that implemented Convolutional Neural Networks (CNNs) on GPUs is AlexNet[23] in 2012. This model used a variant of CNNs of Lecun [24] from 1989 and a variant of back-propagation algorithm by Fukushima[25] from 1980. The theory for such methods go back to 1980s but haven't been significantly effective until 2012 thanks to the convergence of high power GPUs and availability of large scale well annotated data..

In the following sections, we will overview the relevant major datasets and methods for Image recognition, Object Detection and Action detection tasks.

2.2 Image Recognition

Image recognition is a task where the machine learning models are required to recognize the categories of the images. A simple example for this is classifying images of apples and oranges. There have been many different tasks in this area from handwritten digit recognition to scene recognition.

Figure 2.1: Examples from MNIST [1] dataset. The task is to recognize hand written digits and this dataset acts as a initial step to recognize handwritten text.

2.2.1 MNIST

MNIST [1] is one of the first larger datasets available in computer vision. Dataset includes 60,000 training images and 10,000 testing images. The main task is to detect handwritten digits. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting [1]. This dataset acts as a initial step to recognize handwritten text.

2.2.2 ImageNet

ImageNet [2] is the first large-scale in the computer vision community. Currently, the dataset contains 14 million images and grows every year. Main benchmark has 1000 image categories with each category having at least 1000 samples. Initially, the dataset targeted image recognition task with a wide range of categories such from huskies to sailboats. Subsequently, several object detection tasks are added. Object detection is a task which includes recognition of the object and localization of the object within the

Figure 2.2: Examples from ImageNet [2] dataset. The image recognition task includes very fine-detailed classes such as different breeds of dogs.

image.

One major contribution of this dataset is the transferability of learned models. For many vision tasks, such large-scale datasets are not easily available. This poses a challenge for these tasks as deep convolutional networks require large amounts of data for training. However, it has been shown by many works [26, 27, 28, 29, 30, 31, 32] that model weights trained on ImageNet dataset is able to provide good initialization for other tasks. Sometimes without any further fine-tuning, ImageNet models can extract features useful for other tasks where simple classifiers can solve these objectives.

above generalization is often referred to as transfer learning. In deep CNNs lower layer convolutions capture low-level image features such as edges, blobs, patterns and colors while higher layer convolutions capture more complex structures such as body parts and faces. While these higher layer structures are very task specific, the lower layer features are useful for many different tasks. This allows the lower layer features trained on ImageNet to be used directly with new classifier layers for various tasks.

As the first large scale dataset, ImageNet made a very significant impact on the computer vision community and even today is used for pre-training for many tasks.

2.3 Object Detection Datasets

Object detection is a task which includes recognition of the object and localization of the object box. This task adds another layer of difficulty on top of the image recognition tasks as models need to additionally locate the object's location. This is a more realistic case as images can include multiple objects simultaneously.

There have been various datasets for this task. In this section, we will summarize 3 main datasets which are MS-COCO [3] dataset which focuses on general use object detection, KITTI [4] dataset which focuses on object detection from vehicle mounted cameras and VisDrone [5] dataset which focuses on object detection fron drone mounted cameras. Even though these datasets mostly annotated similar object classes, the viewpoints, angles, distances from objects are significantly different in each setting. Due to these different settings, models trained on one setting can not be directly used in another in a robust way.

2.3.1 MS-COCO

MS-COCO [3] is a large-scale object detection dataset. The dataset contains 330,000 images with around 200,000 of them annotated. In the first release, dataset included object bounding boxes for 80 object categories including classes such as "people", "car", "bike", "fork", etc with about 1.5 million object instances. Following many releases, currently the dataset includes multiple tasks which are object segmentation task for 80 classes, "stuff" (grass, wall, sky) segmentation task for 91 classes, caption generation task with 5 captions annotated per image and body pose detection with 250,000 people

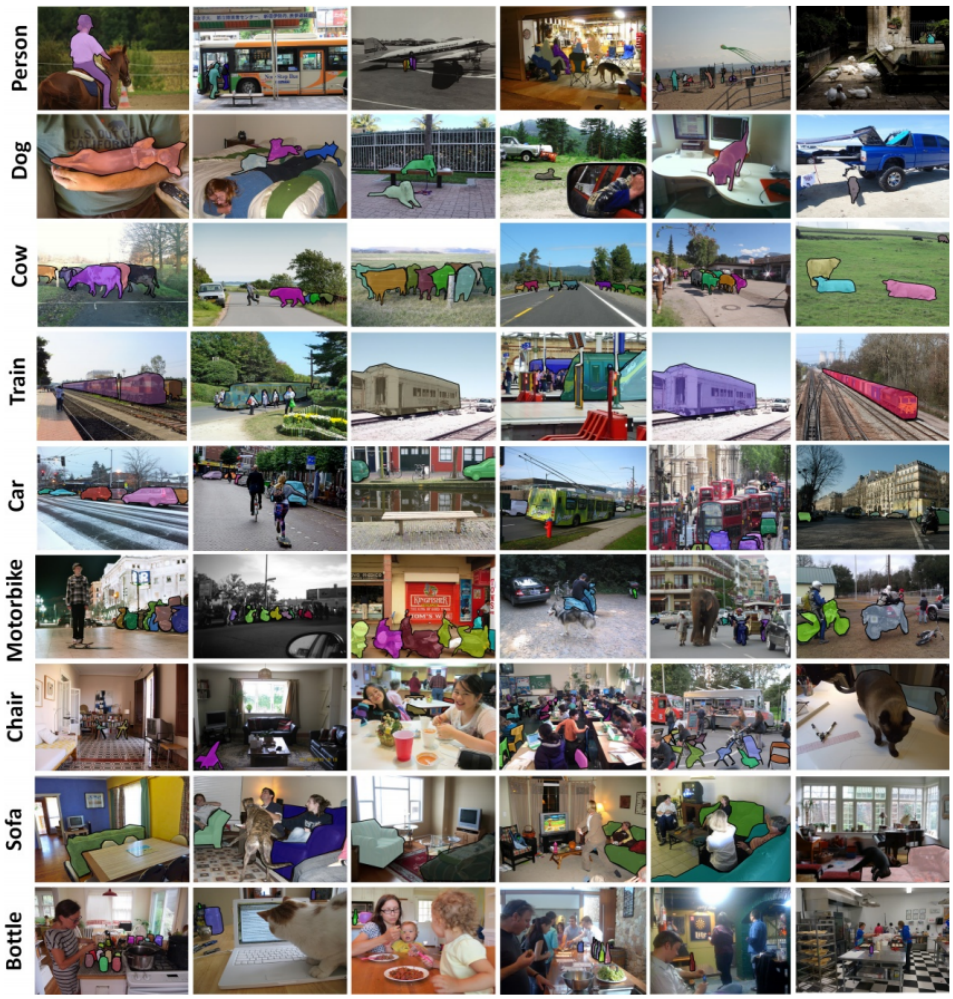


Figure 2.3: Examples from MS-COCO [3] dataset. The categories are widely used objects that can be captured in many different settings.

annotated with body keypoints.

Amongs these various tasks, models trained on object detection task made a significant impact. As the data is collected from many different sources, dataset includes various view point settings and the models can generalize well. Additionally, the object categories are widely used objects that can be captured in many different settings. This allowed many methods to implement out-of-the-box object detection inference systems which are available for public use [33, 34, 19].

Figure 2.3 shows some of the object classes and samples from the dataset. 2019 release

and challenge includes the following tasks:

- **Task 1: Object Detection.** This task includes the localization of 80 classes of objects in images and generating their segmentation maps.
- **Task 2: Keypoints Detection.** This task includes the localization of body keypoints for each person in the images. Keypoints include parts such as hands, arms, heads, legs etc.
- **Task 3: Stuff Detection.** This task includes generation of segmentation maps for "stuff" from images. Stuff categories include things such as "grass", "wall", "sky" etc.
- **Task 4: Panoptic Detection.** This task combines the object categories with stuff categories and generates a segmentation task where the images have to be semantically segmented completely.

2.3.2 KITTI

KITTI [4] is a dataset recorded with vehicle mounted cameras. The main goal of this dataset is to obtain data for implementing self-driving car perception modules. The dataset includes many tasks including 2d object detection, 3d object detection and object tracking, etc.

Figure 2.4: Examples from KITTI [4] dataset. Cars are recorded from many angles and at various illuminations.

The object detection task consists of 7481 training images and 7518 test images, comprising a total of 80.256 labeled objects. For the 3D object detection case, in addition to locating a bounding box for an object, orientation estimate in bird's eye view is also needed to be detected.

The dataset annotates 8 object classes but only evaluates the models on 2 main classes which are vehicle and pedestrian. Remaning classes are annotated for future use. Figure 2.4 shows the vehicle class from various angles and illumunations.

2.3.3 VisDrone

VisDrone [5] is recent a large-scale benchmark drone-view dataset. The dataset consists of 263 video clips formed by 179264 frames and 10209 static images all of which captured by drone-mounted cameras. Data is recorded in 14 different cities in China and in various urban and suburban environments. Figure 2.5 shows example images from the dataset and shows the variety in environments, drone altitudes, view-angles and illumination conditions.

VisDrone benchmark focuses on 4 different tasks.

- **Task 1: Object Detection in Images.** This task focuses on detection of objects (e.g., cars and people) from single images.
- **Task 2: Object Detection in Videos.** This task focuses on detection of objects (e.g., cars and people) from videos recorded from drones. Task uses per-frame detections and does not include tracking.
- **Task 3: Single Object Tracking.** This task focuses on tracking a given object over a period in a video. The object is indicated in the first frame.
- **Task 4: Multi Object Tracking.** This task focuses on tracking multiple objects

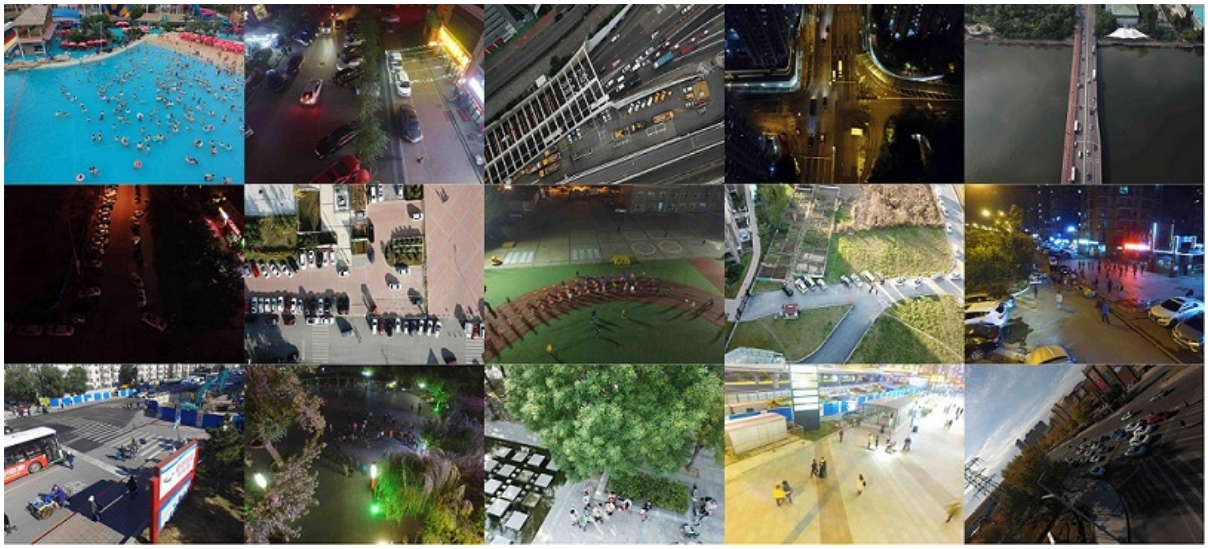


Figure 2.5: Example images from the VisDrone [5] dataset. Videos and images are recorded from drones. Data is recorded in various urban and suburban areas in China including many large cities.

in a video. Two subtasks focus on recovering trajectories with or without detection results.

The dataset annotates the following object classes in the drone-recorded videos:

Objects: Pedestrians, People, Cars, Trucks, Vans, Buses, Motors, Bikes, Tricycles.

One design choice for the dataset is to split pedestrians and people as separate categories. *Pedestrian* are humans who maintain a standing pose or walking compared to *Person* class which might be sitting (e.g., on a bike, on a bench) or in any other poses (e.g., basketball, swimming).

2.4 Overview of Object Detection Methods

Deep neural networks have achieved much success in object detection task. The "Regions with CNN Features" (R-CNN) [30] approach was the first modern CNN archi-

tecture for detecting objects. R-CNN follows a straightforward approach where a fixed image recognition CNN extracts convolutional features from each box proposal and these features are classified by an SVM. However, this approach was not efficient as multiple calculations on overlapping region proposals are done separately. Addressing this issue, Fast-RCNN[35] extracted CNN features over the whole input image once and then cropped the regions on top of the features. This operation is called region of interest pooling. This reduced the redundant calculations and improved the performance and accuracy simultaneously as better features are extracted.

Neural networks can also be used to generate box proposals which can even make this operation faster as in Faster-RCNN [34]. In this architecture all possible box locations for the images are needed to be analyzed to produce region proposals. These box locations are called anchors and they are laid across all the spatial locations of the image and vary in scales and aspect ratios. This allows different shaped object to be detected with spatial flexibility.

Following these works many different model architectures were studied. As analyzed in [19], most of these works follow three main meta-architectures. These are Single Shot Detector (SSD), Faster RCNN and Region Based Fully Convolutional Networks (RFCN) architectures.

2.4.1 Single Shot Detector (SSD)

SSD [36] is a simpler method compared to Faster RCNN. SSD architecture directly uses the RPN stage of Faster RCNN to classify anchors and does not use a specialized second stage for per proposal classification. For each anchor, SSD directly runs the multi-class classification and regression. This allows the model to run quickly in a single run.

As the anchors are directly classified in SSD compared to a 2-stage Faster RCNN, SSD does fewer computations to analyze each given image, uses less memory and runs faster. SSD modules with MobileNet [37] can be used for mobile applications and run on cell phones for inference [37].

2.4.2 Regions with CNN features (R-CNN)

Faster RCNN [34] model builds on top of the RCNN [30] architecture. Fast RCNN [35] and Faster RCNN [34] removes the SVM part, introduces Region of Interest pooling on CNN features and combines region extraction with classification in a single end-to-end network.

In Faster RCNN models, the detection happens in 2 main stages, region proposal network (RPN) and box classifier stages. The first stage, the region proposal network (RPN) generates boxes which are likely to have an object in them, i.e. box proposals. These box proposals are generated from "Anchors". Each anchor gets scored on their objectness using the feature map. After a non-maximum suppression which filters out redundant boxes, remaining high objectness score anchors are sent to box classifier stage as region proposals. The second stage, the box classifier stage takes the high scoring box proposals and crops their feature from feature map using Region of Interest (RoI) pooling. RoI pooling takes a box coordinate and extracts a fixed size feature vector from the convolutional feature map defined by this box coordinate. This allows different shapes box proposals to generate a fixed size feature vector which is then further analyzed by the classifier. Box classifier takes the RoI pooled feature vector and generates classification scores for each of the available object classes.

2.4.3 Region Based Fully Convolutional Networks (R-FCN)

Even though Faster-RCNN improves on Fast-RCNN methods achieves better speed rates, region box classification is repeated for each proposal. R-FCN addresses this issue and instead of applying convolutions individually to each region, the regions are cropped from the final layer. This minimizes the amount of per-region operations done and speeds up the model.

These detectors are further analyzed in Chapter 4.3.1 and quantitative results are provided for MS-COCO dataset.

2.5 Action Recognition and Detection Datasets

Action recognition and detection focuses on human actions and their interactions with the surrounding environment which can include other people, objects and scene. Most of these action datasets include videos and require modeling of temporal dynamics for actions which adds an additional layer of challenge in video annotation compared to image tasks. For human annotators it is more clear to annotate types of objects compared to an video including an action as temporal understanding requires more cognitive load.

Initial datasets such as [38] were simple where subjects acted out basic actions like running, jumping in with a simple background. As the online video communities such as YouTube started to emerge, it became easier to access videos which increased the scale of the video datasets. With such large amounts of videos available, computer vision datasets start to grow also.

Most popular action recognition datasets contain short clips which are trimmed manually to capture a single action. These datasets are used for the action recognition task where the task is to categorize a given video into action classes. Sports events are a good example to such videos. Most event videos are uploaded as soon as the event happens

and these videos are mostly labeled by the type of their sport (basketball, football). This generated an easy way to gather sport videos and make a dataset with them. UCF101[6], Sports-1M[39], ActivityNet[11] are example large-scale datasets collected from YouTube and include sports action classes amongst others.

A recent Kinetics-400/700 [8] datasets increased the number of videos significantly and included more than just sports videos and included 400 (700) classes. This dataset became the ImageNet-like pretraining backbone of video classification.

Action detection tasks may include spatial, temporal or spatio-temporal localization of action instances from videos. A few datasets such as UCF101-24 [6], JHMDB [40], Hollywood2Tubes [41] provided spatio-temporal annotations for the actors who are doing the action. This extends the action classification into action detection where localization of the actor adds an additional challenge. However, for these datasets, the action vocabulary is still restricted to a limited number of composite actions (e.g., basketball dunk). Moreover, they do not densely cover the actions such that only the main actor is annotated whereas the people in the background do not have any annotation.

A more recent dataset, AVA [12], aims to solve this issue. This dataset annotates atomic actions, exhaustively annotates every actor in the scene and annotates multiple existing actions to each actor.

Action and interactions can also be detected in still images. In this setting rather than temporal dynamics of actions, spatial relation between objects is captured. Recent datasets such as V-COCO [9] and HICO-Det [42] annotated the object detection datasets with human and object interactions as these images usually include humans interacting with objects.

Figure 2.6: Example action classes from the UCF-101 [6] dataset.

2.5.1 UCF-101

UCF-101 [6] is a video action recognition dataset which focuses on different classes of actions from Youtube videos. The dataset contains 101 action categories with a total of 13,320 videos. As these videos are taken from Youtube, it includes videos from many different settings from sports videos to "how-to" videos. Some example actions from this dataset is shown in Figure 2.6.

One major issue with the dataset is that the context has most of the information about the video class compared to the temporal dynamics of the actions that are happening. For example, detecting a swimming pool certainly says that action is swimming compared to basketball. This resulted in detection models which are not transferable to other contexts and mostly doing video classification rather than action classification. As a result, if a classifier is trained to detect the video class from a single frame from these videos, they can achieve about 80% accuracy.

2.5.2 Human Motion DataBase HMDB

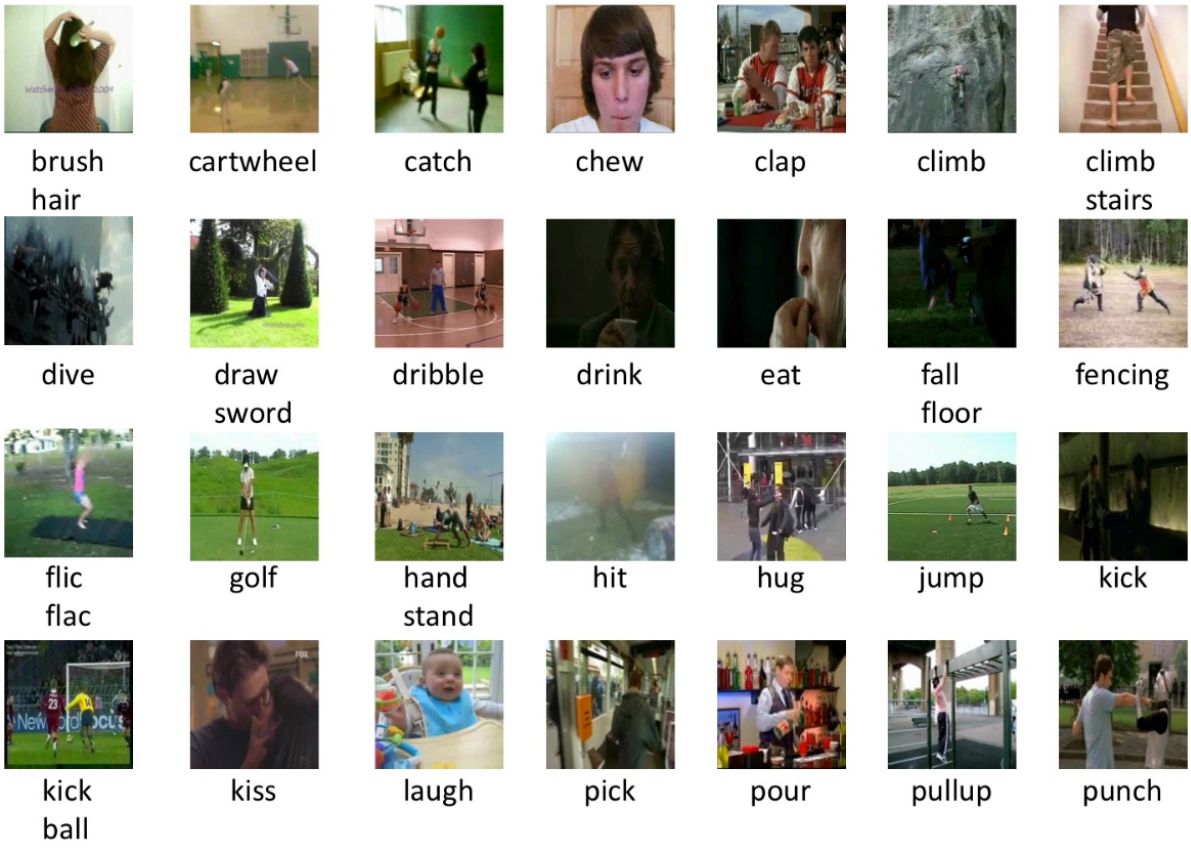


Figure 2.7: Example action classes from the HMDB-51 [7] dataset.

HMDB-51 [7] is a dataset that is also collected from YouTube and Google video search. The dataset contains 51 action categories and 6,849 videos in total. Each video category contains a minimum of 101 video clips. Compared to UCF-101, HMDB-51 dataset includes human actions which are more likely to happen in regular context such as picking up an object or drinking. Some actions are shown in Figure 2.7.

As an extension to this dataset, Joints-HMDB (JHMDB) dataset[40] annotated the joints of people in these videos. This dataset contains 21 actions and 928 videos from the HMDB dataset and uses a 2D human puppet to annotate different body parts on these videos temporally. Originally, this is was used to further improve action recognition

tasks but later it has been used for action detection tasks as well since it includes person localization information.

2.5.3 Kinetics 400/700

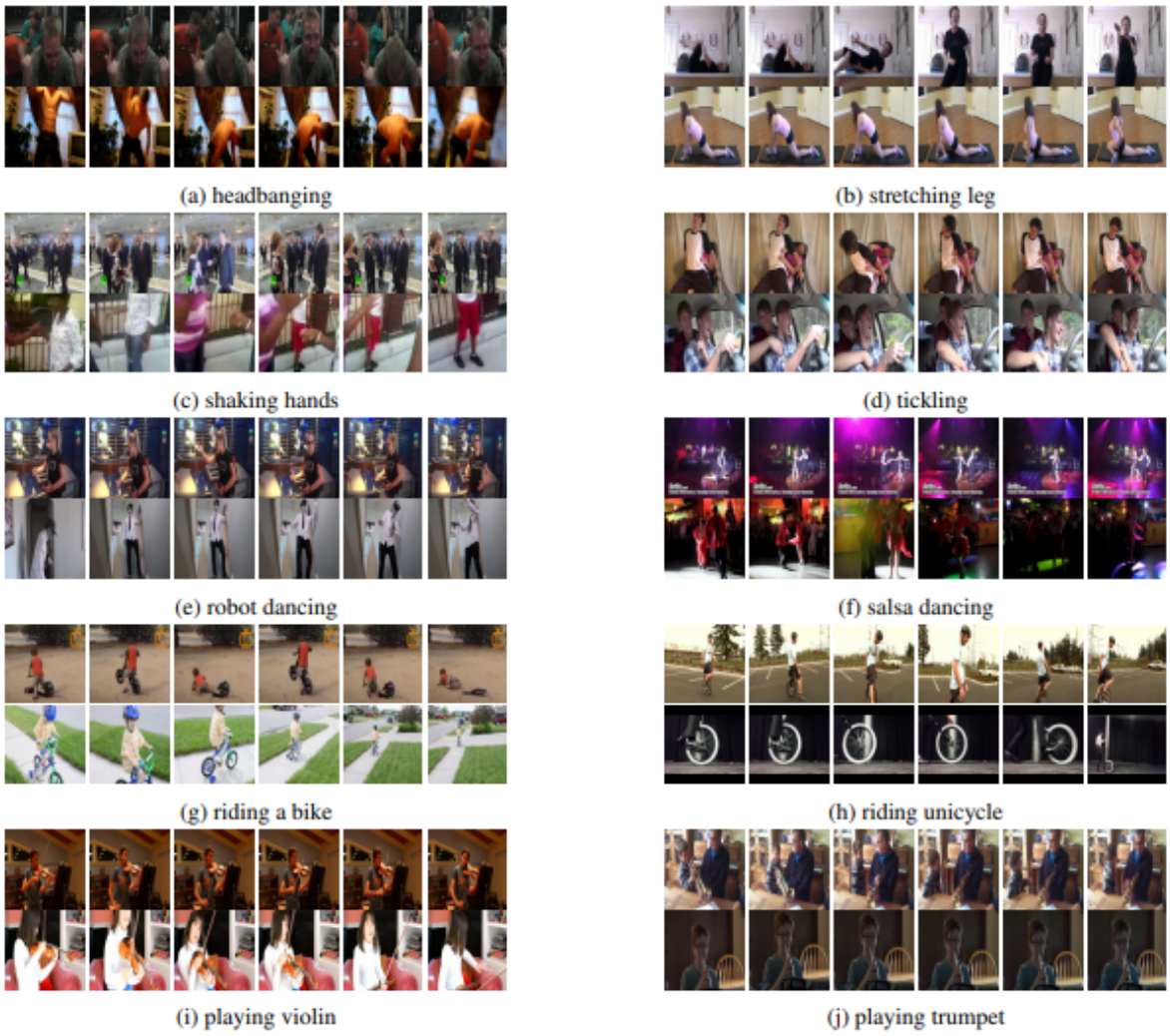


Figure 2.8: Example action classes from the Kinetics [8] dataset.

A recent Kinetics-400/700 [8] datasets increased the number of videos significantly and included 400 action classes initially then expanded to 700. This dataset became the first large-scale dataset where trained models are able to transfer to other datasets and

became the ImageNet-like pretraining backbone of video classification.

Kinetics-700 dataset contains 700 action classes which are human actions, human-object interactions and human-human interactions. Dataset has approximately 650,000 high quality video clips and each clip has a single action and is around 10 seconds. Figure 2.8 shows some example classes from the kinetics dataset.

As this is the first very large scale video classification dataset, the pre-trained models on this dataset have been widely used in many other video classification/detection tasks. Pre-training is essential in training good models which is especially difficult for video tasks as most dataset sizes can't achieve larger scales enought to train models from scratch. In a task where small number of videos are available a pretrained model can be fine-tuned or the features can be used directly for classification.

2.5.4 Atomic Visual Actions - AVA

AVA[12] dataset focuses on atomic actions within short video segments. An atomic action is a simple action which can be described with 1-3 words or atomic body movements. AVA argues that exhaustively listing high-level activities as action classes is impractical as done in previous datasets. For this reason, AVA limits actions to short time intervals (1-2 seconds) where actions will have a very clear signature. Following this AVA annotates movies at a 1 Hz rate. Below are the main properties of the AVA dataset.

- Annotates 80 person-centric atomic actions on movies
- Annotators watch the 2 second video segment, label person's bounding box in the center frame and label that person's actions within that video
- Multiple people in scenes with multiple action labels each
- 60 action classes are used for benchmarking chosen by sample size

The movies are taken from publicly available movies and from each movie multiple 15 minute chunks are annotated at 1 Hz. From a total 430 movies, this generates 386k labeled video segments, 614k labeled person bounding boxes and 1.62M action labels.

This dataset has 80 atomic action classes. These action classes fall into three main categories 1) person poses (e.g., walking, running, standing), 2)person-person interactions (e.g., talking to, hugging), and 3) person-object interactions (e.g., holding, carrying). 60 classes with more than 25 samples in validation and test sets are used for benchmarking and evaluation for detection models.

Atomic actions have the potential to transfer to different contexts, become building blocks for more complex actions and improve the general understanding of human actions/interactions in videos. For these reasons, in Chapter 3 we focus on this dataset and propose a novel action detection model from videos.

2.5.5 Verbs in COCO (V-COCO)

Verbs in COCO (V-COCO) [9] dataset annotates human object interactions on still images. This dataset builds on MS-COCO [3] object detection dataset. The dataset consists of 10,346 images. Withing these images, 2533 images are used for training, 2867 images are for validating and 4946 images are for testing. The training and validation set images are from COCO training set and the test images are from the COCO validation set. Each person in the images are annotated with a label indicating one of the 29 actions. If an object in the image is related to that action then the object is also annotated. Figure 2.9 shows couple example human-object interactions in this dataset.

Compared to video action detection tasks, this task explicitly annotates the interactions between humans and objects. For example, in earlier video datasets for a “carry a cup” action, cups are usually not annotated and only the actor is annotated. However,

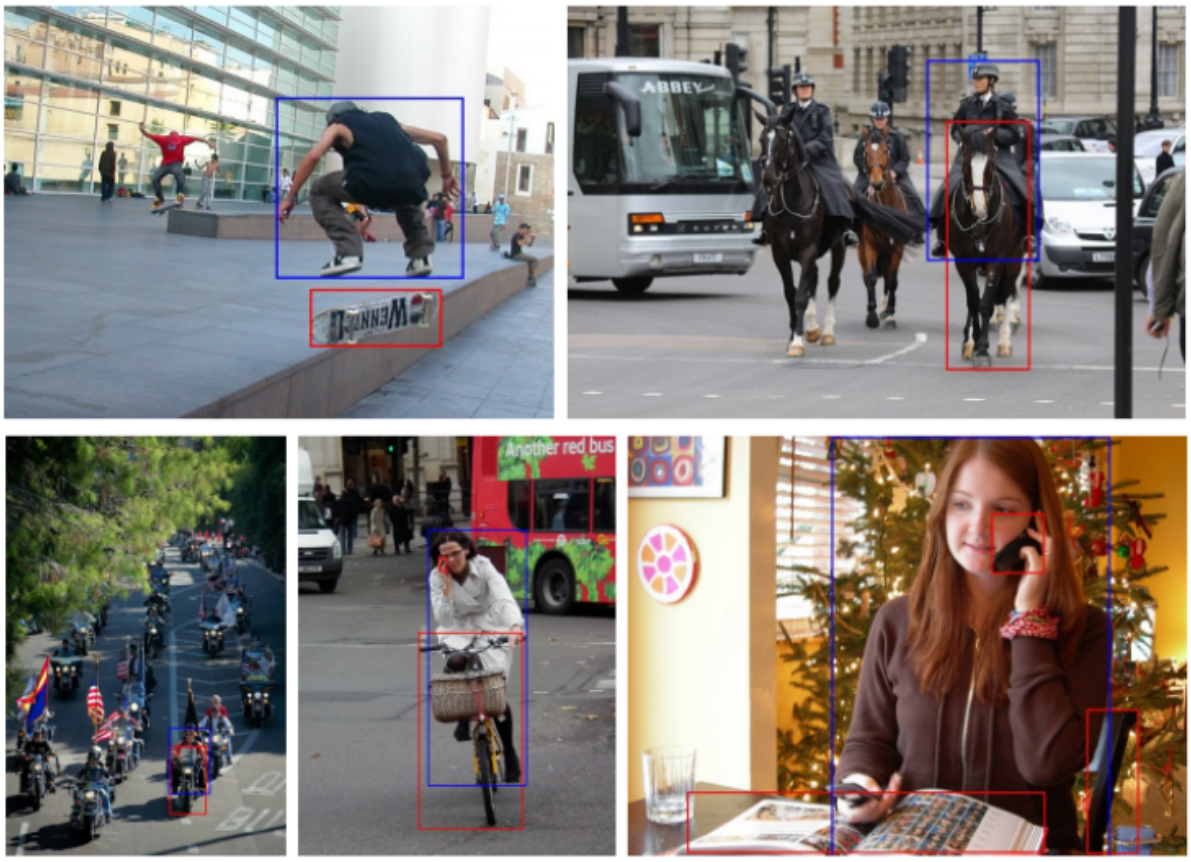


Figure 2.9: Example images from the V-COCO [9] dataset.

as this dataset uses still images, they were able to annotate such interactions explicitly which creates an interesting research problem.

2.6 Overview of Action Detection Methods

With the emergence of large-scale action datasets, various deep neural network models started achieving good results on this task. Compared to image based tasks, video action recognition is more challenging. Since the videos are sequences of 2D frames, each sample is already large in size and this poses computational cost for training models. As the goal is to capture temporal dynamics, models need to utilize the frames together

rather than individually and extract temporal information. This also creates an additional challenge in the design of classification architectures as different temporal analysis methods are possible. Most recent works address these challenges and improve the video action detection task. In this chapter, we provide a brief review of some of these models in the following.

2.6.1 Two Stream Networks

Initial models for video action detection were not able to train very deep networks efficiently. Due to this reason, Two Stream Networks [43] model simplified the model architecture and split the temporal information from spatial context. This model consists of two branches, spatial and temporal streams. The spatial stream only takes a single RGB frame from the whole video and analyzes the spatial context without any temporal information. The temporal stream on the other hand takes multiple frames of pre-computed optical flow images from the input video. This separates out the temporal information from spatial information and the branches are fused at decision level by averaging.

One major issue with this approach is that the temporal aspect is very limited. All K of the optical flow frames are considered as channels and the input to the temporal stream becomes a 2K channel input. This does not generalize as well as 3D convolutions as temporal coherence is not modeled but memorized. This also limits the temporal size of the input and because of that the inputs are sampled randomly/uniformly for K length intervals and the outputs are averaged. This ignores the long range temporal information. Also as this model uses optical flow that uses additional computations which are expensive.

2.6.2 C3D Model

C3D model [44] is one of the first models to use 3D convolutional layers for video action detection. Following a similar layering approach of VGG-16 [45], 16 layers of a network architecture is build utilizing 3D convolutions. 3D convolutions are essential for video understanding as they directly utilize spatio-temporal relationships of the input data.

In 3D CNNs, convolution and pooling operations are performed spatio-temporally. This generates a spatio-temporal output volume which models videos more effectively than compressing the temporal dimension. Additionally, C3D aims to achieve generalizable feature descriptors. They demonstrate that these 3D convolutions are more generalizable by using basic SVM classiffiers with the features extracted by this frozen 3D model on various tasks and achieving better results than models trained specifically on those datasets such as [6, 7].

2.6.3 Inflated 3D Convolutions (I3D) Model

Inflated 3D Convolutions (I3D)[46] model solves the issues of C3D and improves 3D convolutions further. I3D model builds on top of the Inception[47] models. Inception is a state of the art 2D-CNN model for image classification on ImageNet[2] dataset. I3D inflates these 2D convolutions into 3D by temporally extending every convolutional layer (e.g.: 3×3 Convolutions to $3 \times 3 \times 3$ Convolutions).

One challenge in training 3D CNNs is the increased number of parameters. Compared to image tasks, 3D CNNs add the third temporal dimension to all of the convolutions which increase the number of parameters by an order of magnitude. A further difficulty is the lack of pre-training. Image tasks can usually use pre-trained models on ImageNet [23] and fine-tune the model for their tasks. As such datasets are recent, there haven't

been many previous models. This caused models to initialize their model from scratch instead of fine-tuning pre-trained models. Due to these reasons, 3D video CNN couldn't reach the depth of 2D image CNNs until I3D architecture. As the I3D architecture is generated from the inception block, the 2D weights trained on ImageNet can also be extended into 3 dimensions. This extension allows the authors to start the training with ImageNet trained weights which provides a significantly better starting point than random initialization.

Compared to previous 3D CNNs, I3D model trains on the Kinetics 400 [46] dataset with diverse 400 classes. This task, compared to sports video classification, is a more generalizable task and the resulting model becomes more transferable to other datasets as demonstrated in [46].

2.6.4 Non-local Neural Networks

Non-local Neural Networks [48] is a recent paper which achieved good results in video action recognition. This work uses non-local operations which utilizes spatio-temporal relations between different indices. For each index in a feature tensor, its relations with every other spatio-temporal index is generated with an attention mechanism and using this relation and values on the feature tensor, a new compact feature vector is extracted for that index.

Non-local neural networks is a good example of attention mechanisms. Attention mechanisms quantify the relations between different locations/instances and generate new features using these relations. When used with a specific configuration, Non-local neural networks reduces down to the self-attention block from Attention is All You Need [49] work. However, in both of these works, spatio-temporal locations are taken individually without looking for a higher structure between them. To address these, in chapter 3 we

introduce our attention module which utilizes and actor's attention with the surrounding context rather than looking at every pixel value.

2.7 Discussions

Object detection and action detection are very relevant but disparate tasks. Action detection tasks usually include actor detection but also require a deeper understanding of the relations in the scene with surrounding actors, objects or the background. These relations contain the essential information about the actions/interactions happening in the input video. In the next chapter, building up on the ideas from object detection and action detection in videos, we present our context aware model which individually models the scene for each detected actor.

Chapter 3

Context-Aware Attention Model for Action Detection

3.1 Introduction

With the increased use of camera usage in cities, large volumes of videos are recorded from many sources such as security cameras, traffic cameras, drones and personal cell phones. Analyzing such large volumes of videos is a mentally taxing task for human analysts and inefficient for detecting complex events ahead of time. Additionally, more complex events can span over multiple camera views which adds another layer of challenge for analysts. In such setting automating the detection of activities becomes an essential need for security applications. Automation process is comprised of multiple steps. In a multi-camera network where the activities of people are being analyzed, computer vision tasks such as person detection/tracking, re-identification across cameras, action detection of people are needed to be combined. In this chapter, we will focus on context-aware action detection and recognition models.

In a video monitoring setting, the task is to analyze and detect complex activities

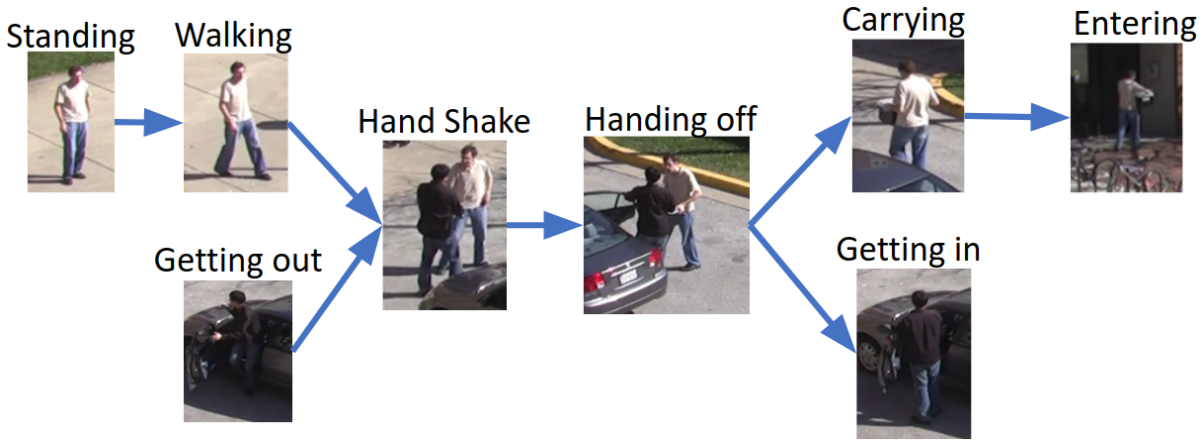


Figure 3.1: An example complex activity "box exchange". Complex activities can be broken down into a sequence of primitive actions for each detected person. Images are taken from VIRAT dataset [10]

which might happen over a long time interval and can include multiple people. We argue that such complex activities can be broken down into a sequence of primitive actions for each detected person. We call these primitive actions "Atomic Actions" as they provide a basis for more complex activities.

Atomic actions are comprised of "individual actions", "interactions" and "person states". This can include basic states such as "walking", "standing" and also more complex interactions as "hand shake", "giving an object". Defining atomic actions this way allows us to continuously analyze the detected people. Figure 3.1 shows a complex activity "box exchange" and breaks it down into sequences of atomic actions.

In order to analyze actions, people in the scene needs to be detected and tracked over-time. Achieving complex activity detection in an end-to-end framework with a single model is not a feasible solution as there are multiple separate but connected tasks. Therefore, we breakdown this task into multiple modules with individual goals. Figure 3.2 shows the breakdown of modules for the task.

Compared to action detection task, object detection/tracking, person re-idenfitication are more explored. There is a large selection of open-source models available for these

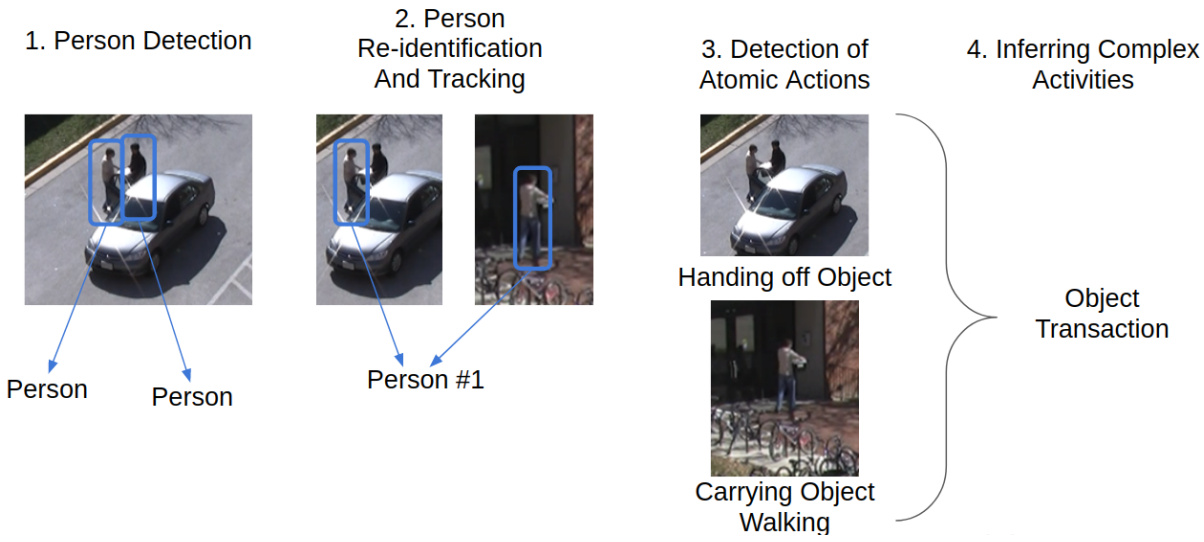


Figure 3.2: Breakdown of modules for detecting complex actions. A video analysis framework needs to 1) detect people in the scenes, 2) track and re-identify people across multiple cameras or different views, 3) detect atomic actions on detected people, 4) form sequences of atomic actions into complex activities. This chapter focuses on step 3, detection of atomic actions. Images are taken from VIRAT dataset [10]

tasks such as object detection [34, 36, 33], tracking [14, 50] and re-identification [15]. Most of these models are compatible with surveillance camera-views and can be used out-of-the-box for this task. However, the same argument cannot be made for atomic action detection as there are no models available which can generalize to surveillance models. One of the reasons for this is the lack of available dataset which can generalize to different input settings. Most of the available action detection models focus more on video classification task and classify very context dependent classes (E.g.: apply eye makeup vs basketball dunk).

A more recent dataset called Atomic Visual Actions (AVA) [12] focuses more on atomic actions. An atomic action is an action which can be described with 1-3 words and can be used as a basis to generate more complicated actions. In this chapter, we utilize the AVA dataset in order to develop atomic action models.

Previous models for atomic action detection followed architecture from object de-

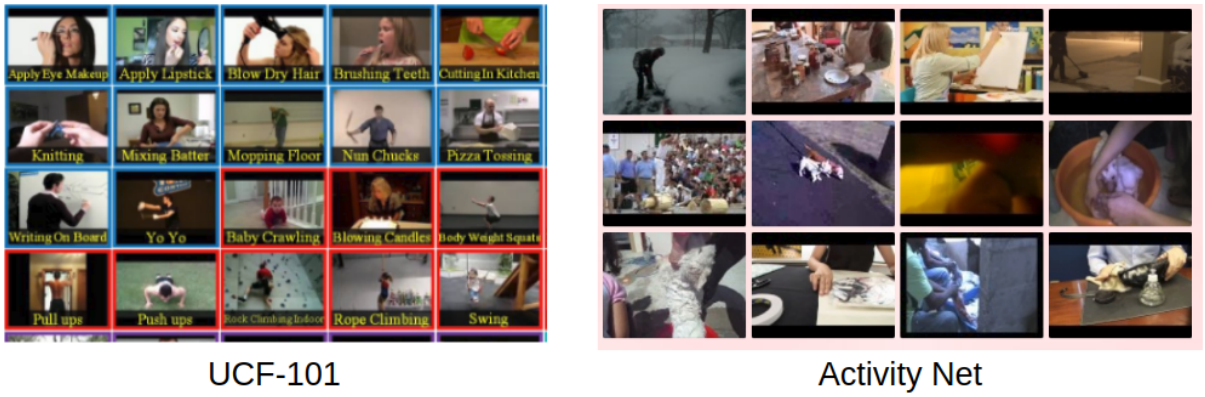


Figure 3.3: Example action classes from the previous action classification datasets. UCF-101[6] is on the left and ActivityNet[11] is on the right.

tection. Even though object detection and action detection tasks are closely related, a fundamental difference is that action/interaction detection usually requires the analysis of the surrounding context. In order to address this fundamental difference, in this chapter we present a novel action detection architecture which models the surrounding context for each detected actor. This is achieved by utilizing an attention mechanism which considers the current actor and the surrounding context instead of looking at the actor individually. Initial results of this research are published in [51].

3.2 Overview of Datasets

Detailed overview of the available datasets and methods are available in Chapter 2. This section provides a summary of previous datasets and provides additional details for the AVA [12] dataset.

UCF101[6], Sports-1M[39], ActivityNet[11] are example large-scale datasets collected from YouTube and include sports action classes amongst others. One major issue with these datasets is that the context has all the information about the video class compared to the actions that are happening. For example, detecting a swimming pool certainly

says that action is swimming compared to basketball. This resulted in detection models which are not transferable to other contexts and mostly doing video classification rather than action classification. Some example video classes from these datasets can be seen in Figure 3.3.

A recent Kinetics-400/700 [8] datasets increased the number of videos significantly and included more than just sports videos. This dataset become the first large-scale dataset where trained models are able to transfer to other datasets and became the ImageNet-like pretraining backbone of video classification. Dataset includes a large amounts of classes (400 or 700).

A more recent dataset, AVA[12] annotates atomic actions, exhaustively annotates every actor in the scene and annotates multiple existing actions to each actor. An atomic action is a simple action which can be described with 1-3 words or atomic body movements. AVA argues that exhaustively listing high-level activities as action classes is impractical as done in previous datasets. For this reason, AVA limits actions to short time intervals (1-2 seconds) where actions will have a very clear signature. Following this AVA annotates movies at a 1 Hz rate.

In AVA dataset, 80 atomic action classes are annotated. Figure 3.4 lists all 80 classes and the 3 main categories. 60 classes with more than 25 samples in validation and test sets are used for benchmarking and evaluation for detection models.

JHMDB [40] dataset is an older and smaller scale dataset which also annotates atomic actions and includes actor bounding boxes. JHMDB dataset is generated as a subset of HMDB-51[7] dataset. JHMDB annotates 21 action classes from the HMDB dataset where the most of the body of the person is visible.

In contrast to AVA, JHMDB does not include multiple labels, multiple actors or wide variety in backgrounds. It is also a much smaller dataset with 928 videos. However, since the objective is similar in the sense of atomic action detection and actor localization, we

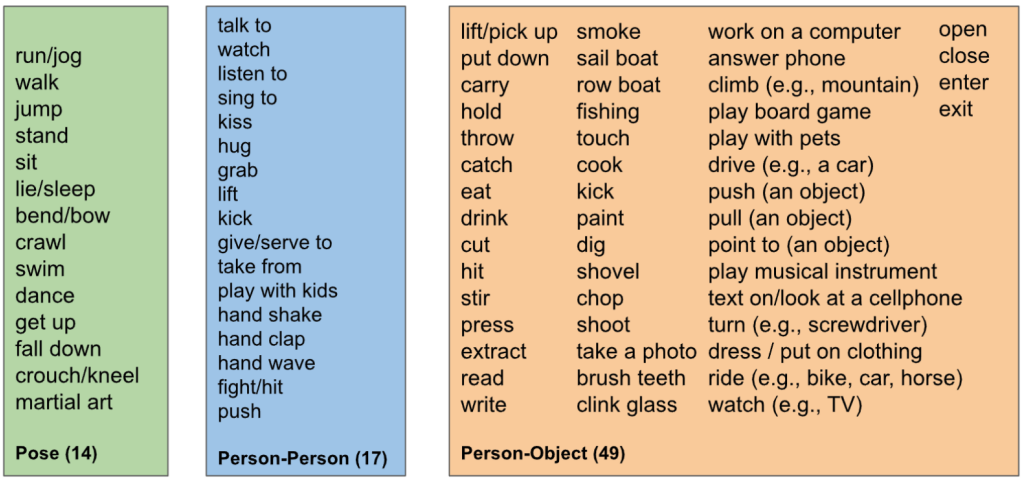


Figure 3.4: Different atomic action classes from the AVA [12] dataset. Actions are split into three main categories, 1) person poses, 2)person-person interactions and 3) person-object interactions. Figure is taken from AVA challenge slides [13].

also evaluate our model in this dataset.

3.3 Action Detection Task

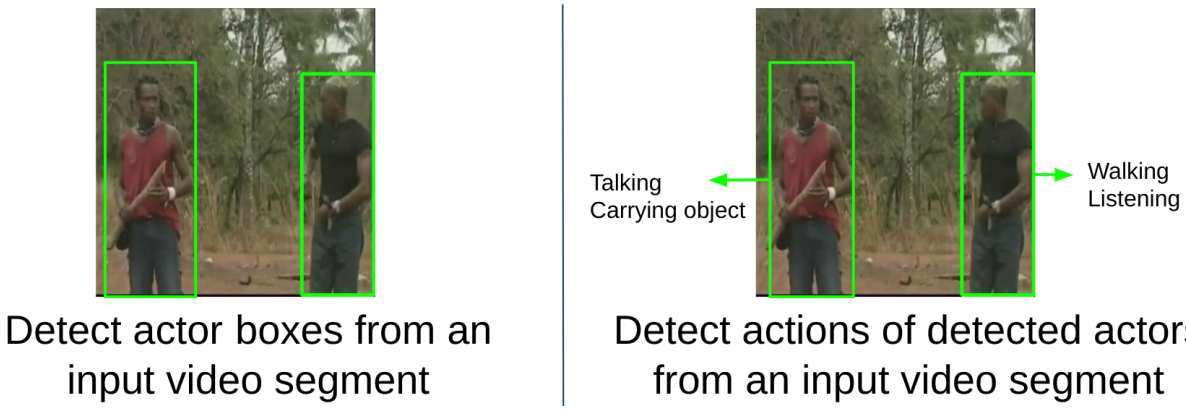


Figure 3.5: Action Detection objective. Left shows the actor localization task and right shows the action classification task on top of detected actors.

The main Action detection task can be split into two sub-tasks, actor localization and action classification. The goal of the localization task is to detect actor bounding

boxes from an input video and the goal of the classification task is to detect actions of these detected actors from their bounding box locations and the input video.

The actor detection task has clear parallels to the object detection and similar methods can be followed to achieve this objective. The most popular object detection architecture is the Faster-RCNN[34] networks. Faster-RCNN also splits the object detection task into object proposal and classifier stages. There have been recent works which follow similar architectures for action detection as well [12]. One major difference is that in our task the input is videos compared to images and the box locates the actor rather than the action. Our context-aware model handles these differences.

Faster-RCNN architectures use backbone 2-dimensional convolutional neural networks (CNNs) to extract features from the images and classify them. Since the input is videos, 3D CNNs are needed to extract spatio-temporal features. A popular 3D CNN for videos is Inflated 3D convolutions (I3D) model [46]. This model is shown to transfer well to other datasets and serve as a good backbone 3D CNN.

Combining Faster-RCNN with a I3D backbone, a baseline action detection model can be implemented. On I3D feature maps, RoIPooling can be used to extract features representing the actors. As this approach follows object detection, actor feature maps are cropped on bounding boxes. In object detection, the object bounding box contains the defining features of the object; but in action detection, the bounding box contains the actor. The action boundaries are not as well defined. Hence, this approach does not model action context. Following chapters address this issue for action detection and defines our context-aware action model.

3.4 Context-Aware Action Detection

While observing actions/activities, humans infer from the entire context and our perception depends on the surrounding objects, actors, and scene. This is a concept that has been widely studied in neuroscience and psychology [52, 53, 54, 55]. The idea of explicitly leveraging context is directly relevant to our action detection task as surroundings of actors provide valuable information.

Studies in action detection task have followed the ideas from the R-CNN architectures and extended it to videos[12, 56, 57, 58]. One major issue in this approach is that in action detection, the bounding box locates the *actor* rather than the *action* itself and datasets do not include explicit interaction labels for the actions, which makes it challenging to model context. In such setting, Region of Interest (RoI) Pooling becomes insufficient for modeling actions and including the contextual information such as actors, objects and scene. In order to address this, we propose attention maps as a replacement on RoIPooling for action detection. Our method learns context in a weakly supervised manner as demonstrated in Fig. 3.6 and improves on top of the RoIPooling approach.

Inspired by the context [48], attention [49] and relation [59] ideas, our model generates attention maps *conditioned on each actor from contextual and actor features*. Replacing the traditional way of cropping an actor RoI from the context feature maps, generated attention maps multiply and scale the context feature maps according to each actor. The scaling operation amplifies relevant regions and dampens the irrelevant regions to the conditioned actor. This allows the model to learn the complex interactions for the current actor with the scene which may include surrounding actors, objects and background.

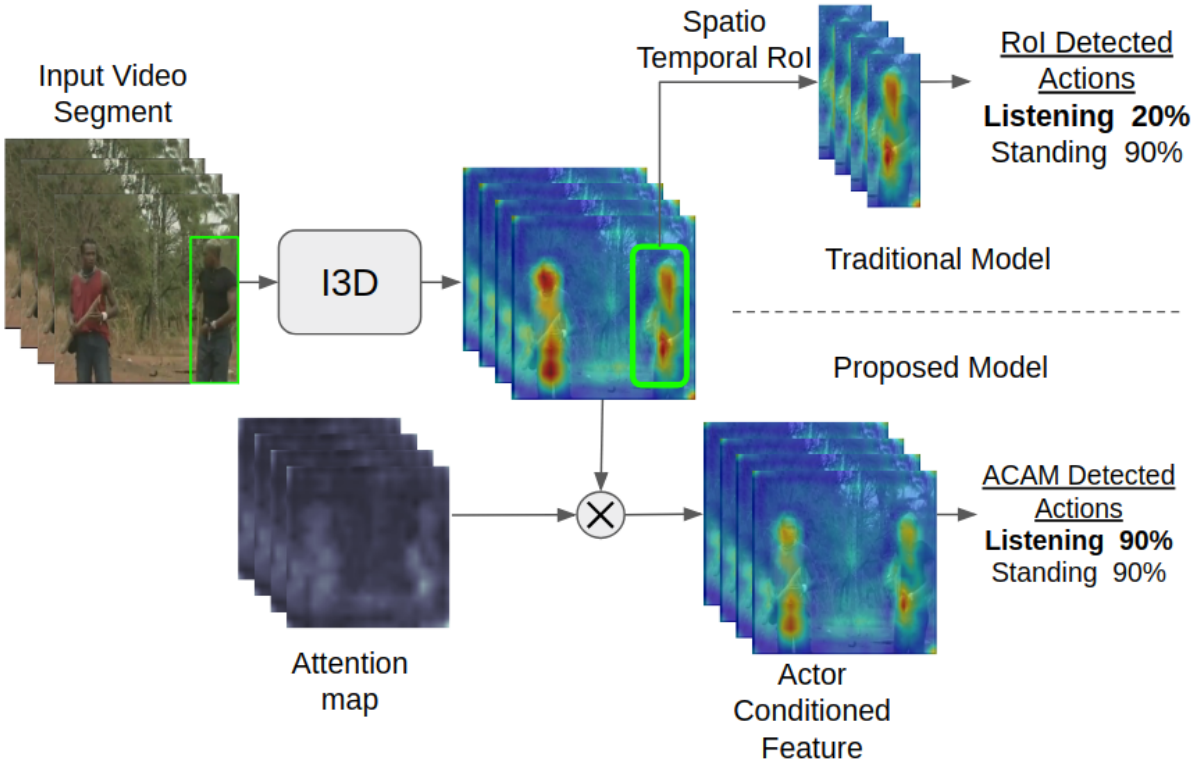


Figure 3.6: Comparing RoI pooling with our ACAM method for video action detection. ACAM explicitly models the surrounding context and generates features from the complete scene by conditioning them on detected actors. For example, presence of a talking person next to the actor is evidence for the "listening" action, which is captured by attention maps.

3.5 Related Work

State of the art models on the earlier action recognition datasets [60, 61, 6] use models such as Two-Stream networks [43] combining RGB with Optical Flow, 2D Convolutions with LSTMs [62] and 3D Convolutions [63]. The release of the large-scale, high quality datasets like Sports 1M [39], Kinetics [8], allowed deeper 3D CNN models such as C3D [44], Inception 3D (I3D) [46] to be trained and achieve high performance.

In this work, we are focusing on the Atomic Visual Actions (AVA v2.1) [12] dataset. This dataset exhaustively annotates the atomic actions and spatial locations of all the actors. Initial methods on the AVA dataset extended the Faster-RCNN [34] architectures

to 3D convolutions, where initial layers generate actor proposals and each proposal is analyzed by subsequent layers [12]. The recent Actor Centric Relation Network (ACRN) [59] model generates features by combining actor and scenes to represent actor's interactions with surrounding context. Our model leverages this relation idea to generate attention maps.

Attention models are used in Natural Language Processing (NLP) [64, 65, 49]. An attention function for relating different positions of a sentence was used in [49]. Studies in Visual Question Answering task focused on generating attention maps from the input question to focus visual model [66, 67, 68]. The relation module in [69] combines questions with vision to generate answers. [70] uses object relations to effectively detect them. This approach improves both instance recognition and duplicate removal. An LSTM structure is used in [71] to generate an attention map to model contextual information. Inspired by these attention models which use positional relations, our model generates a spatio-temporal attention map which scales different positions of the feature map depending on each actor.

Recently, [48] used a compact feature representation that compresses non-local information from contextual features with a weighted sum of pixels for action detection and achieved state-of-the-art results. This shows that contextual information is essential and detection can be improved by replacing the RoIPooling which ignores context. In a zero-shot learning setting [72] uses existence of objects and their locations to detect interactions.

Context has also been studied on image action detection. V-COCO [9] and HICO-Det [73] datasets have exhaustive annotations on persons, objects and their interactions. Unlike our task where the interactions are weakly supervised, these annotations enable models to learn interactions efficiently. Interaction modeling from [74] achieved good performance with a multi-stream network where each stream focused on people, objects

and interactions separately.

Our model builds on top of the relation idea from ACRN [59] where the relation between the actor and the surrounding context is generated. Unlike ACRN where the relation features are used for classification, our model leverages the relation function to generate attention maps. This is similar to [49] where attention maps localize relevant parts in a feature map. However, in our task, the attention maps condition the features to each actor in the scene individually. Since the feature maps are conditioned individually and include context, ACAM models actor-context information more effectively than RoIPooling.

3.6 Technical Approach

This section describes our model for action detection. From each input video segment, the objective is to detect bounding boxes for each actor and classify their actions. Each actor can have multiple action labels (e.g., "sitting" and "talking" simultaneously).

3.6.1 Context for Atomic Actions

Compared to object detection tasks, action boundaries are ill defined and can include interactions with the surrounding context (objects, actors and scene). Different actions require different sizes of visual areas to be considered from the input video. For example, the "walking" action requires the model to consider only the pixels on the actor and close surrounding context, whereas the "listening" action requires the model to look for in a larger context area (e.g., a talking person) around the actor in addition to the actor itself. With such variety in action classes, using traditional object detection methods such as RoIPooling can potentially lose the contextual information around the actors. Even though features cropped using RoIPooling include information from a larger receptive

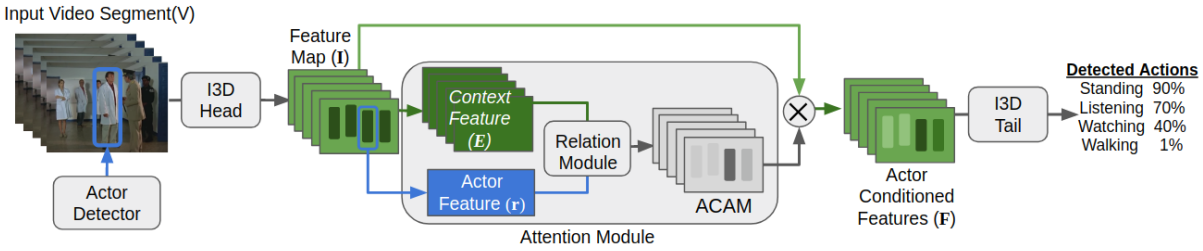


Figure 3.7: ACAM architecture. The input video segments are processed by the I3D back-bone. Feature vectors for each detected actor are generated from their locations on the feature map. A set of weights is generated for every spatio-temporal region in the scene by combining the actor features and contextual features extracted from the entire scene. These weights, i.e., attention maps, are multiplied by the feature map and the result represents the actor conditioned features. Four detected actors are represented by four vertical bars in I. One focused actor (boxed) is listening to a close-by actor. This action is captured by larger weights in the attention map shown as a darker vertical bar.

field, this technique compresses the information into a smaller feature map and does not explicitly model interactions (with other actors and context). Additionally, large-scale video datasets do not provide explicit interaction labels (person 1 is listening to person 2) but weak labels (person 1 - listening, person 2 - talking). These interactions need to be learned via weak supervision. In order to address these challenges, our method generates attention maps for each detected actor to model the importance of each spatio-temporal region in the feature map by conditioning on detected actors. Our model architecture is shown in Fig. 3.7.

3.6.2 Actor Conditioned Attention Maps

Our method generates a set of weights (ACAMs) that represent the attention of different parts of the spatio-temporal input. Our action detection problem contains multiple actors performing concurrent actions that can be either related or disparate. This generates an attention task, where different actors relate differently to spatio-temporal locations in the scene. The model addresses the attention problem by generating ACAMs,

which capture relations between actors and context. Instead of extending RoIPooling [34] to action detection as in [12], the essence of ACAMs is to condition the features extracted from the entire scene on each actor and action dynamics.

Let V represent the input video which is a sequence of frames (32 frames in our implementation). Spatio-temporal features are extracted from the input video V with a 3D convolutional back-bone (e.g., I3D [46]) up-to some layer (Mixed_4f). Let $\mathbf{I}$ represent the extracted feature tensor of size $(T \times H \times W \times C)$ with temporal dimension T and spatial resolution $H \times W$ indexed by t, h, and w and feature channel dimension C. i.e., $\mathbf{I} = conv3d(V)$.

The actor feature vector $\mathbf{r}_a$ of size N (set to $C/4$) is extracted for actor a using RoI pooling extended time via:

$$\mathbf{r}_a = \phi(\mathbf{w}_\rho RoI(\mathbf{I}, a) + \mathbf{b}_\rho), \tag{3.1}$$

where $\phi(x) = ReLU(x) = max(0, x)$, $\mathbf{w}_\rho$ are the weights, and $\mathbf{b}_\rho$ are the biases. Similar to Faster RCNN [34], $\mathbf{r}_a$ can be used for classifying the actions of actor a. Instead of using $\mathbf{r}_a$ directly, we propose to leverage its descriptive potential to generate relations between the actor and the context.

The conditioned feature vector $\mathbf{F}_{t,h,w|a}$ is computed for each actor a in the scene and spatio-temporal indices (t, h, w). This is generated by a conditioning function of actor feature $\mathbf{r}_a$ and contextual features $\mathbf{I}_{t,h,w}$ via:

$$\mathbf{F}_{t,h,w|a} = Condition(\mathbf{I}_{t,h,w}|\mathbf{r}_a,),\ \forall\ (t, h, w) \tag{3.2}$$

Following steps explain the $Condition$ function. Activations in $\mathbf{I}$ are sparse; however,

it is compressed by an additional layer to obtain a denser representation ($\mathbf{E}$) via:

$$\mathbf{E}_{t,h,w} = \phi(\mathbf{w}_\eta \mathbf{I}_{t,h,w} + \mathbf{b}_\eta), \tag{3.3}$$

where $\mathbf{w}_\eta$ and $\mathbf{b}_\eta$ are the weights and biases. The new tensor $\mathbf{E}$ has shape $(T \times H \times W \times M)$ with $M < C$ (set $M = C/4$). This approach reduces the dimensionality of $\mathbf{I}$ and captures higher level features similar to [48, 46, 75].

The relation tensor for actor a (i.e., $\mathbf{R}_a$) is inspired by the "relation" idea from [69] and it is modified to capture the relations between actor a and every location t, h, w in the context as:

$$\mathbf{R}_{a,t,h,w} = \mathbf{w}_\Omega \mathbf{r}_a + \mathbf{w}_\gamma \mathbf{E}_{t,h,w} + \mathbf{b}_\beta, \tag{3.4}$$

where $\mathbf{w}_\Omega$ and $\mathbf{w}_\gamma$ are the weights for actor and context features, respectively; and $\mathbf{b}_\beta$ are the biases. $\mathbf{R}_{a,t,h,w}$ describe the relation of actor features and contextual locations. We set up the shapes of $\mathbf{w}_\Omega, \mathbf{w}_\gamma, \mathbf{b}_\beta$ such that $\mathbf{R_a}$ has the same shape as $\mathbf{I}$. Note that ReLU (ϕ) is not used in Eq. 3.4.

Instead of using relation features for classification directly, we leverage the I3D backbone and its pre-trained weights by conditioning $\mathbf{I}$ on the actor a for an increased performance (Section 3.7.4). Inspired by the "forget" gates of LSTMs, attention module generates the actor conditioned attention maps for a (i.e., $\mathbf{ACAM}_a$) by:

$$\mathbf{ACAM}_{a,t,h,w} = \sigma(\mathbf{R}_{a,t,h,w}), \tag{3.5}$$

and conditioned features $\mathbf{F}$ as follows:

$$\mathbf{F}_{t,h,w|a} = \mathbf{I}_{t,h,w} \odot \mathbf{ACAM}_{a,t,h,w} \tag{3.6}$$

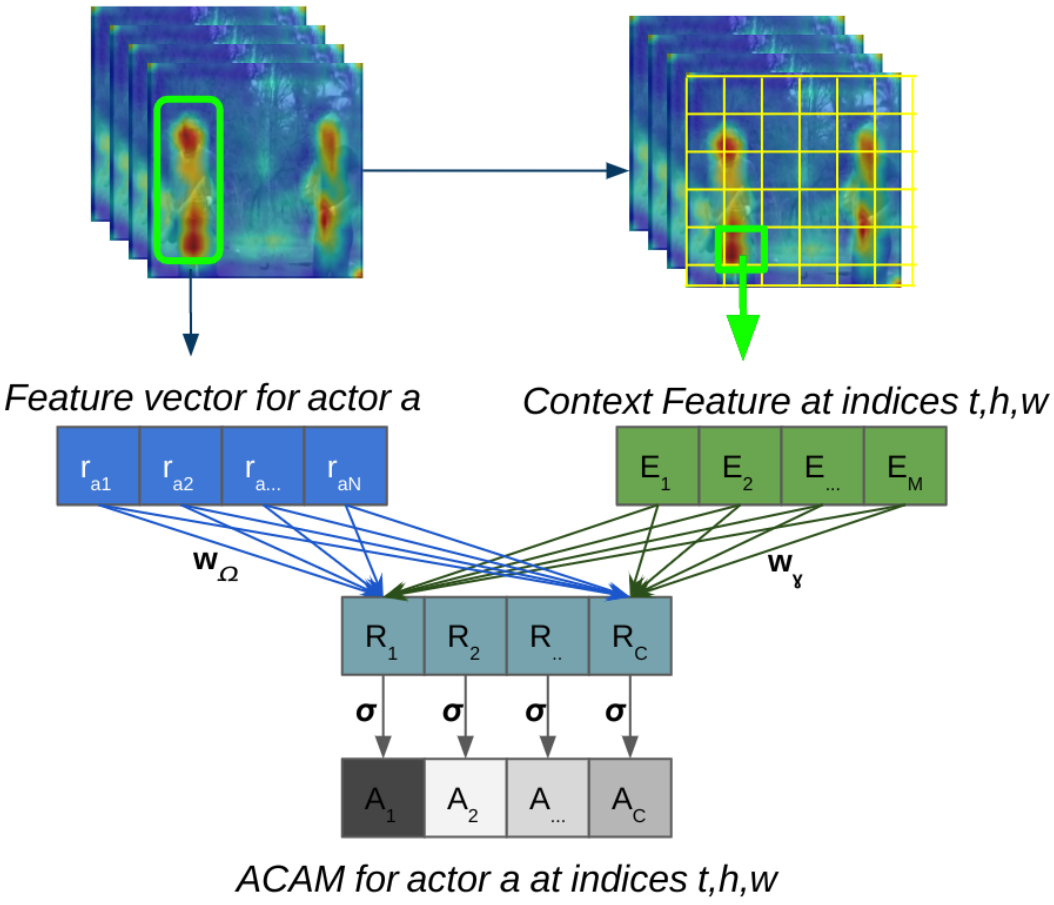


Figure 3.8: Attention module for actor a at a single index t, h, w. Attention weights in ACAM at index t, h, w are generated from the actor feature $\mathbf{r}_a$ and context features at the same index $\mathbf{E}_{t,h,w}$.

where σ is the sigmoid function which scales the attention maps in $[0, 1]$ interval and $\odot$ is the elementwise multiplication of vectors. Attention maps multiplied by $\mathbf{I}_{t,h,w}$ weights the different regions on the context. This process amplifies regions relevant to actor a, while damping the irrelevant regions. The generation of ACAMs is shown in Fig. 3.8 for actor a at a single spatio-temporal index t, h, w.

These operations are efficiently computed using $1 \times 1 \times 1$ convolutions. For instance, Eq. 3.3 are fully connected layers repeated for every t, h, w index, which is equivalent

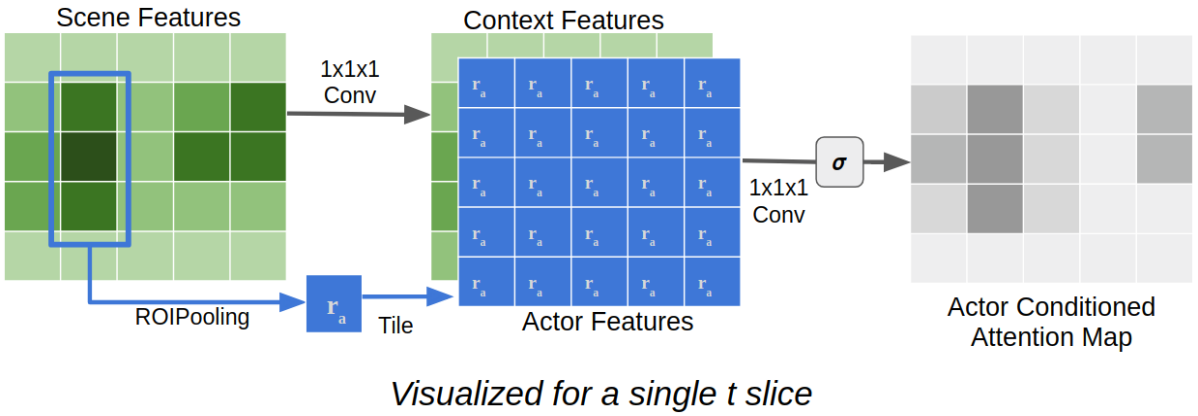


Figure 3.9: Calculation of attention maps with convolutions. Actor feature from the RoI is tiled and concatenated to features extracted from the context at every spatio-temporal index. Convolutions on the combined feature calculate the relations from Eq 3.4 efficiently.

to $1 \times 1 \times 1$ convolutions. In Eq. 3.4, $\mathbf{r}_a$ is constant for all indices t, h, w as shown in Fig. 3.9. This equation is computed by repeating $\mathbf{r}_a$ of shape $1 \times 1 \times 1 \times N$ to match the spatio-temporal shape of $\mathbf{E}$. The repeated actor feature has shape $T \times H \times W \times N$ and is concatenated with $\mathbf{E}$ to produce a tensor with shape $T \times H \times W \times (N + M)$. Applying the $1 \times 1 \times 1$ convolutions to the concatenated tensor is equivalent to Eq. 3.4 and produces $\mathbf{R}$. The sigmoid operation (Eq. 3.5) on $\mathbf{R}$ generates the attention maps (**ACAM**). Element-wise multiplication from Eq. 3.6 generates $\mathbf{F}$, which is then classified by remaining layers of the CNN back-bone.

3.6.3 Person Detectors as RPN

We experiment with pre-trained(COCO) frozen and fine-tuned(AVA) person detectors for actor localization. Our approach has the following three advantages over RPNs:

1. Transferability: Object detectors see large object variations (MS-COCO [3]). This allows models trained on object detection datasets to transfer to videos from different sources. Action datasets, however, usually come from similar sources such as

AVA (Movies), JHMDB (Youtube), which reduces the diversity in actor views and limits transferability of fine-tuned solutions for actor localization.

2. Efficiency: ACAM requires fewer actor proposals than RoI pooling to enable its complex computations. These detections are obtained from pre-trained person detectors.

3. Modularity: The action model is trained using a slow and highly accurate actor detector. The modularity of our method enables replacing detectors based on performance and application requirements.

3.7 Experiments and Evaluations

3.7.1 Datasets and Implementation Details

Datasets: The ACAM model is tested on the AVA v2.1[12] and JHMDB[40] datasets. In this section we summarize these datasets which were analyzed in more detail in Section ??.

AVA contains 2-second video segments of multiple actors with 211k training and 57k validation samples. Actor bounding boxes are annotated for the center frames only. Weak action labels are provided for the complete segment without temporal localization or explicit interactions. Actors can have multiple action labels in each segment. We follow the AVA v2.1 evaluation process and calculate the mean Average Precision (mAP) across 60 classes. There are three action super classes Person Poses (13 classes), Object Interactions (32 classes), Person Interactions (15 classes).

JHMDB contains 1-second video segments of 21 action classes across 928 video clips with single actor-action pairs.

3D CNNs: We use I3D [46] as the 3D CNN back-bone for all of our model candidates. The input video segment of RGB frames is processed by the initial I3D layers until the

"Mixed_4f" layer to obtain the feature tensor $\mathbf{I}$ of size $8 \times 25 \times 25 \times 832$. "Mixed_4f" is the name of a layer in I3D. The actor conditioned features $\mathbf{F}$ are computed using ACAM, where $\mathbf{F}$ is a weighted version of the original feature map ($\mathbf{I}$). The remaining I3D layers are used and initialized with pre-trained weights. We use the remaining layers up to final "Mixed_5c" for classification on $\mathbf{F}$ and call this operation "I3D Tail". A global average pooling across spatio-temporal dimensions is applied to the final feature map to compute class probabilities. Each 2-second video is uniformly subsampled down to 32 frames.

Actor Detection: Detectors process all the videos and store the detected actors locations. We use the Faster R-CNN [34] with NAS [76] detector pre-trained on MS-COCO [3] dataset. Additionally, we fine-tune the detector for actors and compare their performances on AVA and transferability on other datasets. This object detector is further analyzed and available in Tensorflow Object Detection API [19].

Data Augmentations: In addition to cropping and flipping the video sequences, we augment the actor box coordinates from the detector. This generates a slight difference in extracted $\mathbf{r}_a$ at each training step and reduces overfitting.

Training: We initialize our models with I3D weights trained on Kinetics-400 dataset [46] and train our models with Adam optimizer [77] and cosine learning rate [78] between max (0.02) and min (0.0001) for 70 epochs. We use a batch size of 2 per GPU and 4 Nvidia 1080Ti (total batch size of 8). Batch-norm updates are disabled. All models are implemented on Tensorflow [79].

3.7.2 mean Average Precision (mAP) with IoU

For the benchmark performance metric, AVA dataset uses the mean Average Precision(mAP). Average precision estimates the area under the precision-recall curve by sorting the predictions and sampling points on the curve. Following steps are used in

Model Architecture	AVA v2.1 Validation mAP
Single Frame[12]	14.20
I3D [12]	15.10
ACRN [59]	17.40
ACAM	23.29
ACAM - tuned	**24.38**

Table 3.1: Validation mAP results compared to published state of the art results. The ACAM model with fine-tuned actor detector achieves the highest performance on the AVA v2.1 Validation set.

calculating the AP value:

1. Sort predictions by their confidences.

2. For every prediction, calculate precision-recall values for that confidence threshold.

3. Linearly sample K points (101 usually) of recall and get their precision values.

4. Using these points, estimate the area.

Using these steps, AP is calculated for each one of the 60 classes and then averaged to obtain the mean AP score.

Additionally, since the task is also includes detecting the actor bounding boxes intersection over union(IoU) values for prediction and ground truth boxes are calculated. A prediction is true positive if there is a ground truth box which has greater than 0.5 IoU and with the same action class as the prediction.

3.7.3 Comparisons with the State of the Art

Table 3.1 shows ACAM with fine-tuned object detector outperforming the recent ACRN [59] on AVA validation set by 7mAP. We compare our model with validation results of the models from "ActivityNet 2018 AVA challenge"[80]. Table 3.2 shows that

Model Architecture	AVA v2.1 Validation mAP
YH Technologies[81]	19.40
Megvii/Tsinghua[82]	20.01
Deep Mind[83]	21.90
ACAM	23.29
ACAM - tuned	**24.38**

Table 3.2: ACAM mAP results compared with models from the ActivityNet CVPR-2018 AVA challenge. We excluded the ensemble/fusion methods to evaluate the benefits of the proposed layer.

ACAM outperforms in validation. The table excludes results from ensemble models and focuses on comparing their highest performing single RGB model.

Additionally, we compare the effects of fine-tuning the actor detector on AVA dataset. Comparing ACAM with **ACAM-tuned** demonstrates the improvement gained by fine-tuning the actor detector on the AVA dataset. However, this comes with trade-offs as analyzed in Section 3.7.6.

3.7.4 Comparisons of Individual Modules

In this section, we demonstrate the purpose of each module and experiment with alternative models to ACAM for representing contextual interactions. These experiments use actor detectors that are pre-trained, frozen on COCO dataset and are not fine-tuned for AVA. AVA mAP results, inference speeds and number of parameters for these implementations are shown in Table 3.3 to analyze the trade-offs.

I3D Head + RoIPool (Base Model): The base in-house implementation follows the model from [12] and achieves 18.01 mAP. The input video goes through I3D convolutions upto the layer "Mixed_4f" and call it I3D Head. Using RoI pooling the actor feature vector $\mathbf{r}_a$ is obtained and used with fully connected layers for classification.

I3D Head + RoIPool + I3D Tail: This model uses RoI pooling to extract actor

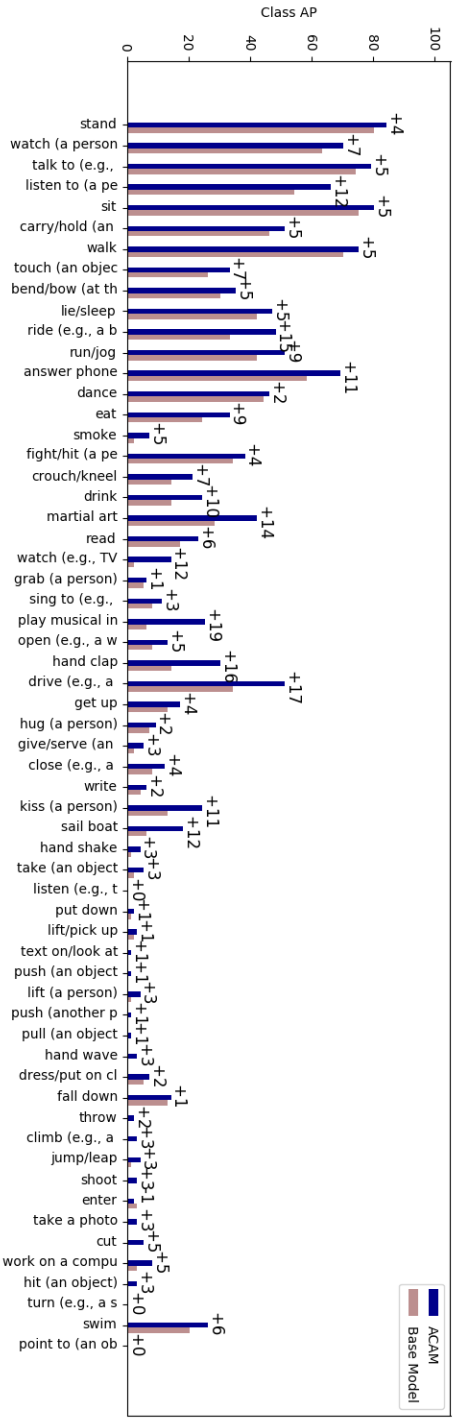


Figure 3.10: Per class AP results for the ACAM model and the base model I3D Head + RoIPool on the AVA dataset. The classes are sorted by the number of training samples available in the dataset. Improvements achieved by ACAM are shown on the bars.

Model Architecture	AVA mAP	Test Speed	# Params
I3D Head + RoIPool (Base)	18.01	8.3 samples/s	7,573,244
I3D Head + RoIPool + Tail	19.83	8.1 samples/s	12,341,484
I3D Head + ACRN + Tail	20.59	7.1 samples/s	13,034,956
I3D Head + NL-RoI + Tail	20.82	7.5 samples/s	13,035,164
I3D Head + ACAM + Tail	**23.29**	6.9 samples/s	13,034,956

Table 3.3: mAP results of our different variants. We compare ACAM with alternate attention modules and base models. Inference speed is measured on a single 1080Ti GPU. Number of parameters include I3D Head and Tail parameters for applicable models.

Model Architecture	Pose	Objects	Interaction
I3D Head + RoIPool	36.88	9.87	19.02
I3D Head + RoIPool + Tail	38.45	12.11	20.16
I3D Head + ACRN + Tail	38.38	12.52	22.37
I3D Head + NL-RoI + Tail	40.50	12.06	22.45
I3D Head + ACAM + Tail	**42.13**	**15.02**	**24.22**

Table 3.4: mAP comparisons of different RoI variants on AVA action super classes. Pose: Person Pose actions (E.g., walking, standing), Objects: Object Manipulation (E.g.: drink, pull), Interaction: Person Interaction (E.g., talk to a person, watch a person).

features. Instead of vectorizing the RoI and using fully connected layers, we use the remaining I3D layers from "Mixed_4f" to "Mixed_5c" similar to the model from [83]. This method achieves 19.83 mAP and demonstrates that using I3D Tail improves the performance.

I3D Head + NL-RoI + I3D Tail: Non-Local Neural Networks [48] uses a compact representation to model interactions between different spatio-temporal regions in a video segment. We modify this model to generate non-local features between the detected actor features $\mathbf{r}_a$ and scene context features $\mathbf{I}$. This model achieves 20.82 mAP.

I3D Head + ACRN + I3D Tail: Similar to the ACRN [59], this implementation classifies on the relation features ($\mathbf{R}$). To improve performance, we use "I3D Tail" instead

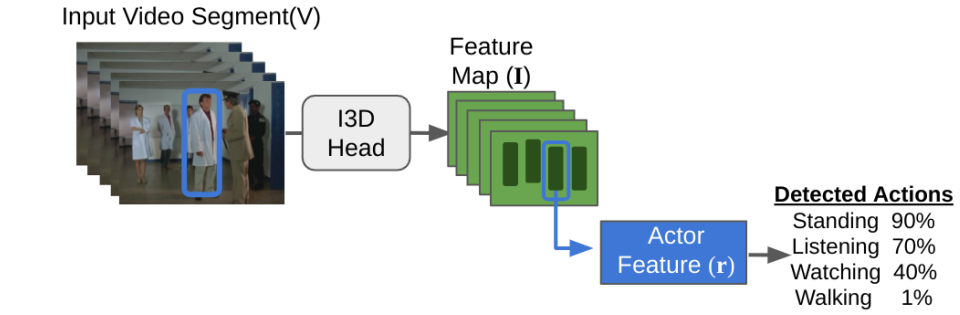
of the added 3×3 convolutions. This model achieves 20.59 mAP. Comparing ACAM to this model demonstrates that attention based context is better than relation features alone.

I3D Head + ACAM + I3D Tail: This model uses the ACAMs to condition context features on actors and classifies the actions using I3D Tail. This model achieves 23.29 mAP, which is the highest performance when compared to the alternative relation models.

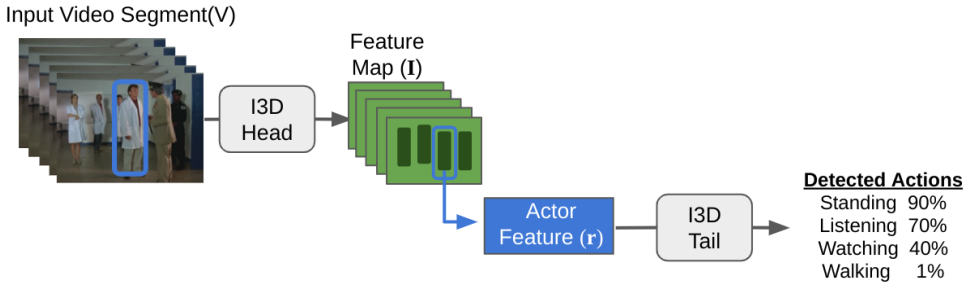
The breakdown of performance per action super class is demonstrated in Table 3.4. the AP values in the table are averaged across super classes. This experiment demonstrates leveraging contextual information with ACAM improves the performance for every super class.

The per class performance (AP) comparison on the ACAM model and the base model is shown in Fig. 3.10. We observe significant (above 10 AP) improvements in passive actions such as "listen a person" and "watch TV" as in those classes context is active. Scene context improves the detection of classes such as "drive" and "play instrument".

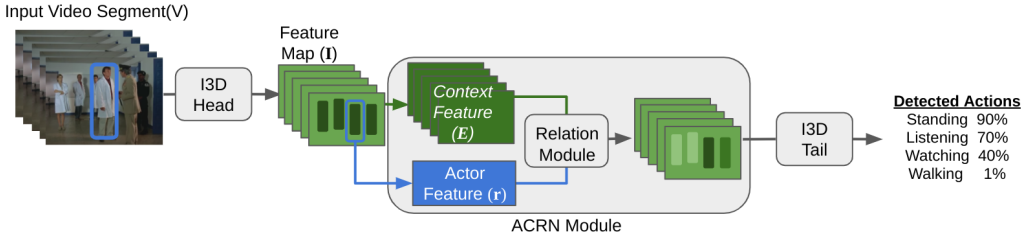
Figure 3.11 shows the model architectures of the alternative models.



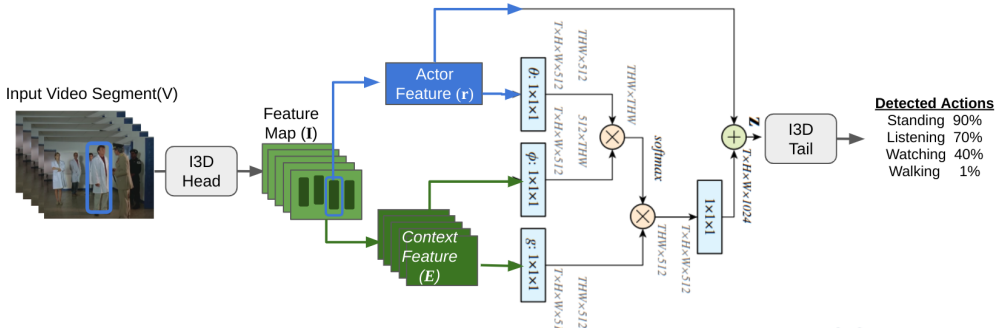
a) I3D Head + RoIPool (Base Model)



b) I3D Head + RoIPool + I3D Tail



c) I3D Head + ACAM + I3D Tail



d) I3D Head + NL-RoI + I3D Tail

Figure 3.11: Alternative actor RoI models and contextual attention architectures. These models are individually implemented and compared to our ACAM model.

3.7.5 Results on JHMDB

In addition to the AVA dataset, we evaluate our models on the JHMDB [40] dataset. We follow the evaluation protocol and cross-validate and report the video and frame mAP results on three splits. We edit and use the evaluation script from [56]. Our models are fine-tuned from Kinetics-400 weights and are trained for 10 epochs. Table 3.5 shows the video mAP scores of three of our models on JHMDB and demonstrates that the ACAM model achieves the best performance. ACAM model achieves the best performance across all implementations which is consistent with the AVA dataset results.

JHMDB - Models	Split1	Split2	Split3	AVG
I3D Head + RoIPool	77.57	73.91	75.64	75.71
I3D Head + RoIPool + Tail	80.53	81.44	80.77	80.91
I3D Head + ACAM + Tail	**84.68**	**83.78**	**83.30**	**83.92**

Table 3.5: Video mAP results on 3 splits of JHMDB and the average.

Table 3.6 compares video and frame mAP of our ACAM model with state-of-the-art models. ACAM outperforms the by 3.80 video mAP and 1 frame mAP without optical flow.

JHMDB - Models	Frame mAP	Video mAP
Action-RCNN[56]	58.5	73.1
ACT-Tubelet[84]	65.7	73.7
I3D-RoI[12]	73.3	78.6
ACRN[59]	77.9	80.1
ACAM	**78.9**	**83.92**

Table 3.6: mAP values averaged across 3 splits of JHMDB dataset.

3.7.6 Ablation Analysis

Visualization of Attention Maps: In ACAM, an attention map for each feature channel is generated. This allows us to model different types of interactions efficiently. Since the feature maps are sparse, to visualize them, we average the attention map values across the feature dimension where they have non-zero values in their respective feature map. This generates a representation where each actor's relation with the scene is visible. Fig. 3.12 shows this visualization on different examples. Note that a higher attention value is obtained on objects and actor faces/hands.

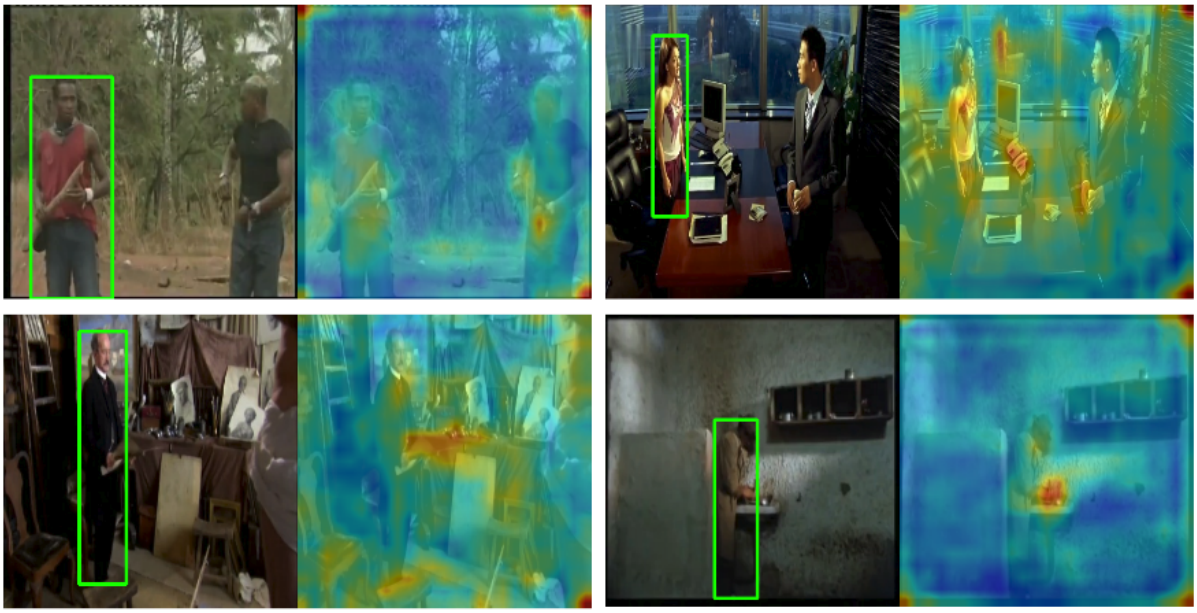


Figure 3.12: Generated Actor Conditioned Attention Maps. Higher attention values are usually observed around objects (paper, chairs, teapot, phones), on faces and hands of the actors.

Class Activation Maps: Using the global pooling layer at the last layer, we can generate class activation maps for each class (similar to [85]). As the final layer learns to associate the averaged features, these weights can be used before the averaging operation to visualize where each class is getting activated.

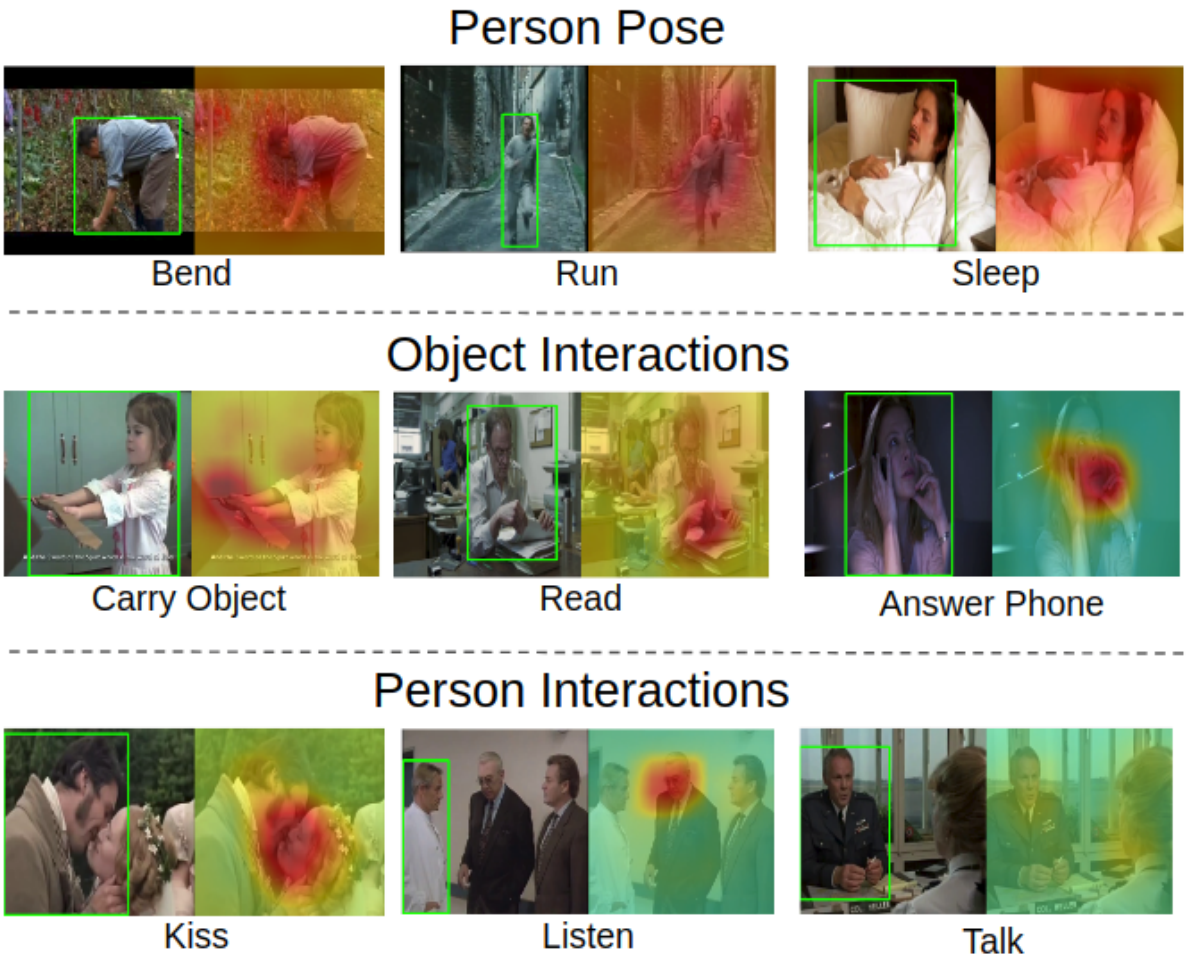


Figure 3.13: Class activation maps for detected actors in validation set. Each image represents the activation maps for the actor annotated by the green box and the given class. Red regions on the activation maps represent larger values.

We demonstrate activation maps for several different cases. Fig. 3.13 shows activation maps for different categories of actions. Activation maps are shown for actors annotated in green bounding boxes. Maximum activations across timesteps are visualized in the figures as these activations are also time sequences.

We observe that pose actions such as run/bend get activated around the actor while object interaction actions such as carry object/read are activated around the relevant objects and the actors. Person interactions also show some interesting results. The passive actions such as “watch a person”, “listen to a person” gets activated where there

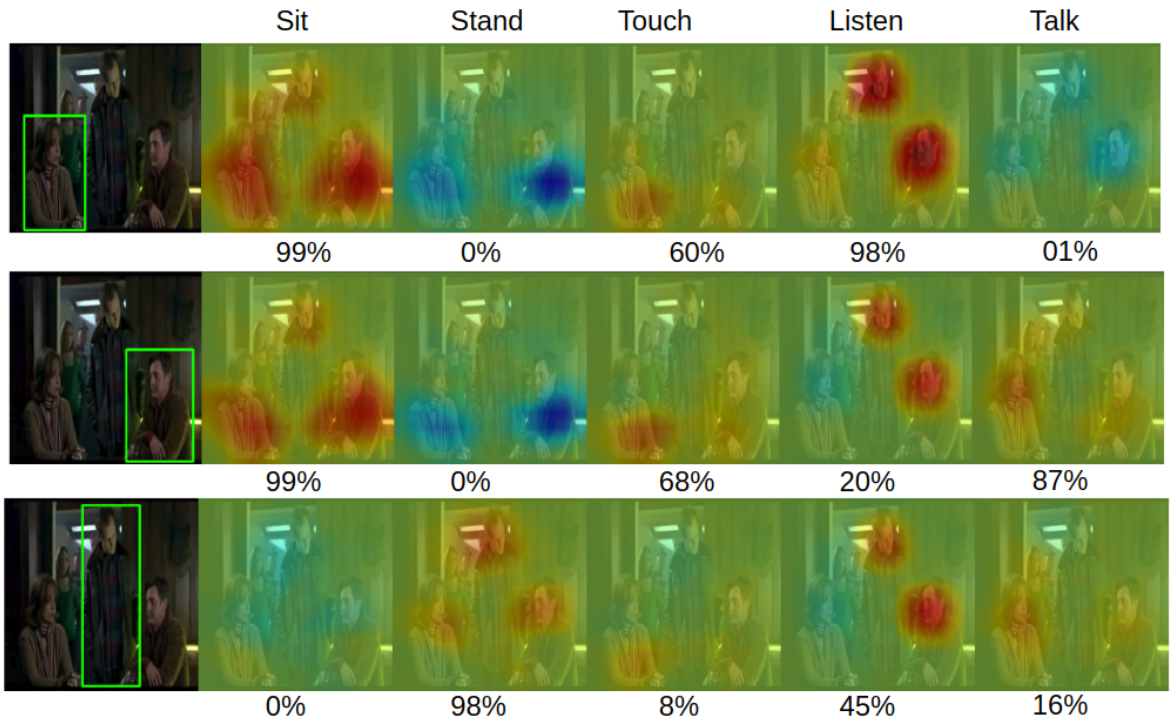


Figure 3.14: Class activation maps for detected actors. Each row represents the activation maps for the actor annotated by the green bounding box. Person on the right is talking in the video. Red regions on the activation maps represent the higher values.

is another person in the scene that is "talking" or relevant.

Fig. 3.14 shows a scene with three people and their conditioned activation maps for specific actions. Each row represents the activation maps that are conditioned on the person in the green bounding box. We observe that complementary actions such as "talking" and "listening" gets activated on the person with the opposite action. This is due to our model architecture. As we initially extract the actor feature vector $\mathbf{r}_a$ from the actor's location, this feature vector contains the information that the current actor a is "listening". Therefore the attention map generated from actor's vector $\mathbf{r}_a$ and context $\mathbf{E}$ looks for a person that is "talking" and focuses the attention on those locations.

Actor Detection Performance: To test the performance of the actor detector, we

calculate the detection AP of every actor for every class on validation set. Table 3.7 shows the detection frame AP scores for the AVA v2.1 validation set.

Object Detector	AVA Actor Detection AP	Speed(ms/frame)
F RCNN-NAS	97.10	1833
F RCNN-Resnet101	95.97	106
SSD - MobileNetV2	66.16	31

Table 3.7: AP results for actor detection rate for different detectors and their detection speed. This demonstrates that detectors work well without fine tuning and shows the speed trade-off.

Transferable Actor Detection: The main reason of using a pre-trained frozen person detector instead of training an RPN on the action dataset is transferability. During training, object detectors see large variations in objects from large datasets such as MS-COCO [3] compared to action datasets. This makes object detectors more transferable to other datasets compared to retrained RPNs. To test this hypothesis, we compare the actor detection rates of same model architecture with and without fine-tuning. AVA model is fine-tuned on top of the COCO weights whereas COCO model is frozen and is not fine-tuned.

Actor Detection	F RCNN AVA	F RCNN COCO	Δ
AVA	98.60	95.97	+2.63
VIRAT	9.94	30.44	−20.50
KITTI	27.04	54.57	−27.53

Table 3.8: AP results for actor detection rate of the same object detector trained on AVA and COCO and tested on different datasets. Actor detectors lose transferability to different domains when fine-tuned on action datasets (AVA in this case), which is shown by the difference (Δ: F RCNN AVA - F RCNN COCO).

Table 3.8 shows their comparisons on different datasets. Even though fine-tuned model achieve slightly better actor detection rate on the AVA dataset, the performance degradation is significant on datasets such as VIRAT [10] and KITTI [4].

3.7.7 Comparisons of Attention Mechanisms

We compare ACAM with similar attention studies including: Actor-Centric Relation Network (ACRN) [59], Attentive Contexts for Object Detection (ACOD) [71] and Non-Local Neural Networks (NL) [48].

ACRN uses the Relation module (Eq. 3.4) without the sigmoid function to generate relation features for actors and context. The relation features are directly used for action classification with convolutional layers which are trained from scratch. In contrast, ACAM leverages the relation features to generate attention maps, which amplify/dampen the I3D feature map $\mathbf{I}$. The actor conditioned features $\mathbf{F}$ are scaled versions of $\mathbf{I}$ and the model is able to effectively leverage the pre-trained I3D-Tail on $\mathbf{F}$.

ACOD uses an LSTM branch to generate an attention map. It leverages context to generate a single global feature for every region proposal. This "attention branch" omits the individual differences of region proposals. ACAM generates attention maps by conditioning the context on each detected region (actor) individually to represent individual interactions which is better suited for action detection tasks.

NL, similar to the self-attention [49], uses matrix multiplication to find relations among pixel pairs in a spatio-temporal feature tensor. The relations are normalized through a softmax function to emulate an attention map. In an action detection setting where the actions are focused on actors, instead of finding relations among all pixel pairs, it is more effective to find relations between all context pixels and condition on individual actor features as in ACAM.

3.8 Conclusions

We presented a novel action detection model which explicitly captures the contextual information of actor surroundings. Compared to traditional methods which use Region

of Interest pooling to extract actor features, our method finds actor's relationship with the whole spatio-temporal context. Context is modeled by using attention maps as a set of weights to highlight the spatio-temporal regions that are relevant to the actor, while damping irrelevant ones. ACAM is more suited for preserving interactions with surrounding context such as objects, other actors and scene. We demonstrated through thorough experimentation that ACAM improves the performance on multiple datasets and outperforms the state-of-the-art.

Our action detection model is one of the first ones in the community. In the following chapters, we will be building on top of this model and demonstrate that the action detection can transfer to other datasets and settings without fine-tuning.

Chapter 4

Real-time Action Detection

4.1 Introduction

Detecting actions near real-time has wide variety of use cases. Such frameworks can allow your smart home environments to recognize actions and act accordingly, allow smart robotics (such as autonomous cars, robot assistants) to understand and follow commands, automated surveillance systems to analyze their environment and conduct threat assessment efficiently. Imagine a police officer guiding a self-driving car through a blocked road or a selfie drone which is guided by basic gestures from the user. Action/gesture recognition framework on humans will make such frameworks possible in the future and change how humans interact with machines as gestures can act as an interface. In this chapter we will discuss the challenges of building a near real-time framework for action detection and demonstrate our solution to this task. Our real-time solution is open sourced and is available on GitHub[1]. Initial results of this research are published in [51].

Understanding human actions in a video stream depends directly on the frameworks

[1]Real-time action detection repository: https://github.com/oulutan/ACAM_Demo

ability to detect objects and humans, understanding human dynamics and analyzing the scene. To analyze a human's action, first we need to be able to detect and track the human. Previous works in action detection task have followed the ideas from the R-CNN[34] architectures and extended it to videos[12, 56, 57, 58] in order to get an "end-to-end solution". Similar to object detection tasks, these models have "region proposal networks" which generate bounding box proposals for actor locations. These region proposal networks are basically human detection networks trained on action detection datasets. This approach has the following of disadvantages:

- Human detection models trained on action detection datasets lose their transferability. Action datasets include wide variety of views for actions but they lack in variations of actors. This causes the region proposal network to overfit on the actors from the action dataset and prevents it from generalizing to inputs such as drone-views, surveillance cameras, webcams etc.

- This approach is not modular. Region proposal networks are intertwined with the action detection networks and can not be replaced during test time. These region proposal networks are usually computation heavy as their goal is to boost performance on the action dataset. However, during test time, more suitable networks might be available for actor detection in specific applications such as a faster network or a network that is trained on specific views (ex. surveillance cameras have different views compared to movies).

- In action detection the bounding box locates the actor rather than the action itself. Real action boundaries are not well-defined as they can include objects or humans of interest for the current actor. Combining the training process for action detection with actor detection adds another layer of difficulty on an already challenging task.

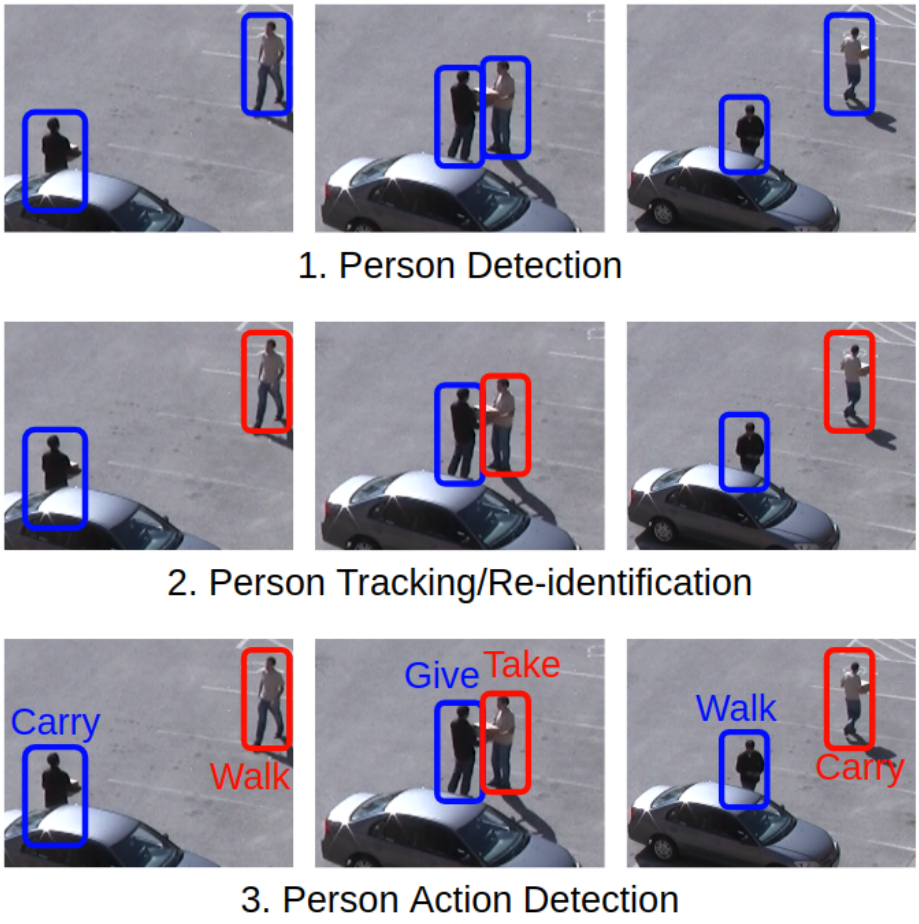


Figure 4.1: We combine 1) object detection models trained on object detection datasets, 2) trackers trained on human tracking problems and 3) action detection models trained on action detection datasets. Images are taken from VIRAT dataset [10]

Following these reasonings, we implement a modular approach where each of the modules are trained specifically on their own task. Our framework for action detection combines modules for 1) object detection models trained on object detection datasets, 2) trackers trained on human tracking problems, and 3) action detection models trained on action detection datasets. Figure 4.1 shows the breakdown of modules for our action detection framework and shows the individual tasks for each module.

4.2 Problem Statement

Given an input video segment, the goal is to detect and analyze the actions of every person in the scene. Following the atomic actions idea from the previous chapter, we aim to analyze shorter video segments rather than large chunks. This allows the use of atomic action detectors which can be used as a basis to form more complex actions.

Our framework analyzes 32-frame segments which represents a 2 second video (16 fps input) with a 75% overlap. Given a 32-frame input segment with frames $F_1, ..., F_{32}$, initially each frame is individually analyzed and a set of objects O_i for each frame F_i is obtained.

$$O_i = Object\_Detector(F_i) \tag{4.1}$$

Given the objects at every frame, the tracker/re-identifier module will use the visual similarity and geometric movements of boxes to match people across frames. The tracker has a state which keeps track of the previous box locations and guesses a box location for the current frame (e.g.: A Kalman Filter).

$$T_{i-1,i} = Tracker_i(F_i, O_i; F_{i-1}, O_{i-1}) \tag{4.2}$$

Combining the tracks over the 32 frames, we obtain the final tracks for each detected person. Tracks are filtered out if they are shorter than 24 frames. Remaining P person tracks, i.e. $Tube_p$ (bounding boxes over time for person p), and input frames are sent to the action detector to detect atomic actions individually.

$$A_p = Action\_Detector(Tube_p, (F_1, ..., F_{32})) \tag{4.3}$$

Where $p < P$ indicates the person id. In this implementation, action detector sees

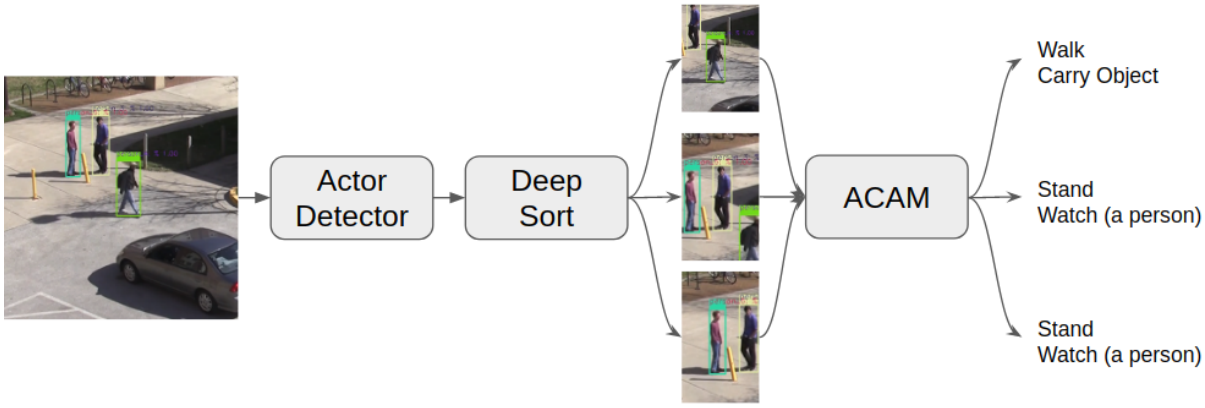


Figure 4.2: ACAM action detection framework running on a surveillance video from VIRAT [10]. Actors are detected by object detectors and tracked over frames by Deep Sort [14]. The generated tubes for each person is analyzed by the ACAM action detector.

the whole input frames rather than just cropping the persons' bounding boxes over time as a track. That allows the action detector to use the context which is essential in person-object and person-person interaction classes.

4.3 Framework Details

In this section, we individually define tasks for each module, provide implementation details for these modules and evaluate their performance quantitatively and qualitatively. Figure 4.2 shows the overall framework and modules used for this task.

4.3.1 Actor Detection

Significant progress have been achieved in recent years on object detection task using the convolutional neural networks. With the release of large scale object detection datasets such as common-view MS-COCO[3] dataset, drone-view VisDrone[5] dataset, autonomous driving vehicle KITTI[4] dataset, object detection models achieved high

performance in various view-angles and settings.

Modern object detectors have achieved high detection performance and fast speeds using wide range of configurations. Some configurations are even able to run on mobile devices with near-real time inference speeds. There are two main design choices that affect the detection performance and inference speed trade-off, convolutional neural network feature (CNN) extractor and detection architecture.

Modern object detectors use/fine-tune CNNs trained on image recognition datasets such as ImageNet [2] as a backbone feature extractor. These CNNs vary significantly between number of layers, size of weights which affects performance and inference speed directly. Table 4.1 shows accuracy results and inferences speeds of common CNNs on the ImageNet dataset. These models are usually used in object detectors as feature extractors. This table shows the trade-offs between these CNNs. A detailed analysis of these models can be found in [18].

CNN Architecture	ImageNet Accuracy	Titan XP Speed	Jetson TX2 Speed
MobileNetV2 [37]	71.81	3.34	20.51
Resnet101 [20]	77.31	8.90	84.52
InceptionV3 [86]	77.50	10.10	79.39
NASNET [76]	82.50	32.30	437.20

Table 4.1: Imagenet accuracy and speed trade-offs for common convolutional neural networks used as backbones for object detection. Speed is measured in ms/image. Results are taken from [18]

Detection architecture is another factor that affects the granularity of the detection model. This also affects the performance-speed trade-off. Most modern detection architecture essentially follow either the Faster-RCNN [34] or Single Shot Detector (SSD) [36]. The main difference between two models is that Faster RCNN models use a 2-stage model with region proposal networks and classifiers while SSD models directly classify on the region proposal stage. This allows allows SSD architectures to run faster while

trading-off some detection performance.

Faster-RCNN: Faster RCNN [34] model builds on top of the RCNN[30] architecture. RCNN architecture is the first significant achievement for using neural network methods for object detection by extracting fixed length convolutional features and uses an SVM to analyze the object classes. Fast RCNN[35] and then Faster RCNN[34] removes the SVM part, introduces Region of Interest pooling on CNN features and combines region extraction with classification in a single end-to-end network.

In Faster RCNN models, the detection happens in 2 main stages, region proposal network (RPN) and box classifier stages. The first stage, the region proposal network (RPN) generates boxes which are likely to have an object in them, i.e. box proposals. These box proposals are generated from "Anchors". Each anchor gets scored on their objectness using the feature map. After a non-maximum suppression which filters out redundant boxes, remaining high objectness score anchors are sent to box classifier stage as region proposals. The second stage, the box classifier stage takes the high scoring box proposals and crops their feature from feature map using Region of Interest (RoI) pooling. RoI pooling takes a box coordinate and extracts a fixed size feature vector from the convolutional feature map defined by this box coordinate. This allows different shapes box proposals to generate a fixed size feature vector which is then further analyzed by the classifier. Box classifier takes the RoI pooled feature vector and generates classification scores for each of the available object classes.

Single Shot Detector(SSD): SSD [36] is a simpler method compared to Faster RCNN. SSD architecture directly uses the RPN stage of Faster RCNN to classify anchors and does not use a specialized second stage for per proposal classification. For each anchor, SSD directly runs the multi-class classification and regression. This allows the model to run quickly in a single run.

As the anchors are directly classified in SSD compared to a 2-stage Faster RCNN,

SSD does fewer computations to analyze each given image, uses less memory and runs faster. SSD modules with MobileNet [37] can be used for mobile applications and run on cell phones for inference [37].

Table 4.2 shows some results of SSD and Faster RCNN architectures with various, commonly used CNN backbones. This shows the speed-performance trade-off for these architectures and backbone CNNs. FRCNN-Nasnet achieves about twice the mAP on COCO compared to SSD-MobileNetV2 while running about 60 times slower.

CNN Architecture	MS-COCO mAP	Titan X Speed(ms)
SSD-MobileNetV2	22	31
SSD-InceptionV2	24	42
FRCNN-InceptionV2	28	58
FRCNN-Resnet101	32	106
FRCNN-NASNET	43	1833

Table 4.2: MS-COCO detection rates and speed trade-offs for various detection architectures and backbone CNNs. Results are taken from [19]

Since our framework is implemented with modularity as goal, our model can use any of the object detectors defined in the Tensorflow Object Detection [19] framework. This allows the object detector to be replaced depending on the input and speed/performance needs.

4.3.2 Trackers - Deepsort

In our implementation, object detectors are run on every frame and analyze frame individually. Next required step is to link these detection across time. In order to achieve this, we use publicly available, real-time and accurate DeepSort [14].

Deepsort combines the traditional tracking algorithm Kalman filter with deep neural network matching models. This approach combines the appearance information of the matching neural network with the motion information of the Kalman filter.

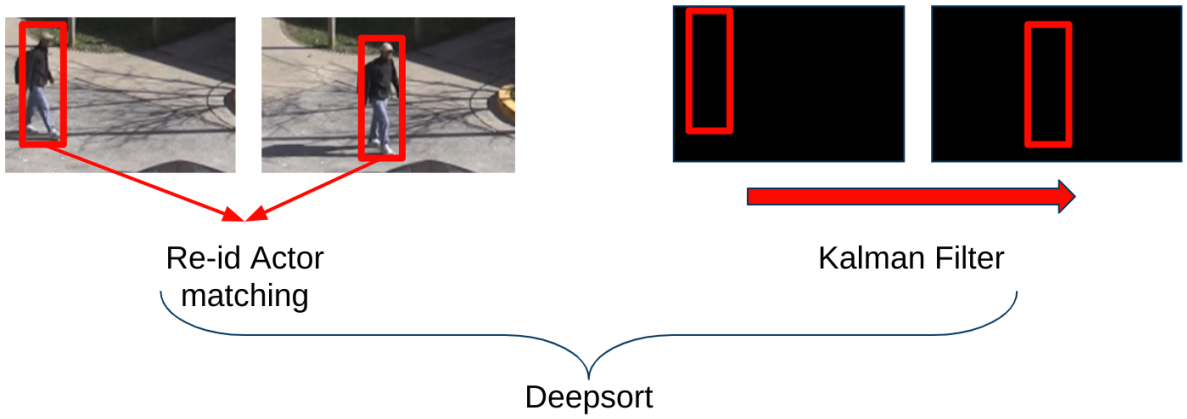


Figure 4.3: Deepsort[14] tracking algorithm visualized. This algorithm combines the traditional Kalman filters on bounding box locations and sizes with an appearance matching model which uses a re-identification CNN[15] to match detections with tracks.

Given a track of bounding box locations and sizes over time, Kalman filter predicts a new state, which estimates the bounding box location and size in the current frame. The estimated bounding box location and sizes are compared to every detection box in the current frame and the distances are calculated. These distances are thresholded with a pre-defined value and most of the unlikely detections for this specific track are filtered out. However, this approach does not consider the appearance of the detection and tracks over time and can easily miss out cases like camera movements and rapid displacements. In order to address these issues an appearance model is combined with the Kalman filter.

Given a track, a set of features are extracted from bounding box locations in every frame and collected over time. The features are extracted using the bounding box coordinates and a CNN trained for person re-identification [15]. This re-id CNN establishes the association process by genering features which can be used to match similar appaerances. For each detection in the current frame, distances are calculated with track features and the minimum distance track frame is used for the matching algorithm. This minimum distance of track and detection features are thresholded to eliminate unlikely matches.

These two distance metrics, appearance and box location distances, are combined to make a hybrid decision. Figure 4.3 shows the hybrid approach. In our framework, this combined approach associates detection boxes over frames and generates tracks for each actor. The next step is to analyze actions on these tracks.

4.3.3 Action Detection

After the actors are detected over the input video streams and linked over time to generate tracks, actions of these actors are analyzed. For this purpose, we use the ACAM action detection module discussed in Chapter 3.

While observing complex events with multiple actors, humans do not assess each actor separately, but infer from the context. The surrounding context provides essential information for understanding actions. Inspired by object detection models, traditional action detection modules extend region of interest (RoI) pooling to videos and analyze actors by cropping individual features. This approach does not model context explicitly which is problematic for action detection tasks. To this end, ACAM replaces RoI pooling with an attention module, which uses weights to rank each spatio-temporal region's relevance to a detected actor instead of cropping. These weights are referred as Actor-Conditioned Attention Maps (ACAM). The resulting actor-conditioned features focus the model on regions that are relevant to the conditioned actor from the whole scene.

Using this module, actions of actors can be analyzed from the scene and interactions can be efficiently modeled using the context. Previous modules so far generated sets of tracks which represent each actor's bounding box location over time.

The simplest way to use these coordinates would be to crop actor bounding boxes over time and combine them into tubelets. However, as the scene context is used in the ACAM model, surroundings of the actor is also required for detecting actions. Cropping

actors directly will lose information about the surrounding context.

Following ACAM's inference procedure, which is to analyze entire input with CNN features and extract attention maps for every pixel location, is computationally unfeasible for surveillance-camera videos. As the field of view is significantly larger in surveillance-camera type videos compared to ACAM's dataset, inputs are higher resolution and multiple irrelevant events can be happening simultaneously.

With the assumption that for each actor only the close surroundings are relevant, for each actor we extract a large and square bounding box centered on that actor's bounding box. This operation is repeated for every frame in the track and a person tube (person regions over time) is extracted. These tubes include the close surroundings and are shown in Figure 4.4. In that example, we can see that two people who are talking to each other is captured in both actor tubes which can been analyzed by the ACAM module.

As the input to ACAM is $32 \times 400 \times 400$ (time, height, width), frames across tubes are resized to 400×400. The action model is learned to represent 2 seconds interval as 32 frames. This is also preserved in this implementation as framerate is essential in temporal modeling of actions. If this is changed during inference actions such as walk and run will be confused with each other. Additionally, as the action model analyzes 32 frames at a time, overlap across analyzes frames becomes another design choice. Analyzing every 32 frames with 31 frame overlap becomes unfeasible as the action is mostly the same, it includes redundant information and it is computationally expensive. Experimentally, we have found 75% overlap to give best results and capture short living actions such as picking up an object.

This concludes the module details in our framework. Three main modules are described in this section: the object(actor) detectors, the tracker, and the action detector. One important detail to note is that running these modules sequentially is not an efficient implementation and is unable to achieve real-time performance. In the following section,

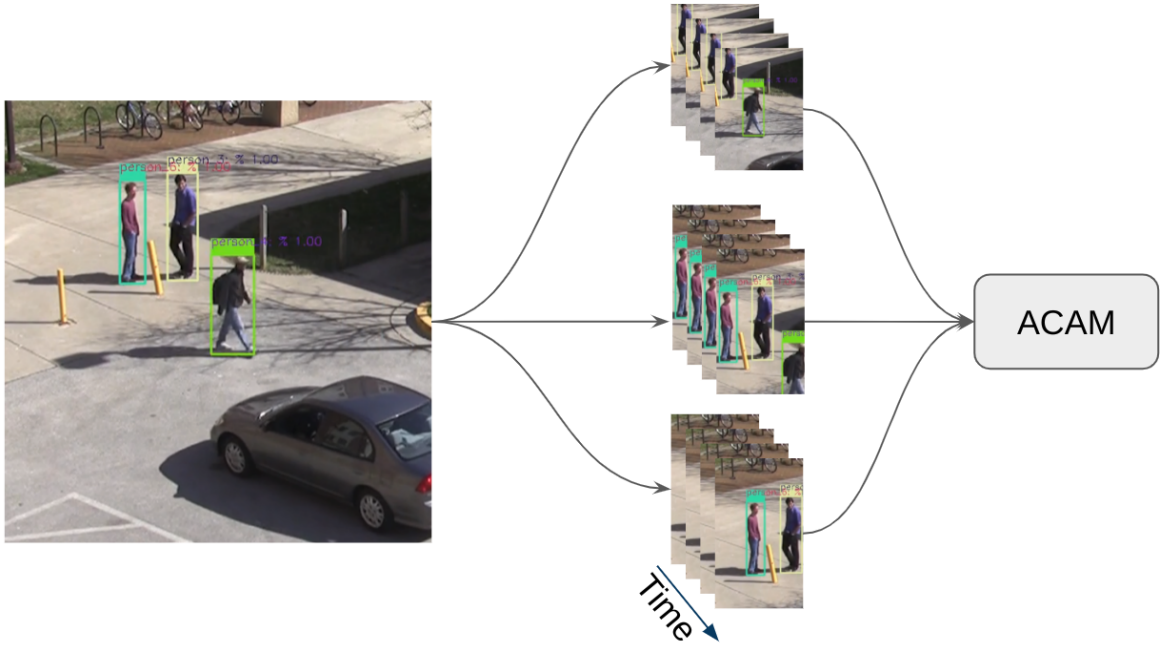


Figure 4.4: Actor tubes extracted from the actor tracks over time. A larger context for each actor is included as context has valuable information about the actions and our ACAM module models the surrounding context for each actors.

we will describe our implementation and that achieves real-time performance.

4.4 Real-time implementation

Each module in our framework uses a deep neural network to achieve their tasks. Individual modules differ significantly in their computational costs where action detection being the most expensive, then object detector and finally the lowest the tracker (re-identification CNN). These modules depend on each other and feed each other their detection results. Simplest implementation in this setting will use these modules sequentially and run a module when needed. However, such implementation will be slowed by the differences in running speeds of each modules and will be inefficient. In order to achieve real-time performances, we implement a multi-processing approach which can

utilize multiple GPUs.

4.4.1 Multiprocessing

The modules sequentially require each other's outputs to analyze their own inputs. In order to achieve multi-processing in such a sequential setting, a latency between the input and the output needs to be introduced. With a latency introduced, outputs of the individual modules can be stored up in queues to be processed whenever the next module is ready. This allows us to run these modules in parallel on multiple GPUs for faster processing.. Experimentally, with about 2 seconds of latency between the input and the outputs, modules are able to run in parallel with high efficiency. Figure 4.5 shows these processes and the GPU utilization for each module. In this implementation, we dedicate the same GPU to object detector and tracker as they both can run efficiently on a single GPU.

4.4.2 GPU Optimizations

As there is a latency between the input and the output, another improvement that can be made is batched processing of object detector. Instead of processing each frame individually, multiple frames can be processed at a single run which is a more efficient process in terms of speed (image/second). In our implementation with SSD-MobileNetV2, we use a batch size of 16. One limitation in this approach is the GPU VRAM size. If a larger model is used, smaller batch size is required depending on the GPU.

Input to the action detector is 32 frames and we have 75% overlap between their intervals. This means that every time the action detector runs, it only uses 8 new frames. Uploading 32 frames to the GPU is a slow process without pre-fetching. In order to reduce the redundant GPU transferring, we implement a frame queue on the

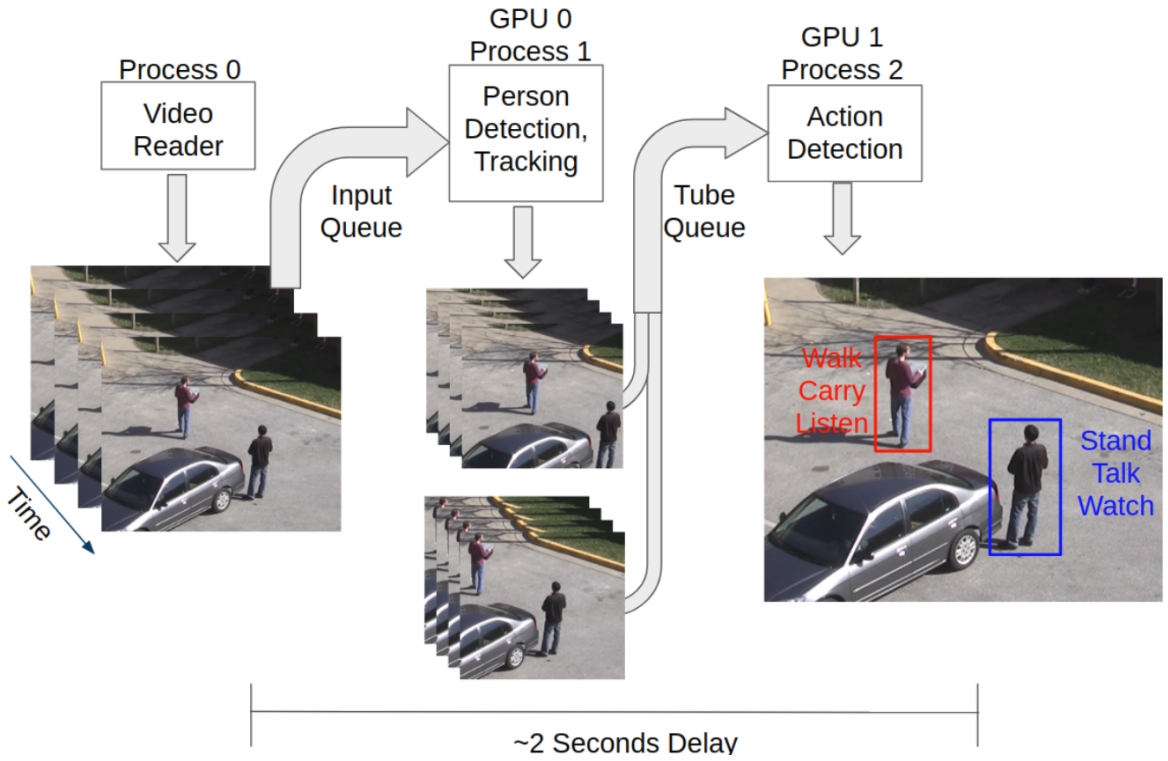


Figure 4.5: Visualization of multi-process multi-gpu implementation

GPU which updates 8 frames at a time while the previous 24 frames are preserved in the queue and stay on the GPU. This adds about 30-40% speed improvement in the overal framework.

The planned use for this model was for camera views such as surveillance videos where multiple people and multiple event can be observed. For each detected person, instead of using the whole input frame, we crop a larger area centered on each detected person tube to include surrounding context. However, uploading a different cropped region to the GPU for each detected person is also a slow process. Instead of uploading the cropped tubes individually, the whole frame is uploaded to the GPU initially and regions(tubes) of interest are extracted on the GPU.

Using these optimization with object detector SSD-MobileNetV2, tracker Deepsort

and action detector ACAM, we are able to achieve 16 frames per second on 2 Nvidia GTX1080Ti GPUs. Note that our action detector is trained also at 16fps and the input video rate is 16fps, thus achieving a real-time overall performance.

4.5 Applications

4.5.1 Atomic Action Detector

Combining this framework with the action detector ACAM trained on the AVA [12] dataset, we implement a system for atomic action detector. This real-time atomic action detection framework is open sourced and is available on GitHub[2].

As the AVA [12] dataset contains atomic actions, the resulting system can act as a general-use video-action analysis tool which can continuously detect human actions. We demonstrate this by providing qualitative results of our system on Figure 4.6 for various datasets. These include the vehicle-mounted camera dataset KITTI [4], surveillance dataset VIRAT, webcam videos and campus camera network. Action detection models such as this allow wide variety of applications to be implemented.

KITTI [4] dataset contains videos from vehicle mounted cameras and is used for generating models for self-driving cars. Detecting objects and people is an essential part of self-driving car perception models. However, additional information can be obtained using action detection models. Action detection can analyze if a pedestrian is walking or standing, if they are in a distracted state such as using their phones or if a pedestrian needs additional attention by using waving gestures. Detecting such actions have the potential to improve a vehicle's understanding of surroundings and improve safety for both the vehicle and pedestrians significantly. Qualitative results are shown on Figure

[2]Real-time action detection repository: https://github.com/oulutan/ACAM_Demo

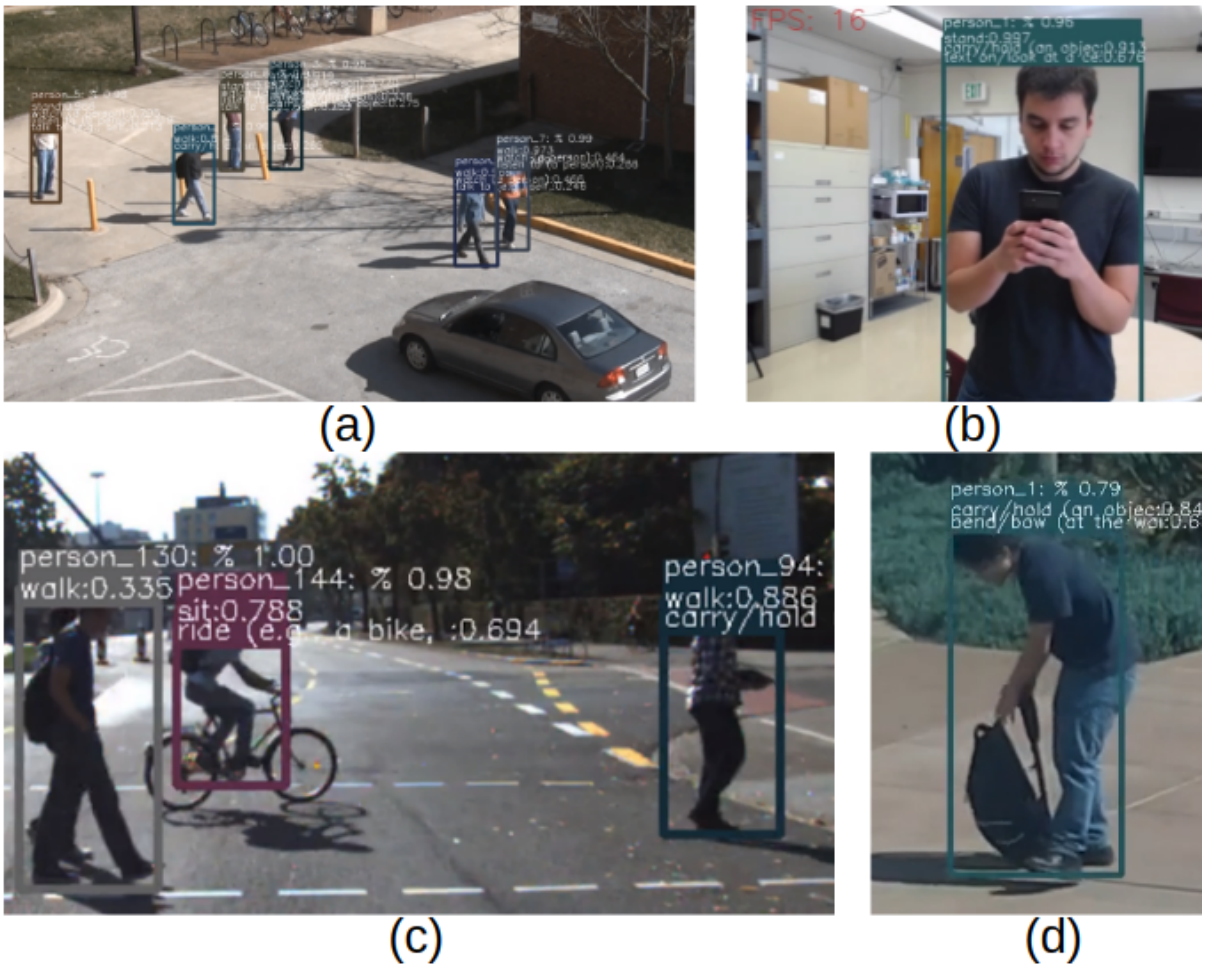


Figure 4.6: Qualitative results of ACAM video action detection framework shown on different sources. a) VIRAT dataset [10], b) Webcam inputs at 16 fps, c) KITTI [4] autonomous driving dataset, d) Campus camera network videos.

4.6-c.

VIRAT [10] dataset contains videos from camera networks. The dataset includes annotations for humans, cars, objects and some objects such as boxes. Action detection models for such camera inputs have the potential to improve and automate the security systems.

As these datasets include variety of video settings, we evaluate the detection performance of the actor detection module. Our model is trained on the MS-COCO dataset

which transfers to most use cases without fine tuning. We provide the results in Table 4.3.

Actor Detection	F RCNN COCO
COCO	32.05
AVA	95.97
VIRAT	30.44
KITTI	54.57

Table 4.3: Detection AP results for the Faster-RCNN archtiectures with Resnet[20] backbone. VIRAT[10] dataset contains videos from multiple cameras and KITTI[4] dataset contains videos from vehicle mounted cameras and is used for autonomous driving tasks.

4.5.2 Gesture Recognition

In order to demonstrate the generalizability of our framework and to show the additional applications, we train our action detection model ACAM with a gesture dataset. This dataset includes military gestures which are used as basic commands such as "follow me", "move forward" etc. This dataset includes synthetic data which is generated using a video game engine as additional data. The synthetic data is not used in these experiments. Figure 4.7 shows the gesture commands, real examples with actors and synthetic data from the dataset.

Initially, we quantitatively analyze the actor detections performance on this dataset. We split the data into training and testing splits by the actor and background. This way, the same background and actor will not be observed in the training and the testing data splits. Our ACAM achieve good performance in this dataset as well. Table 4.4 shows the Average Precision (AP) scores for the individual classes and the mAP score for all of the classes. As idle class represents the no action case, this task is a one of K classification task. Each sample is 2 seconds long. Table 4.5 shows the precision, recall

Figure 4.7: The gestures dataset: A) Reference gestures from the US Army Field Manual [16]; B) Real data examples; C) Synthetic data examples.

and F1 score values for this model at argmax classification setting. This means that instead of thresholding the prediction score, for each prediction maximum class score is taken as the guessed class. These results demonstrate that this architecture works well for this dataset as well.

Gesture Class	# test samples	AP
Idle	439	99.5%
Advance	71	94.2%
Attention	62	95.9%
Rally	30	95.2%
Forward	57	95.6%
Halt	101	99.2%
FollowMe	37	88.5%
Reverse	44	70.7%
mAP	-	92.4%

Table 4.4: Average Precision rates for individual gesture classes and the mean AP.

Gesture Class	Precision	Recall	F1-Score
Idle	99	90	94
Advance	91	83	87
Attention	93	81	86
Rally	73	90	81
Forward	87	91	89
Halt	89	100	94
FollowMe	90	73	81
Reverse	49	91	64
Average	84	87	84

Table 4.5: Precision, recall and F1-score values for argmax classification setting.

Qualitatively we test the gesture recognition models on webcam recorded videos to see the transferability to different settings. Figure 4.8 shows three example gestures in a lab setting from various angles.

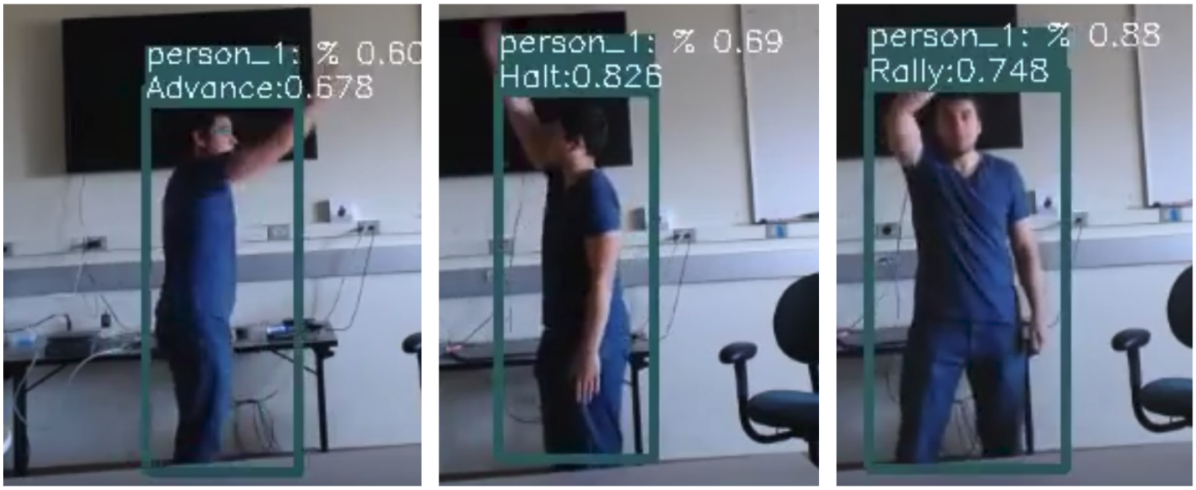
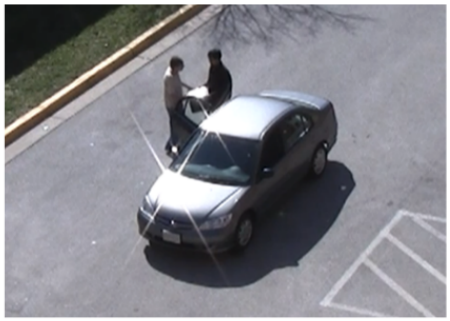


Figure 4.8: Qualitative results on gesture detection from a webcam stream in a lab setting from various angles.

Synthetic Data - Future work

Synthetic data may be critical to sustain the deep learning revolution. Deep learning models are very flexible and have been successfully applied in various domains, but they require vast amounts of labeled training data. Collecting and annotating all this data is costly, time-consuming, error-prone, and requires measures to address any potential ethical concerns. Synthetic data, in contrast, is less expensive, already comes with error-free ground truth annotation, and avoids many ethical issues.

As a future direcion, this dataset can be used for the analyzing the importance of synthetic data for gesture recognition task. The dataset includes the same types of gestures performed by real humans and computer generated synthetic data. This can be further utilized to reduce the amount of real-data required for action detection task which is especially costly and difficult to collect.



Cam 1: Person A hands off a box to Person B

Cam 2: Person B enters a campus building

Figure 4.9: An example complex activity. This scene can be described by the following two event descriptions, 1) A person leaves the car, takes a box and gives it to another person and gets back in to the car, 2) A person takes a box from another person and walks to a building and enters it. Images are taken from [10].

4.6 Complex Activity Detection Framework

Complex activities are events that can be defined as a sequence of simpler activities and can include multiple people and objects. These activities can describe events which can span longer time intervals and camera views. An example complex activity is shown in Figure 4.9. In this event, two people meet up next to a car, a person gives a box to another person and the person with the box enters the building in a separate camera view. Detecting such events pose multiple challenges such as detecting actions and interactions accurately within a camera view, stitching actions over space/time for temporal analysis and re-identifying actors within and across camera views.

This section focuses on implementing a framework which can automatically detect such complex activities building on top of our previous atomic action detection framework. Machine learning/deep learning methods have achieved great performances in recent years on tasks of object detection [34, 36, 33], tracking [14, 50], re-identification [15], action and video detection [12, 48, 59]. Most of these works implement an "end-to-end" approach, in which the whole task is learned directly from data without a structure or a modular framework. An "end-to-end" learned approach requires large amounts of

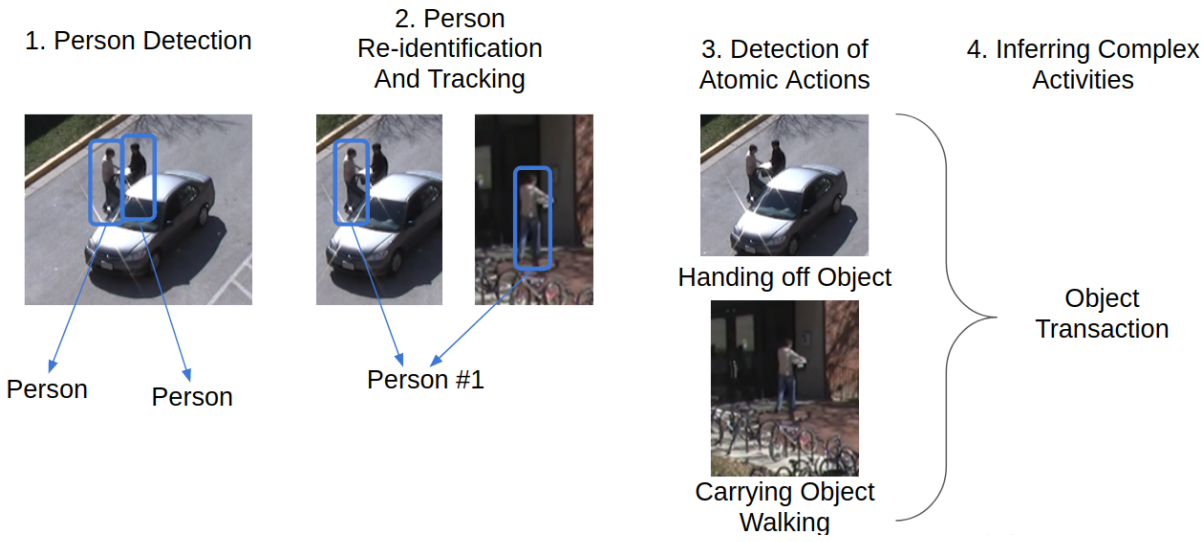


Figure 4.10: Breakdown of modules for detecting complex actions. A video analysis framework needs to 1) detect people in the scenes, 2) track and re-identify people across multiple cameras or different views, 3) detect atomic actions on detected people, 4) form sequences of atomic actions into complex activities. This section focuses on the entire 4 steps. Images are taken from VIRAT dataset [10]

data for even simpler tasks which makes it unfeasible for more complicated tasks such as complex activity detection. It is not feasible for a single model to learn to detect the person, track them over time, detect and analyze their actions. Additionally, complex activities are sequential by definition since they can be created from any set of individual actions/events. This creates combinatorially many complex activities from any set of simpler actions. Collecting data and training models for every combination is not a feasible approach (E.g..: 1-Pick up box from **person** and carry, 2-Pick up box from a **car** and carry).

In order to address these issues, we implement a modular approach. Deep learning models perform extremely well in their individual simpler/well-defined tasks. These individual models can be combined in order to achieve the larger final goal which is to detect complex activities. Therefore, we breakdown this task into multiple modules with individual goals. Figure 4.10 shows the breakdown of modules for the task. Our video

analysis framework consists of 1) object detectors, 2) object trackers and re-identifiers, 3) atomic action detectors and 4) complex activity detectors which combine atomic actions.

Research noted in this section is done in collaboration with Prof. Ramesh Govindan and Xiaochen Liu from the Networked Systems Laboratory at the USC. Initial results of this research are published in [87].

4.6.1 Complex Activities

Complex activities of interest are defined by a set of rules over time. These rules include people, objects, actions/interactions and define their temporal relations as a sequence. Let R_t define a rule at a single time instance:

$$R_t =< person_i, action > | < person_i, action, person_j > | < person_i, action, object_k > \tag{4.4}$$

A complex activity can be defined as a sequence of R_t for $t \in [1, T]$. Rules at different time steps will share object and person ids which makes it possible to define meaningful activities over time. A complex activity C_i can be defined as:

$$C_i =< R_1, R_2, ..., R_T > \tag{4.5}$$

Given a set of complex activities C_is defined by these rules, the task is to match the pre-defined actions to the detected actions in the scene. For every C_i, model will continuously look to match R_1 with the detected actions. Whenever there is a match, the model will keep trying to match the rules of the next time instance. A complex activity C_i is detected whenever all R_1 to R_T is matched.

Case for Atomic Actions

The complex activities of interest are higher level activities where multiple people/objects can be involved, can span various time intervals and happen rarely on videos. Detecting such models with a single module is challenging as exhaustively listing these high-level activities as separate classes and training modules on them is an impractical approach [12]. However, if the actions are limited to shorter time scales or atomic body movements, detection tasks become more defined. In order to achieve detection of complex activities, we initially detect these atomic actions and combine them temporally.

An atomic action is defined as any action that can be defined by 1-3 words or a single body movement. These actions can be detected by simpler models for short time intervals. This definition of atomic actions allows us to continuously analyze each persons' states as basic actions happen continuously as well. Compared to detecting high-level activities in an discrete manner, this continuous approach allows us to better analyze actions in videos.

Hybrid Approach

In order to detect these atomic actions, we implement a hybrid approach that combines outputs from a model which looks for spatial configurations between objects (e.g.: near, approach, close, overlap, move) and our action detection framework ACAM. Figure 4.11 shows the hybrid approach.

Spatial configurations between people and objects/other people can provide essential information about their interactions. These spatial configurations can be utilized by tracking the bounding box locations and their temporal dynamics over time. A simple example to this can be seen in Figure 4.11. A walking person will have a moving bounding box and if a bag is also moving then this will define the "carry bag" action.

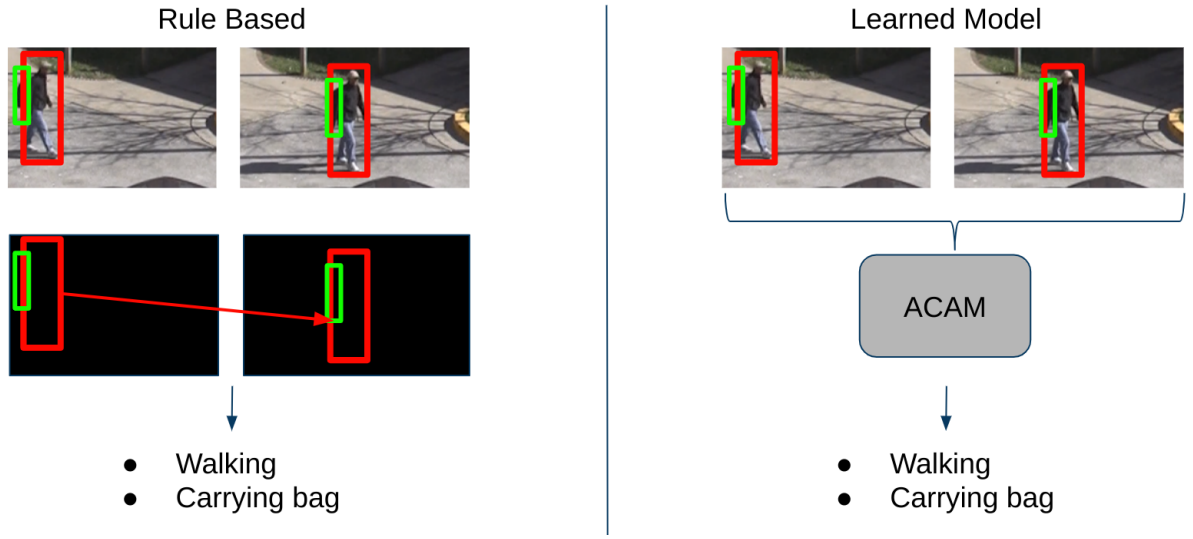


Figure 4.11: Hybrid Approach for action detection. Rule Based methods use basic bounding box coordinates for objects and use their relations to define actions. Learned method uses an action detection model trained on AVA [12] dataset which detects atomic actions.

In addition to the spatial configurations, our ACAM module can provide a higher level understanding of actions. As this model is trained for detecting the temporal dynamics of atomic actions on humans, it can detect cases such as "carry object" or "use phone" without explicitly detecting the objects. This can be effective in cases where the cameras are located far away from people and objects such as phones are missed by object detectors.

Combining these two modules, our complex activity detection module is able to improve the video understanding task.

4.6.2 Extension to Drone-Mounted Cameras

A promising extension of activity detection is using drones to detect such actions. Drones equipped with cameras are used in many applications including agricultural, aerial photography, fast delivery, surveillance, military applications etc. With a mounted camera, these drones have the potential to respond the visual commands from users, an-

alyze and navigate the surroundings autonomously, deliver packages, provide intelligence to security forces and many more applications.

One major challenge in this setting is the significantly different view-angles, altitudes and distances from the target. These factors affect the object detectors' performance and they start failing to detect the objects of interest. These detectors are usually trained on datasets like MS-COCO[3]. These datasets are collected online and consist of images taken with personal cameras or mobile phones. This creates an impact on object detectors' performance on drone-view images. In order to detect the activities of targets, the targets are required to be detected first by an object detector.

Object Detection from Drones

In order detect objects from drone-mounted cameras, we need to train object detector models on similar data recorded in similar conditions. Recently, a large-scale drone-view dataset named VisDrone [5] is released to the public. This dataset is recorded from drones in various cities of China. As this is a large-scale drone dataset, we train an object detector to be used in our complex activity framework.

The dataset annotates the following object classes in the drone-recorded videos:

Objects: Pedestrians, People, Cars, Trucks, Vans, Buses, Motors, Bikes, Tricycles.

One design choice for the dataset is to split pedestrians and people as separate categories. *Pedestrian* are humans who maintain a standing pose or walking compared to *Person* class which might be sitting (E.g.: on a bike, on a bench) or in any other poses (E.g.: basketball, swimming).

Using this dataset, we train our object detector. Our detection model is based on Faster RCNN architectures [34]. As a backbone CNN in our Faster RCNN implementation, we use ResNext-101 32x8d model [88] which achieves great performance on Imagenet[2] and achieves good results as a backbone CNN for MS-COCO[3] detection

models as well.

Object Class	AP
Pedestrian	43.17
Person	32.30
Bike	16.22
Car	76.51
Van	37.44
Truck	34.17
Tricycle	23.96
Awning-tricycle	12.12
Bus	53.54
Motor	41.76
Others	4.89
mAP	31.50

Table 4.6: Per class average precision (AP) and the overall mean AP (mAP) scores at intersection over union rate of 0.5.

Table 4.6 shows the average precision (AP) scores for each individual class and the mean AP across all classes at intersection over union(IoU) threshold of 0.5. IoU value measures how well the prediction box aligns with the ground truth boxes. If the IoU value of a prediction box and a ground truth is greater than this value (0.5) and the predicted the class is correct, then the prediction is a true positive.

Figure 4.12 shows detection results on an image from the VisDrone dataset. This image is taken from the test set and is not used during the training process. This image also shows some of the challenges in this dataset. The bounding boxes of object vary both inter class and intra class. Additionally, the person and pedestrian classes usually are observed very close together (walking together) and can easily be missed in such large fields of view.

We qualitatively test our object detection module on VIRAT[10] air dataset. This dataset contains videos recorded from drones which are significantly different than the VisDrone dataset that our model is trained on. Detection results on a few images are

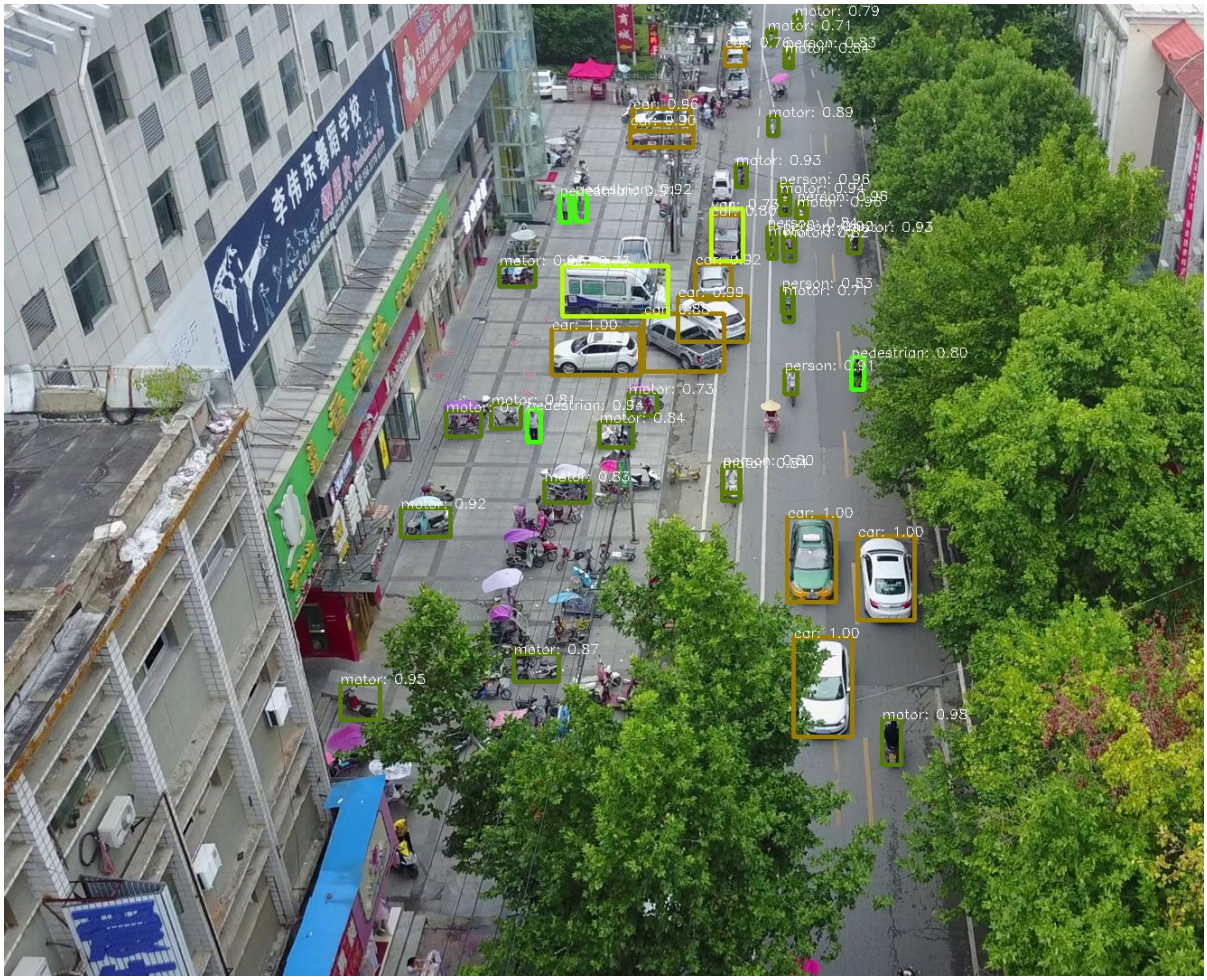


Figure 4.12: Qualitative results on an image from VisDrone[5] test set. Different colored boxes indicate different object classes.

shown in Figure 4.13. The resolutions of the videos are lower, altitude of the drones are higher and there are rapid camera movements. However, our model is still able to detect objects with some false positives and class confusion.

Drone-View Action Detection

We replace the object detector in the Complex Activity Framework with the drone-view object detection module to detect actions. One challenge is that the ACAM action detection module is not trained for such view angles and it starts failing. In this setting

Figure 4.13: Qualitative results on images from VIRAT[10] Air dataset. Different colored boxes indicate different object classes. The altitude of the drone is significantly higher and the quality of the video is visibly worse than the Visdrone dataset. However, our model is still able to detect objects with some false positives and class confusion.

we only use the rule-based spatial logic to detect the actions/interactions.

We define rules for multiple activities that we are interested in detecting. Following are couple example rules for these activities where each rule part defines a single time interval:

- Get Inside Car: Person approach Car → Person near Car → Person Disappear
- Walking Together: Person near Person → Person move → Person near Person
- Interacting with car: Person approach Car → Person near Car → Person near Car
- Get out of a Car: Person appear and Person near Car → Person near Car

We qualitatively test these rules on videos recorded from drones. Figure 4.14 shows the detections of some of these rules. This figure only shows the bounding boxes as the data is private.

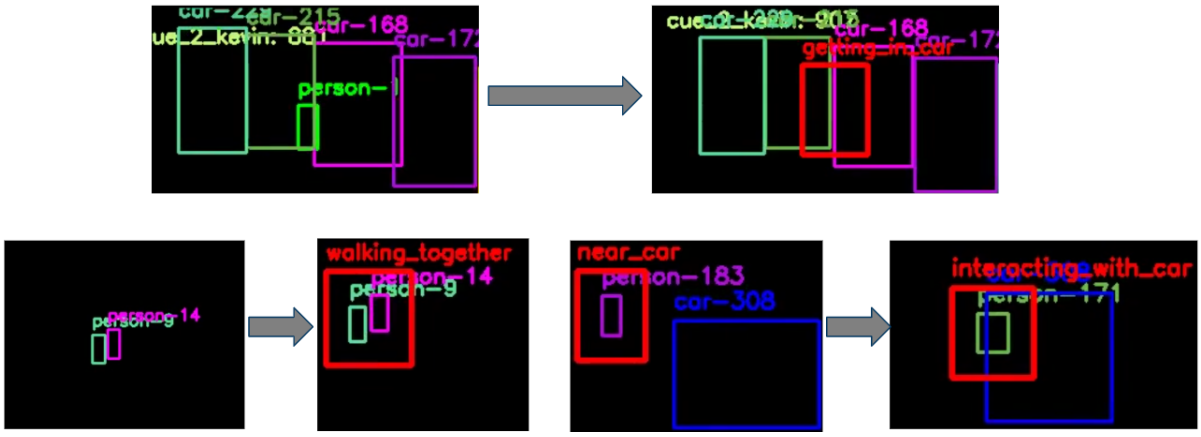


Figure 4.14: Couple of activity rules detected on videos recorded from drones. Only bounding box locations are showed as the data is private.

4.7 Conclusions

Detecting actions near real-time has a wide variety of applications. In this chapter, we presented an end-to-end, modular framework which analyzes the human actions from a given video. This framework contains of three main modules which are object (actor) detection, tracker and action detection modules.

Additionally, we demonstrated the generalizability of our framework by training a model for gesture detection. This model demonstrates that our application setting is not limited to atomic actions and with the data, various actions/gestures models can be trained and this framework can be used for inference.

Combining the action detection model from the previous chapter, this chapter builds a generalizable framework for action detection. Building on top of this framework, we combine these sequences of atomic actions into complex activities which is one of the main goals for security systems for video understanding.

As an extension, we demonstrated that this framework can be used with input videos recorded from drones. Due to significant changes of view-points, object detectors start

failing for drone views. In order to address this issue, we implemented a drone object detector and demonstrated the results on drone videos.

Chapter 5

VSGNet: Spatial Attention for Human-Object Interactions

5.1 Introduction

Comprehensive visual understanding requires detection frameworks that can effectively learn and utilize object interactions while analyzing objects individually. This is the main objective in Human-Object Interaction (HOI) detection task. In particular, relative spatial reasoning and structural connections between objects are essential cues for analyzing interaction. For modeling interactions, in this chapter we study Human-Object Interaction detection from images task.

Human-object interaction is an important task for visual understanding. Datasets for video action detection, which we used in previous chapters, usually don't have explicit interaction labels and focus on actors. Even though interaction classes exist in video datasets, they don't label the interacted object or humans. In order to learn such interactions, we focus on detecting human-object interactions from images. Even though this input is missing the temporal dynamics, they are usually more comprehensively anno-

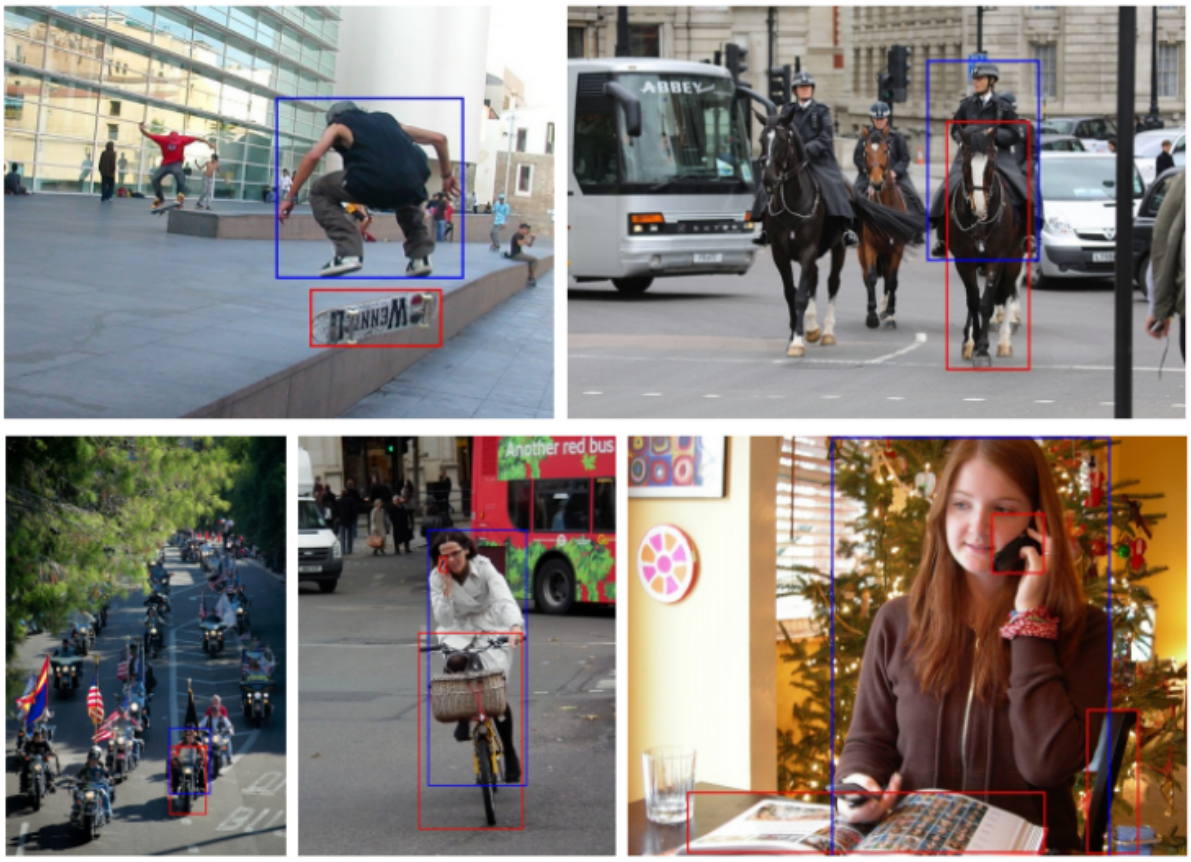


Figure 5.1: Example human-object interactions from the V-COCO [9] dataset. Current human of interest is shown by the blue box while the objects that this person is interacting with is shown in red boxes.

tated for objects and their interactions with humans. In this setting spatial configurations and structural relations provide essential information for detecting if the human-object pairs are interacting and classifying the interaction.

Compared to tasks such as image recognition or object detection, interaction detection task requires a deeper understanding of the scene in which the objects are needed to be modeled individually and also collectively. In scenes with multiple humans and objects, connections can grow exponentially and models need efficient ways to detect the interactions and filter out pairs that are not interacting. Some example interactions are shown in Figure 5.1

The task of detecting human object interaction (HOI) in images refers to detecting the interactions between a human and object pair and localizing them. HOI detection can be considered a part of the task of visual scene understanding [89, 90, 91], visual question answering [92, 93, 94, 95], and activity recognition in videos [87, 96, 51]. Although there has been significant improvements in recent years for detecting and recognizing objects [30, 34, 97], HOI detection still poses various challenges. For example, interactions usually happen in a subtle way, same types of relations vary significantly across different settings, multiple humans can interact with the same object or vice-versa, and different relations might have visually subtle differences [9, 42].

Most of the existing methods in HOI detection task [98, 99, 100] follow a similar structure. Using an object detection framework, human and object features are extracted. These features are paired exhaustively along with some other features (e.g. pose, relative geometric locations) [98, 99] and then fed into a multi-branch deep neural network to detect the relationship between humans and objects. Even though this approach achieves good results for detecting HOIs, it does not explicitly utilize the interaction information or the spatial relations between pairs. HOIs such as person on a skateboard or a person holding a bat have well defined spatial relations and structural interactions which should be leveraged in this detection task.

5.2 Spatial Configurations for HOI Detection

Building up from the rule-based spatial configurations from the previous chapter, we introduce a novel neural network model which can utilize the spatial configurations of the human-object pairs and their structural connections. The model uses spatial configurations to generate attention on each human-object pairs, see 5.2.

For utilizing spatial configurations, VSGNet uses a spatial attention branch that ex-

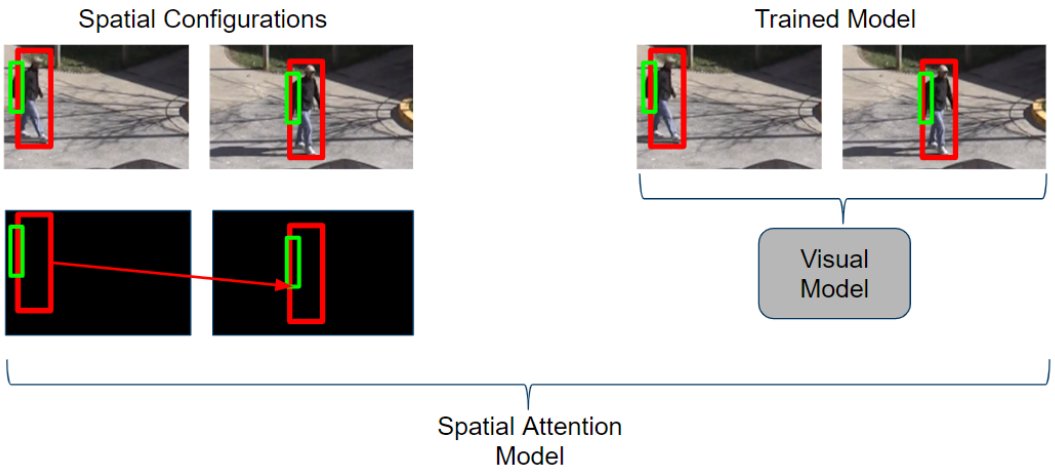


Figure 5.2: Previous works utilize trained models and the spatial configurations separately. In this chapter we focus on generating a single spatial attention model which can use the spatial configurations to improve visual features.

plicitly uses the spatial relations of the pairs to refine the visual features. Instead of modeling humans and objects individually, our attention module uses the spatial configurations of the pairs to extract attention weights which refine the visual features. Although a few past works [42, 99] have used such spatial configurations as features for classification directly, these models do not combine the visual information with spatial information. Spatial configurations are more useful for refining the visual features and providing an attention mechanism for modeling the interactions of the human-object pairs explicitly.

For modeling the interactions, an image can be represented as a graph. Nodes in this graph are the humans and objects, with edges defining their interactions. Our proposed model utilizes the interaction proposal scores as the intensities of the edges in the graph. Interaction proposal scores are generated from the spatially refined visual features and they quantify if the human-object pair is interacting.

The VSGNet for HOI detection refines the visual features using spatial relations of humans and objects. This approach amplifies the visual features of spatially relevant pairs

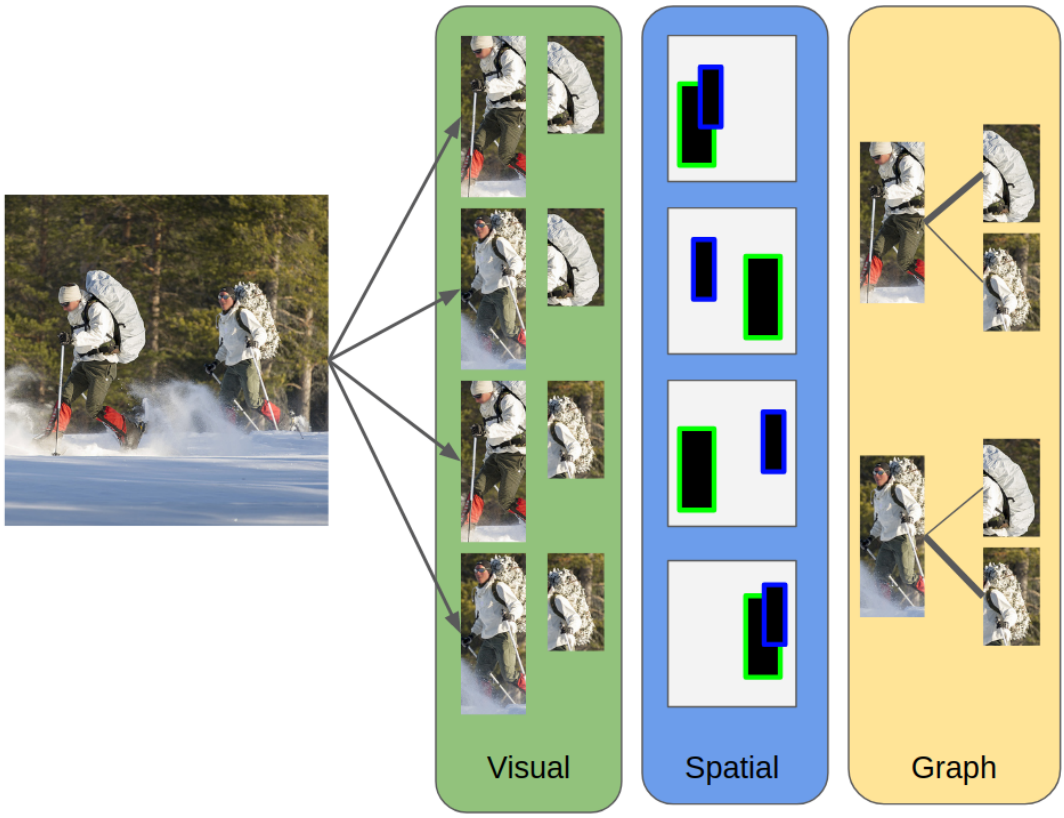


Figure 5.3: Visual, Spatial and Graph branches of our VSGNet model. Visual branch analyzes humans/objects/context invidually, Spatial branch uses spatial configurations of the pairs to refine visual features and the Graph branch utilizes the structural connections by Graph convolutions which uses interaction proposal scores as edge intensities between human-object nodes.

while damping the others. Additionally, this model uses graph convolutional networks to model the interactions between humans and objects. The resulting model consists of multiple specialized branches. We evaluate our model on V-COCO[9] and HICO-Det[42] dataset and demonstrate 4 mAP (8%) improvement over the state of the art methods on V-COCO and 2.8 mAP (15%) on HICO-Det. Figure 5.3 visualizes the three branches of the VSGNet model.

Technical Contributions:

- We introduce a new spatial attention branch that leverages the spatial configuration

of human-object pairs and refines the visual features such that spatially relevant human-object pairs are amplified.

- We use a graph convolutional branch which utilizes the structural connections between humans and objects. The interaction proposal score, generated from the spatially refined features, are used to define the edge intensities between human and object nodes.

- We implement a robust, multi-branch pipeline that contains Visual, Spatial and Graph based branches named VSGNet. This model achieves state-of-the-art results for HOI detection task on V-COCO dataset.

5.3 Related Work

Object Detection: For detecting HOIs the first step is to detect humans and objects properly. With the recent object detection frameworks like RCNN [30], Faster RCNN [34], YOLO [33], Feature Pyramid Network [101] and SSD [36], models are able to detect multi scale objects robustly in images. Following this we utilize a pre-trained Faster-RCNN model in our network for detecting humans and objects. Additionally, we utilize the region proposal network idea from Faster-RCNN and extend it to interaction proposals which predict if an human-object pair is interacting.

Human Object Interaction: Activity recognition is a research area in computer vision that has received interest for a long time. There are different datasets like UCF-101 [6], Thumos [102] with a focus on detecting human actions in videos. But in these datasets, the goal is to detect one action in a short video which is not representative of real life scenarios. To extend human activity recognition in images Gupta et al. [9] introduced V-COCO dataset and Chao et al. [42] introduces HICO-DET dataset. These

datasets are different from the previous datasets as they require models to explicitly detect humans, objects and their interactions. This extends the task to include detection of human activities while localizing the humans and the objects.

For the HOI detection task, Gkioxari et al. [100] proposed a human-centric approach arguing that human appearance provides strong cues in both detecting the action and localizing the object. This method does not consider interactions where the object is far away from the human. Qi et al. [103] proposed a graph based network which depends on detecting an adjacency matrix between various nodes (here, nodes are humans and objects) but does not utilize any spatial relation cues between pairs. Kolesnikov et al. [104] incorporates the HOI detection process with the object detection by individually analyzing humans and objects without considering the relationship between the pairs.

Gao et al. [99] proposed an attention network based on the previous work of [49]. They derived an attention map from the human and object features over the whole convolutional feature map. Although they used a binary spatial map similar to [42], they use the spatial map to extract features and concatenate them with human visual features. As these are two completely different features defining separate things, concatenation does not enforce spatial configurations as much as an attention mechanism. To address this in our network we use the spatial features as attention maps which refines our visual features.

Li et al. [98] integrated pose estimation with the iCAN [99] and predicted the interaction probabilities between a human and object pair. This demonstrates that the pose estimation can help with the HOI task. These methods however, do not explicitly leverage the interaction probabilities to detect the relational structure between the human and object pairs.

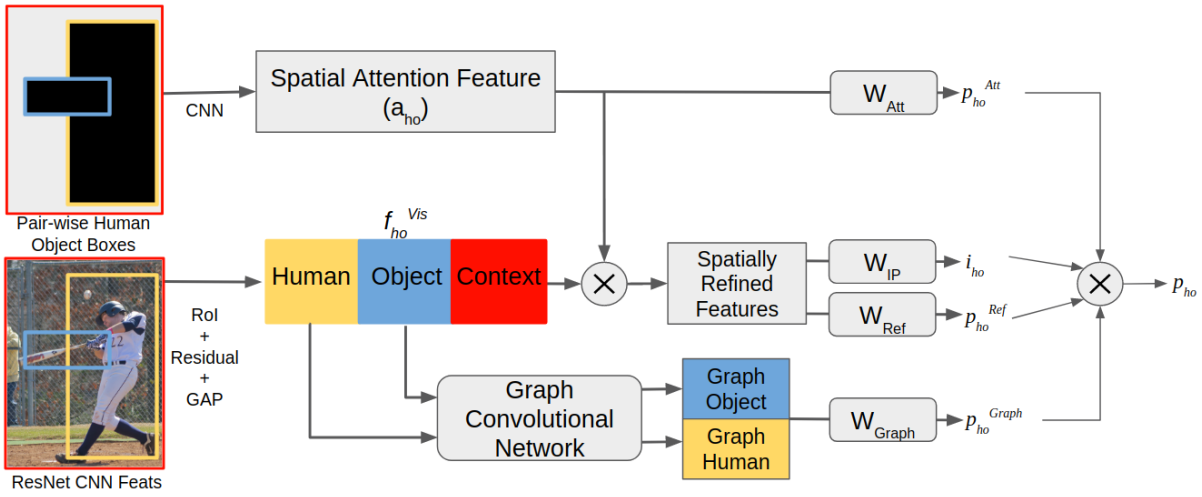


Figure 5.4: Model Architecture. Rounded rectangles represent operations whereas sharp rectangles represent extracted features. $\otimes$ operation represents element-wise multiplication. The model consists of three main branches. Visual branch extracts human,object and context features, Spatial Attention branch refines the visual features by utilizing the spatial configuration of the human-object pair and the Graph Convolutional branch extracts interaction features by considering humans/objects as nodes and their interactions as edges. Action class probabilities from each branch and the interaction proposal score are multiplied together to aggregate the final prediction. These operations are repeated for every human-object pair.

5.4 Proposed Method

This section introduces our proposed VSGNet for detecting human-object interactions (HOI). From each given image, the task is to detect bounding boxes for the humans, objects and correctly label the interactions between them. Each human-object pair can have multiple interaction labels and each scene can include multiple humans and objects in them. We simplify the task by running a pre-trained object detector which detects humans and objects in an image.

Detecting the interactions between human-object pairs is a challenging task. Simple methods such as extracting features from human and object locations individually and analyzing them are ineffective as these methods ignore the contextual information of the surroundings and the spatial relations of the human-object pair. Extensions such as using

union boxes to model the spatial relations/context also fall short as they don't explicitly model the interactions. To address these issues, we propose a multi-branch network with specialized branches. The proposed VSGNet consists of the Visual Branch (Section 5.4.2) which extracts visual features from human, object and surrounding context individually; the Spatial Attention Branch (Section 5.4.3) which models spatial relations between the human-object pair; and the Graph Convolutional Branch (Section 5.4.4) which considers the scene as a graph with humans and objects as nodes and models the structural interactions. The proposed model architecture with the branches is shown in Fig.5.4.

5.4.1 Overview

The inputs to our model is image features $\mathbf{F}$ from a backbone CNN (e.g. ResNet-152 [97]) and bounding boxes x_h for human $h \in [1, H]$ and x_o for object $o \in [1, O]$. H and O represents the total number of humans and objects in the scene respectively. Bounding boxes are obtained from a pre-trained object detector. We define the objective of this model as:

- Detect if human h is interacting with object o with an interaction proposal score $i_{h,o}$.
- Predict the action class probability vector $\mathbf{p}_{h,o}$ of size A where A is the number of classes.

5.4.2 Visual Branch

This branch focuses on extracting visual features for the human-object pairs. Following the object detection methods, we use region of interest (RoI) pooling on the human/object regions to extract features. This operation is followed by a residual block

(Res) [97] and global average pooling (GAP) operations to extract the visual feature vectors for objects and humans.

$$\mathbf{f}_h = GAP(Res_h(RoI(\mathbf{F}, x_h))) \tag{5.1}$$

$$\mathbf{f}_o = GAP(Res_o(RoI(\mathbf{F}, x_o))) \tag{5.2}$$

where $Res_{\{\}}$ represents residual blocks, $\mathbf{f}_h$ and $\mathbf{f}_o$ are visual feature vectors of sizes R. This operation is repeated for each human h and object o.

Context plays an important role in detecting HOI. Surrounding objects, background and other humans can help detecting the interactions. We include the surrounding context in our network by extracting features from the entire input image.

$$\mathbf{f}_C = GAP(Res_C(\mathbf{F})) \tag{5.3}$$

where $\mathbf{f}_C$ is a feature vector of size R.

Finally, this branch combines all the visual feature vectors by concatenating them and projecting it by a fully connected layer.

$$\mathbf{f}_{ho}^{Vis} = \mathbf{W}_{vis}(\mathbf{f}_h \oplus \mathbf{f}_o \oplus \mathbf{f}_C) \tag{5.4}$$

where $\oplus$ is the concatenation operation, $\mathbf{W}_{\{\}}$ is the projection matrix, $\mathbf{f}_{ho}^{Vis}$ is the combined visual feature vector of dimension D which represents the human-object pair ho.

A simple way to achieve the HOI detection task is to use the feature $\mathbf{f}_{ho}^{Vis}$ directly for classifying actions. We implement this as a base model for comparisons.

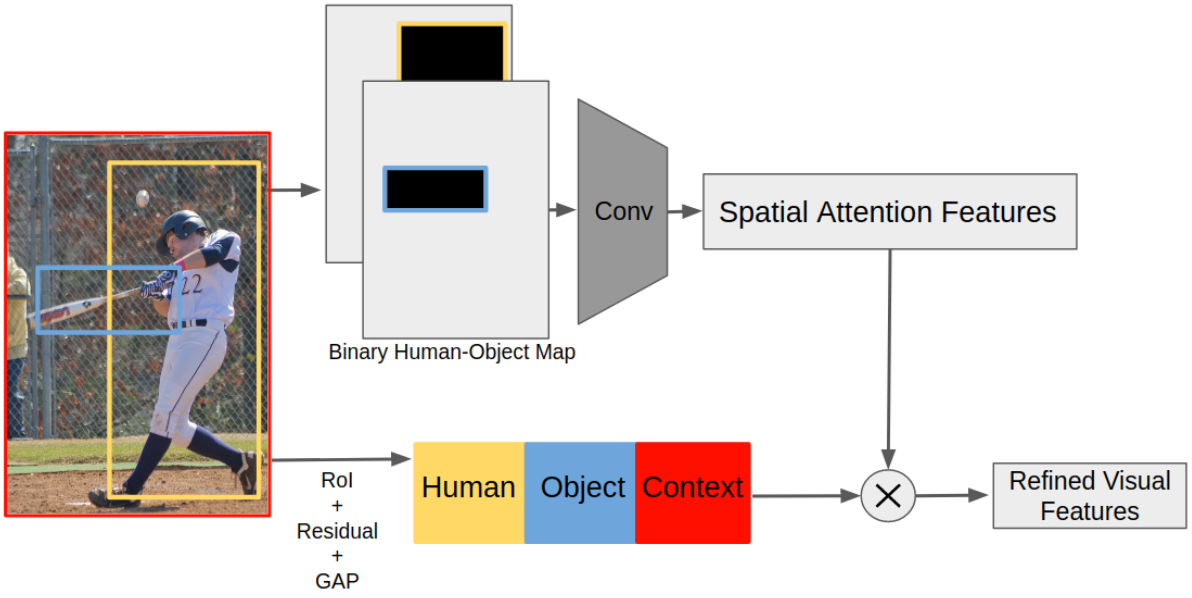


Figure 5.5: Spatial Attention Branch. Initially human, object and context visual features are extracted from the image using RoI pooling. Using binary maps of human and object locations, spatial attention features are extracted using convolutions. These attention features encode the spatial configuration of the human-object pair. Attention features are used to refine the visual features by amplifying the pairs with high spatial correlation.

5.4.3 Spatial Attention Branch

This branch focuses on learning the spatial interaction patterns between humans and objects. The main task is to generate attention features which are used to refine the visual features by amplifying the pairs with high spatial correlation. This branch is shown in Fig.5.5.

Given the human bounding box x_h and object bounding box x_o, we generate two binary maps. These binary maps have zeros everywhere except in locations defined by human and object box coordinates x_h and x_o for each map respectively. This generates a 2-channel binary spatial configuration map $\mathbf{B}_{ho}$.

Similar to [42, 99], we use 2 layers of convolutions to analyze the binary spatial

configuration map. This is followed by a GAP operation and a fully connected layer.

$$\mathbf{a}_{ho} = \mathbf{W}_{Spat}(GAP(Conv(\mathbf{B}_{ho}))) \tag{5.5}$$

where $\mathbf{a}_{ho}$ is an attention feature vector of size D and represents the spatial configuration of the human-object pair *ho*. As the objects and humans are defined in different channels, using convolutions on the binary spatial configuration maps $\mathbf{B}_{ho}$ allows the model to learn the possible spatial relations between humans and objects.

Since $\mathbf{a}_{ho}$ encodes the spatial configuration, it can be used directly to classify the HOIs as in [42]. We keep this classification as an auxiliary prediction but mainly use $\mathbf{a}_{ho}$ as an attention mechanism for refining visual features. Auxiliary predictions can be defined as:

$$\mathbf{p}_{ho}^{Att} = \sigma(\mathbf{W}_{Att}(\mathbf{a}_{ho})) \tag{5.6}$$

where $\mathbf{p}_{ho}^{Att}$ is the action class probabilities of size A and σ is the sigmoid function.

The attention vector $\mathbf{a}_{ho}$ and the visual feature vector $\mathbf{f}_{ho}^{Vis}$ are set to be the same size D. This allows us to multiply these two vectors together in order to refine the visual features with spatial configuration. We use $\mathbf{a}_{ho}$ as an attention function and multiply $\mathbf{a}_{ho}$ and $\mathbf{f}_{ho}^{Vis}$ elementwise.

$$\mathbf{f}_{ho}^{Ref} = \mathbf{a}_{ho} \otimes \mathbf{f}_{ho}^{Vis} \tag{5.7}$$

where $\otimes$ is element-wise multiplication and f_{ho}^{Ref} is the spatially refined feature vector of size D.

The refined feature vector is then used to predict the interaction proposal score of human-object pair *ho* and to predict the action class probabilities.

$$i_{ho} = \sigma(\mathbf{W}_{IP}(\mathbf{f}_{ho}^{Ref})) \tag{5.8}$$

$$\mathbf{p}_{ho}^{Ref} = \sigma(\mathbf{W}_{Ref}(\mathbf{f}_{ho}^{Ref})) \tag{5.9}$$

where i_{ho} is the interaction proposal probability of size 1 and $\mathbf{p}_{ho}^{Ref}$ is the action class probabilities of size A.

5.4.4 Graph Convolutional Interaction Branch

This branch uses a graph convolutional network to generate effective features for humans and objects. Graph convolutional networks extract features that model the structural relations between nodes. This is done by traversing and updating the nodes in the graph using their edges. In this setting, we propose to use humans and objects as nodes and their relations as edges.

Instead of having a fully connected graph, we connect each human with every object and each object with every human. However, without this simplification, proposed model can also be extended to fully connected settings.

Given the visual features $\mathbf{f}_h$, $\mathbf{f}_o$ and connecting the edges between humans and objects, graph features $\mathbf{f}'_h$ and $\mathbf{f}'_o$ are defined as follows:

$$\mathbf{f}'_h = \mathbf{f}_h + \sum_{o=1}^{O} \alpha_{ho} \mathbf{W}_{oh}(\mathbf{f}_o) \tag{5.10}$$

$$\mathbf{f}'_o = \mathbf{f}_o + \sum_{h=1}^{H} \alpha_{oh} \mathbf{W}_{ho}(\mathbf{f}_h) \tag{5.11}$$

where α_{ho} defines the adjacency between h and o and $\mathbf{W}_{oh}$, $\mathbf{W}_{ho}$ are mapping functions

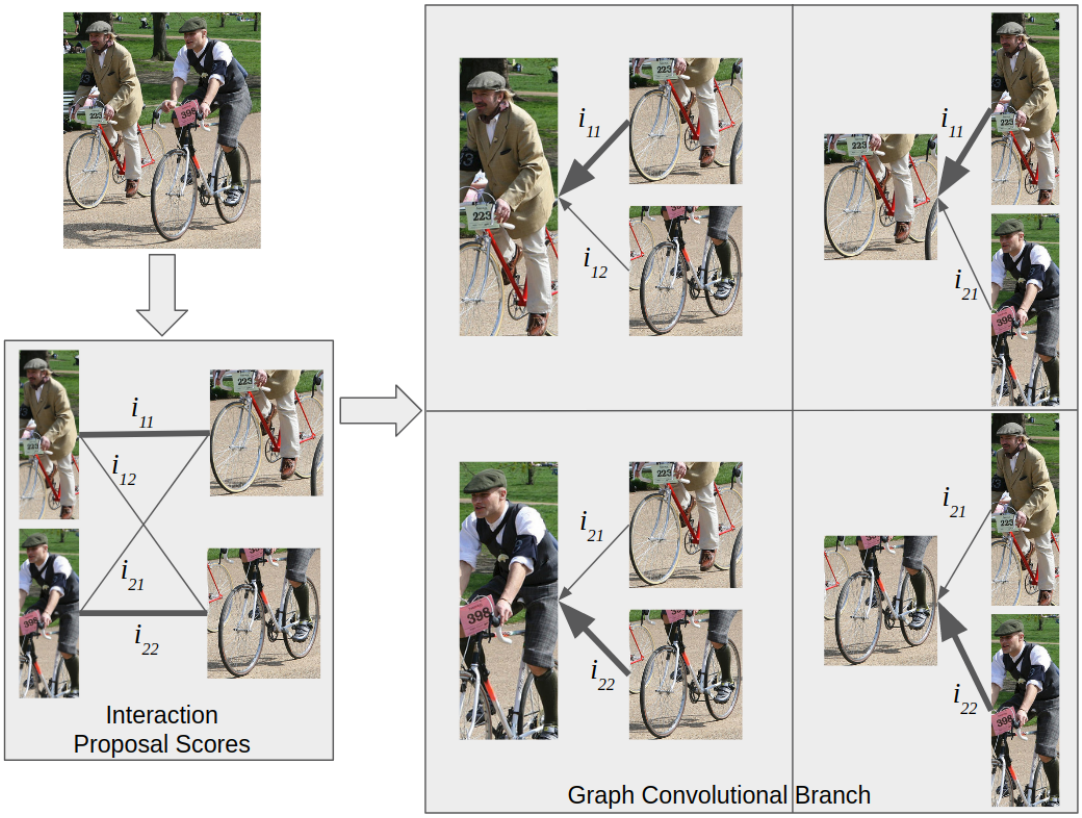


Figure 5.6: Graph Convolutional Branch. This model learns the structural connections between humans and objects. For this task, we define the humans and objects as nodes and only connecting edges between human-object pairs. Instead of using visual similarity as the edge adjacency, we propose to use the interaction proposal scores. This allows the edges to utilize the interactions between human-object pairs and generates better features.

which project the object features to human feature space and vice versa. Previous works [105, 103] defined the adjacency as visual similarity. In our task, however, adjacency defines interactions between nodes of visually unsimilar things which are human and object. Following this idea, we define adjacency values between h and o pair as:

$$\alpha_{ho} = \alpha_{oh} = i_{ho} \tag{5.12}$$

where i_{ho} is the interaction proposal score which are generated from the spatially refined

visual features and measure the interactions of the human-object pair. Pairing up the graph features, classification predictions are calculated as:

$$\mathbf{p}_{ho}^{Graph} = \sigma(\mathbf{W}_{graph}(f_h' \oplus f_o')) \tag{5.13}$$

where $\oplus$ is concatenation operation and $\mathbf{p}_{ho}^{Graph}$ is the action class probabilities of size A.

The graph convolutional branch is shown in Figure 5.6. This concludes all of the outputs of the proposed network. Finally we combine the action predictions and the interaction proposal scores by multiplying the probabilities similar to previous works [99, 98, 100].

$$\mathbf{p}_{ho} = \mathbf{p}_{ho}^{Att} \times \mathbf{p}_{ho}^{Ref} \times \mathbf{p}_{ho}^{Graph} \times i_{ho} \tag{5.14}$$

where P_{ho} is the final prediction vector of size A.

5.5 Experiments

We first introduce the datasets and our evaluation metrics along with our implementation details and then perform extensive quantitative and qualitative analysis on our model and show the improvements over the existing methods.

5.5.1 Datasets and Evaluation Metrics

Datasets: To evaluate our model's performance, we use the V-COCO [9] and HICO-Det [42] datasets.

V-COCO is derived from COCO [3] dataset. It has 10,346 images. 2533 images are for training, 2867 images are for validating and 4946 images are for testing. The training and validation set images are from COCO training set and the test images are from the

COCO validation set. Each person in the images are annotated with a label indicating one of the 29 actions. If an object in the image is related to that action then the object is also annotated. Among these 29 actions, four of them has no object pair and one of them(point) has only 21 samples. Following the previous HOI detection works, we are not going to report our performance in these classes. We report our performance for the rest of the 24 classes.

HICO-DET is a large dataset for detecting HOIs with 38118 training and 9658 testing images. HICO-Det annotates the images for 600 human-object interactions. However, these are not 600 separate classes (as in Kinetics 700[8] for example). These 600 classes contain combinations of 117 unique interaction labels with 80 unique objects. In this setting, "Chase-Cat", "Chase-Dog" and "Chase-Bird" becomes different classes in this 600-class evaluation scheme. This results in 600 action-object combinations as classes from 117 unique interactions and 80 objects. Mean across 600 class Average Precision (AP) values are calculated to achieve final score. This also limits the generalizability of the resulting models as interaction-object pairs are strictly matched with each other. For example, HICO-DET does not have the object-intreaction pair "Throw-Spoon" as this is not a usual action but in practice it can happen and models trained on HICO-Det scheme will miss such pairings.

Metrics: For V-COCO, following [9] we evaluate our performance on two types of average precision(AP) metrics: Scenario 1 and Scenario 2. During AP calculation in both metrics, a prediction for a human-object pair is considered correct (1) if the human and object bounding boxes have an IoU greater than 0.5 with the ground-truth boxes and (2) the interaction class label of the prediction for the pair is correct. For the cases when there is no object(human only), in Scenario 1 a prediction is correct if the corresponding bounding box for the object is empty and in Scenario 2 the bounding box of the object is not considered. This makes Scenario 1 much harsher than Scenario 2 [9].

In HICO-Det, following [42], for each of the HOI class Average Precision (AP) is calculated by considering a prediction for a human-object pair is correct if the HOI label is correct along with the human and object bounding boxes have an IoU (intersection over union) greater than 0.5. This setting is termed as "Default" setting where mean AP is reported over three classes of samples: 1) considering all 600 categories (**Full**) 2) considering all 138 categories with less than 10 training samples (**Rare**) 3)2) considering all 462 categories with more than or equal 10 training samples (**Non-Rare**) [42].

5.5.2 Implementation Details

For the backbone feature extractor network, Resnet-152 [97] is used. We extract the input feature map before the last residual block of Resnet-152. This serves as the input to the rest of the network. We extract 10×10 feature maps for all the humans and objects from the input feature map by region of interest pooling [35]. Extracted RoIs and input feature map(context) pass through a residual block followed by a global average pooling similar to [99]. After these steps, we obtain three feature vectors of size $R = 1024$ for human, object and context. These are fed to the rest of the network. For the spatial attention branch we have used $64 \times 64 \times 2$ binary inputs. Before the element wise multiplication with the attention vector in the spatial attention branch, we project all our input feature vectors to a $D = 512$ dimensional space followed by a ReLU(Rectified Linear Unit). In our final classification layer for all the branches, we have one linear layer.

For training the network, we utilize off-the-shelf Faster-RCNN [34] to generate human and object bounding boxes. We have filtered the detected bounding boxes by setting 0.6 confidence threshold for human bounding boxes and 0.3 for object bounding boxes. The threshold values are chosen experimentally. Following [100] we did not fine tune the

backbone CNN Resnet-152 [97] and Faster-RCNN during our training process. Faster-RCNN was trained on the COCO [3] training set and did not see any image from V-COCO testing sets. Unlike previous works[99, 98], we do not use ground truth boxes during training as object proposals. As our object detector is robust, we directly use the bounding boxes generated from the detector which generates sufficient amount of positive and negative boxes.

Initially, we have trained the model on the training set of V-COCO while validating with the validation set. Then we train the model in both training and validation set like [100]. Our initial learning rate is set to 0.01 with a batch size of 8. As optimizer, Stochastic Gradient Descent(SGD) have been used with a weight decay of 0.0001 and a momentum of 0.9. To reduce the training time we have increased our learning rate to 0.01 for all the layers except for the spatial attention branch between epoch 9 to epoch 21. We trained the whole model for 50 epochs.

During inference, we multiply all the prediction outputs from the different branches of our network as in 5.14. Additionally, we multiply the final prediction output with the detection confidences of the human and object from the object deetctor. To differentiate between high and low quality detection scores we have adopted Low grade Instance Suppressive Function (LIS) [98]. We additionally remove the incompatible interaction-object pairs by using a post processing similar to iCAN [99] (e.g. if the object is not phone then the interaction can not be talk on the phone).

While making inference most of the existing [99, 98, 100] models multiply all the outputs from different modules but these modules are optimized separately while training. Following [106] we have used a single cross entropy loss function for each action class to optimize the network. One thing to note is that as in Eq. 5.14, interaction proposal score is also multiplied in these predictions and included in predictions for every 29 classes. This allows the proposal score to quantify if there are interactions between the

Method	mAP(Sc 1)	mAP(Sc 2)
InteractNet [100]	40.0	47.98
Kolesnikov et al. [104]	41.0	-
GPNN [103]	44.0	-
iCAN [99]	45.3	52.4
Li et al. [98]	47.8	-
VSGNet	**51.76**	**57.03**

Table 5.1: Comparison of results in V-COCO [9] test set on Scenario 1 and Scenario 2. Our method outperforms the closest method by 8%. For actor only classes (no object), scenario 1 requires the model to detect it specifically as no object, whereas scenario 2 ignores if there is an object assigned to that prediction. Some of these methods did not provide results for scenario 2.

human-object pair regardless of the class of that interaction. Our experiments show that combining all the predictions and using a single loss function improves the performance.

5.5.3 Comparisons with the State of the Art in V-COCO

We compare our model's performance with five recent state of the art methods [100, 104, 103, 99, 98]. We report mean Average Precision (mAP) score in the settings provided by [9]. Table 5.1 shows that our method outperforms all the existing models and achieves an improvement of 4 mAP in scenario 1. We also reported our performance in scenario 2 which outperforms all the available existing methods which reported their results in that scenario.

The closest work to our results is Li et al. [98] which builds on top of iCAN [99] by adding an interaction proposal network and utilizing person poses. Addition of interaction proposal and person poses improve ~ 2 mAP on top of iCAN with a computational cost of calculating the poses for each human. Our model achieves better results without the pose extraction and can possibly improve another 5% if the pose features are added to our visual feature branch.

HOI Class	InteractNet [100]	iCAN [99]	VSGNet
hold-obj	26.38	29.06	**48.27**
sit-instr	19.88	26.04	**29.9**
ride-instr	55.23	61.9	**70.84**
look-obj	20.2	26.49	**42.78**
hit-instr	62.32	74.11	**76.08**
hit-obj	43.32	46.13	**48.6**
eat-obj	32.37	37.73	**38.3**
eat-instr	1.97	**8.26**	6.3
jump-instr	45.14	51.45	**52.66**
lay-instr	20.99	**22.4**	21.66
talk_on_phone	31.77	52.81	**62.23**
carry-obj	33.11	32.02	**39.09**
throw-obj	40.44	40.62	**45.12**
catch-obj	42.52	**47.61**	44.84
cut-instr	22.97	37.18	**46.78**
cut-obj	36.4	34.76	**36.58**
work_on_comp	57.26	56.29	**64.6**
ski-instr	36.47	41.69	**50.59**
surf-instr	65.59	77.15	**82.22**
skateboard-instr	75.51	79.35	**87.8**
drink-instr	33.81	32.19	**54.41**
kick-obj	69.44	66.89	**69.85**
read-obj	23.85	30.74	**42.83**
snowboard-instr	63.85	74.35	**79.9**
Average	40.0	45.3	**51.76**

Table 5.2: Performance comparison in per class AP compared to the existing methods using Scenario 1. Our proposed VSGNet model demonstrates superior performance in majority of the classes. We only compared to the methods which have reported the per class AP values. Here, obj refers to as object and instr refers to as instrument [9].

In Table 5.2 we report per-class performances compare with the existing methods which reported per-class APs. Our proposed VSGNet achieves better performance in majority of the classes compared to the other methods. Additionally, per-class performances show that some of the action classes perform badly due to the failure of object detectors (e.g. eat instruments which usually have small objects and commonly become occluded in the images). As our main task is to detect HOIs, we did not fine-tune the existing object detectors according to our needs which can also possibly handle these cases.

5.5.4 Comparisons with the State of the Art in HICO-Det

Model training and implementation details for HICO-Det mostly follow the details from the V-COCO implementation. However, following the issues mentioned in previous sections, we train the model on 117 unique interaction classes instead of the 600 action-object interactions. This allows the model to learn the dynamics of the action.

We compare our work with the state of the art methods. Table 5.3 shows the results compared to other methods and our model achieves the best results among the previous works, improving the state of the art by 2.8 mAP and 15%.

Method (mAP)	Full	Rare	Non-Rare
HO-RCNN [42]	7.81	5.37	8.54
InteractNet [100]	9.94	7.16	10.77
GPNN [103]	13.11	9.34	9.34
iCAN [99]	14.84	10.45	16.15
Li et al. [98]	17.03	13.42	18.11
VSGNet	**19.80**	**16.05**	**20.91**

Table 5.3: Comparison of results in HICO-DET [9] test set. VSGNet outperforms the closest method by 16%.

5.5.5 Ablation Studies

Analysis of Individual Branches: Our overall architecture consists of three main branches as explained in Sections 5.4.2, 5.4.3, 5.4.4. To evaluate how these branches are affecting our overall performance, we evaluate these branches individually in the V-COCO [9] test set. Our evaluation method and metrics are same as Table 5.1. We consider the base model as the Visual branch without the spatial attention or the graph convolutions. In this setting, interaction proposal score I_{ho} and the class probabilities P_{ho} are predicted from the visual features f_{ho}^{Vis} directly.

We have added the graph network and the spatial network with our base model individually to evaluate each of the branch's performance separately. The results can be found in Table 5.4. With the addition of each individual branches, model performance also improves gradually. Visual+Spatial branch achieves state of the art results by itself without the Graph branch. Addition of the graph branch adds additional 1.5 mAP and a total of 4mAP over the state of the art.

Branches	mAP(Sc 1)	mAP(Sc 2)
Visual (Base)	47.3	52.15
Visual+Graph	48.19	53.12
Visual+Spatial	50.33	55.32
Visual+Spatial+Graph(VSG)	51.76	57.03

Table 5.4: Analysis of the branches. Our base model consists of only the Visual branch. We add the graph branch and the spatial attention branch to this base model separately to analyze their performances. Individually, both branches improve the performance upon the base model. Visual+Spatial model beats the state of the art results and all three branches combined adds another 1.5 mAP.

An important detail is that the graph branch directly depends on the quality of the interaction proposal score i_{ho} as it is used to determine the edge interactions. Without the spatial attention, visual features generate inferior i_{ho} which affects the graph branch. This is the reason that addition of Graph to Visual branch only adds 0.9mAP whereas

addition of Graph to Visual+Spat makes a larger improvement and adds 1.5 mAP.

Spatial attention branch improves the result by 3 mAP when added to the visual branch. This demonstrates the importance of the spatial reasoning and refining the visual features. Graph and Spatial attention combined improves the performance by about 4.5 mAP over the base model.

Analysis of Backbone CNNs: Most of the existing works use versions of Resnet [97] models as their backbone CNN feature extractors for the HOI task [99, 98, 100]. In addition to all Resnet models(34, 50, 101, 152), we implement our model with various CNNs used in image analysis including VGG-19 [45], InceptionV3 [107], SqueezeNet [108] and models. Table 5.5 shows the results of VSGNet implemented with these various backbone CNNs and ResNet152 achieves the best performance and is used in our VSGNet model.

Branch	mAP (Scenario 1)
VGG-19[45]	48.37
InceptionV3[107]	49.39
SqueezeNet[108]	43.4
Resnet34[97]	50.88
Resnet50[97]	51.01
Resnet101[97]	50.01
Resnet152[97]	**51.76**

Table 5.5: Effects of the backbone CNN. VSGNet is implemented using various common backbone CNNs. Scenario 1) results are reported. Resnet-152 model with VSGNet achieves the best performance.

Qualitative Results: Figure 5.7 shows qualitative results and compares the VSGNet with the base model (visual only). The interaction prediction probabilities for the correct action is visualized. The images show the variance in object sizes, human sizes and different interaction classes. VSGNet performs better than the base model. Even in the cases when the object is not entirely visible (image 9) or the interaction is very subtle

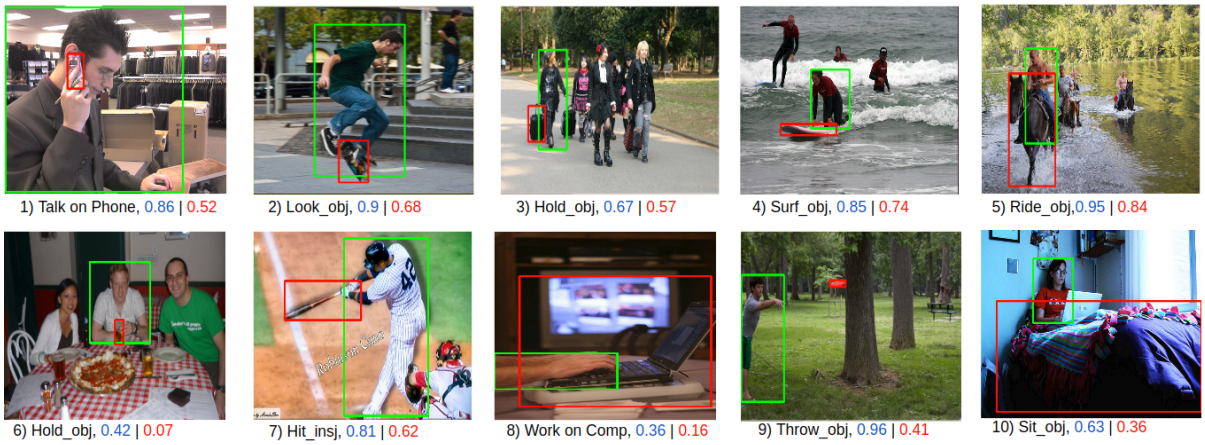


Figure 5.7: Qualitative results. Red values show the confidences for the base model (Visual only) and blue values are the results for the VSGNet. The prediction results and the correct action labels are shown for the human-object pair with the bounding boxes.

(image 2) VSGNet performs well and improves upon the base model.

5.6 Discussions

5.6.1 Comparisons with similar works

We compare our VSGNet with similar methods using spatial relations [42, 99, 98], attention models [99] and graph convolutions [103, 105].

There have been previous works which use spatial relation maps such [42, 99, 98]. These methods have either used the spatial relation maps directly for classification [42] or concatenated the spatial relation features to their visual features[99, 98]. Directly using them for classification ignores the visual features which in turn only learns relationship between the interaction label and spatial configuration. Concatenation of visual and spatial relations is also inferior to our method. As these are two completely different features defining separate things, concatenation does not enforce spatial configurations as much as an attention mechanism. In contrast, we use the spatial relations to extract

attention features which are then used to alter the visual features. This is more effective as it models the relations between the visual feature channels and spatial configuration due to the element-wise multiplication.

Attention models also have been used on HOI task. iCAN[99] model uses an attention model inspired from [49] and models the attention of the human or object region with the whole input scene individually. However, this approach does not consider the relation between the pairs and they only include the spatial configuration at the end. Our approach uses the spatial configuration directly to alter the visual features of the pairs which amplifies connected ones and dampens irrelevant ones at feature level.

Graph convolutions [103, 105] have been effective in various tasks. These tasks learn or use visual similarity as adjacency values between nodes and extract graph features. However, for our task, interaction proposal scores already defines the adjacencies between human-object node pairs and are used as edge intensities. This approach effectively extracts graph features by traversing relevant object nodes for the humans and relevant human nodes for objects.

5.6.2 Summary

In this chapter, we presented a novel human-object interaction detection model VSGNet which utilizes Visual, Spatial and Graph branches. VSGNet generates spatial attention features which alter the visual features and uses graph convolutions to model the interactions between pairs. The altered visual features generate interaction proposal scores which are used as edge intensities between human-object node pairs. We demonstrated with thorough experimentation that VSGNet improves the performance and outperforms the state-of-the-art.

Chapter 6

Person Detection in multi-modal setting

6.1 Introduction

Object detection is a frequently studied problem in the computer vision community [33, 34, 36]. Most of these recent works train and test their models on well annotated, higher resolution object detection datasets such as MS-COCO [3]. These models work well on many scenarios and can be used "out of the box" for inputs such as webcam recordings, mobile phone photos, online videos etc. However, in surveillance applications visual conditions are usually challenging and such object detection methods don't transfer well due to lower resolution, distance from the cameras and weather conditions. In such conditions, specialized models are needed to be trained for detecting objects of interest. For special surveillance cases, the visual models/inputs might not be sufficient for detection. In those cases, additional sensor modalities can be used to robustly achieve the detection task. Sensors such as lidars, microphones, seismic geophones could be used where the visual input is occluded or have low quality.

In security applications such as border surveillance, a very large area (1000s of miles) need to be watched over very long periods of time. In such applications, due to bandwidth and power limitations, the surveillance cameras need to be placed close to computational resources and on strategic locations, such as on top of the hills, where the field of view is very large and the cameras are far away from the targets. These visual conditions prevent robust detection from visual sensors only as the targets will have a size of couple of pixels (approximately 40 pixels). In such conditions, object detectors trained on traditional detection datasets fail and a model has to be trained for detecting humans on these types of inputs.

In addition to visual sensors, cheap seismic geophones can be utilized for detecting the humans walking around the sensor. Geophones are seismic sensors which can detect footsteps when an human is walking within 15 meters to the sensor. As these are cheap/low-power sensors, many of these sensors can be distributed in the large area of interest with each sensor monitoring its own range. Additionally, unlike videos cameras, seismic sensor records a temporal-only signal which means the bandwidth for communicating such signal and power requirements are very low and they don't have to be placed close to the computational resources and their signals can be transmitted there.

This work focuses on the surveillance task of detecting humans in a very large area by leveraging multi-modal sensor data. Low resolution visual videos are fused with a low-cost seismic sensor recordings to detect humans walking in the field. Low-resolution videos are recorded from a high resolution camera with a wide field of view (FOV) which is placed close to a power source but far away from the field with targets. This requires visual detection frameworks to search for small (few pixels) objects on a large field. Seismic sensors on the other hand can provide reliable information about their close surroundings and can easily be distributed on a large field. This allows the data from a seismic sensor to improve the detection of cameras in regions where camera view and

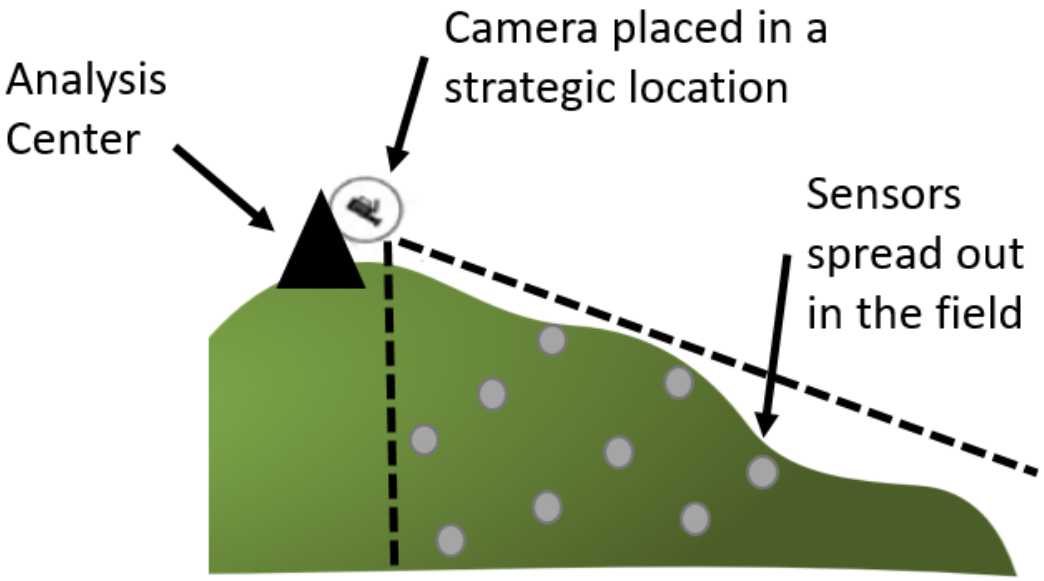


Figure 6.1: An example scenario. Cameras are placed in strategic locations such that they can monitor large areas. Analysis centers with computational resources are placed close to video cameras. Seismic sensors transmit their signals to the analysis center data/information fusion. Objects of interest include people, animals and vehicles.

sensor range intersects. Figure 6.1 shows a typical application setting for multimodal surveillance.

In this context, we propose a new order-preserving bilinear fusion model for person detection, leveraging pairwise interactions between convolutional features in a new way. We compare the proposed model with traditional fusion methods and single-modality models and show it's advantage over such methods. We demonstrate that sparse feature selection combined with bilinear fusion selects the optimal combinations of spatio-temporal features. We show that the proposed fusion method is differentiable and the final model is end-to-end trainable. The performance of our fusion model is tested in a multi-modal person detection dataset with syncronized seismic sensors and video cameras [109]. This work is published in [75].

6.2 Dataset Details

The data to be used in this setting is collected by the Army Research Lab (ARL). For an extended report on the details of the collection process, please refer to [109]. This section summarizes the details of the dataset and our pre-processing of the data for the person detection task.

The data is collected in a sensor field comprised of seismic, acoustic, magnetic sensors. Additionally, 5 video cameras are set up around the field to record the activity. Due to noisy nature of acoustic and magnetic sensors for this setting, we have only used the seismic sensors and video cameras for the person detection task. To keep the data collection homogenous for seismic sensors across the field, only the vertical axis of the 3-axis geophones are used in this task.

In the experiments, different people walk within the sensor field over multiple times while the cameras and seismic sensors are capturing the event. Seismic sensor and video camera locations are known by attached GPS trackers on each sensor. Additionally, each person walking in the field carries a GPS tracker which gives the precise location for the person and gives the ground truth for our dataset.

Each of the video cameras have very wide field of views. Cameras are able to monitor areas covered by many sensors. From each camera view, every sensor location can be approximated by the GPS locations of the sensors. Using the location information and the time readings, seismic sensors and cameras can be synchronized. Figure 6.2 shows a view from one of the cameras in the field. White dots in the image represent the locations for the seismic sensors and the red bounding box shows a person walking in the field.

Seismic sensors are able to detect a person's steps if the person is walking close-by. In our experiments, seismic signature of the footsteps become detectable when the person is within 15 meters to the seismic sensor. In order to co-register the visual input with the

Figure 6.2: Wide field of view seen from one of the video cameras. White dots indicate the sensor locations and the red bounding box indicate the person's location.

seismic sensor readings, we cropped a 100×100 pixels square region from video cameras centered on the seismic sensors location. This cropped region of interest monitors an approximately 15 meters range around the seismic sensor.

Figure 6.3 shows a pair of co-registered visual and seismic inputs over a 15 seconds interval. Seismic sensor is located at the center of the visual input. As the person approaches to the center in the visual input, magnitude of the seismic recording increases. Each one of the visualized "impulses" are generated from a footstep and the person is closest to the seismic sensor around 8 seconds mark where the seismic magnitude reaches the maximum amount.

Seismic sensors and their co-registered visual input cropped from the camera are paired up and labeled using the GPS trackers. We temporally extract signals over 1 second intervals and with a 50% overlap across samples. Whenever a person is within 15 meters to the seismic sensor, that sample is annotated as a positive sample. We assign 4 seismic sensors and their visual inputs as the testing set and remaining 12 sensors are used for training. Using this annotation scheme, 4646 positive and 26064 negative test samples and 16481 positive and 69483 negative training samples are annotated. Different

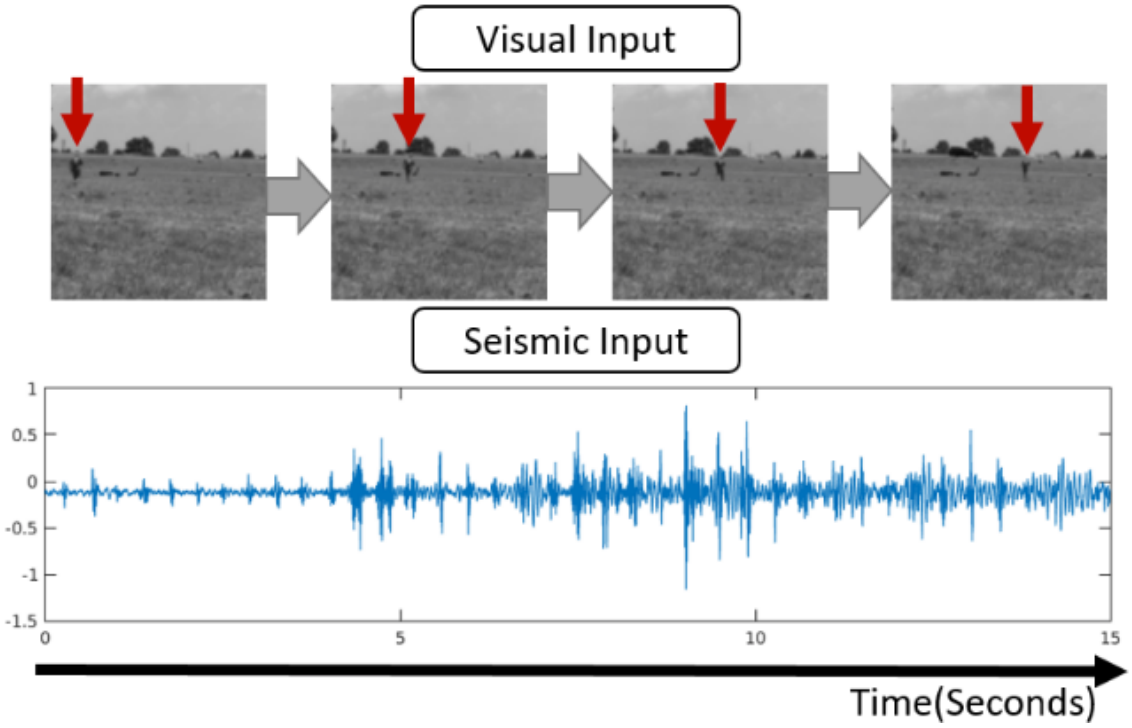


Figure 6.3: Co-registered visual and seismic inputs which are synchronized in time. A walking person is recorded near the sensor and the inputs are visualized over a 15 seconds interval.

combinations of seismic sensor and video camera pairs are considered as separate samples. Since the seismic sensor-camera pairs used for training and testing are comprised of different pairs, such data splitting process provides valid generalization results.

6.3 Related Work

In a surveillance setting, traditional detection methods for multimodal sensor data depend on hand-crafted features such as frequency domain analysis [110, 111], Symbolic Dynamic Filtering [112], and Cepstral features [113]. Damarla et al.[114] extracts and fuses hand-crafted features from multiple different modalities for person detection. Recently, with the advances of computational hardware and the increase of available data, feature learning has been integrated with classification to achieve end-to-end trainable

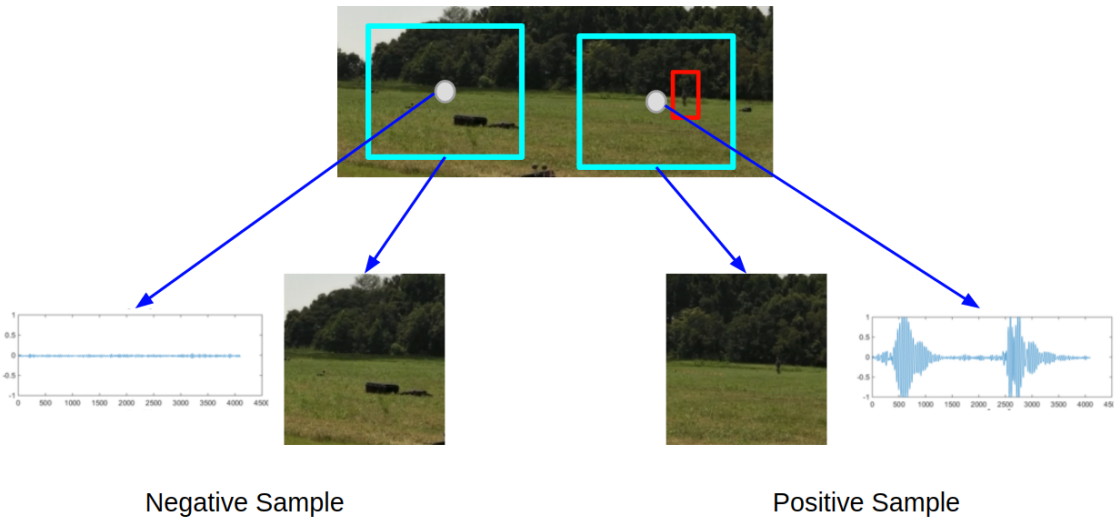


Figure 6.4: Generation of negative and positive samples. Given co-registered seismic and visual inputs, a sample is assigned a positive label if a person is within 15 meters to the sensor.

systems [23].

Ngiam et al. [115] analyzed the relations between different modalities in deep networks and showed that cross-modality feature learning can improve single modality performance. Riggan et al. [116] used Coupled AutoEncoders for cross-modal face recognition fusing visible and thermal imaging. [117, 118] achieved fusion by concatenating features from CNNs trained on RGB and depth images.

Fusing different features extracted from a single modality has been achieved using multiple different methods which are also applicable to multi-modal fusion. [43, 119] achieved late fusion between optical flow and RGB by averaging the confidence scores of single CNNs for video classification. Karpathy et al. [120] analyzed concatenating features from different time instances and trained fully connected layers to fuse information over time in a video.

Bilinear models were first analyzed by Tenenbaum and Freeman [121] to manipulate two factors from images, style and content. Recently bilinear models have achieved success in multiple tasks. Lin et al. [122] fused two convolutional neural networks

to obtain orderless descriptors and improved results in fine-grained visual recognition. Carreira et al. [123] used second order statistics of the local descriptors for semantic segmentation. RoyChowdhury et al. [124] used bilinear CNNs to improve results in face identification tasks. Gao et al. [125] improves the bilinear methods by developing a compact pooling method.

The main difference between recent bilinear methods [123, 125, 122, 124] and our method is that we use the outer product of vectors and obtain the pairwise feature interactions at each spatio-temporal indices. This is in contrast with these methods that use pooling methods over all indices and obtain an 'orderless' descriptor without preserving the order.

6.4 Technical Approach

The goal is to analyze the region of interest(ROI) and detect if a person walking in a field that is being monitored by a multi-modal sensor network data consisting of video cameras and seismic geophones. In this context, a ROI is any contiguous set of pixels and co-registered sensor data. Detection is defined on ROIs with corresponding camera and sensor pairs. We pose this as a binary classification problem for each ROI. Fig. 6.4 shows example ROIs located around the sensor locations which are known a priori. The inputs to our model are a single optical flow frame and its corresponding seismic signal for the same time interval.

In the following sections, we define the problem as a general multi-modal fusion problem and derive our fusion model by explaining each of the modules.

6.4.1 Problem Definition

Let X and Z be two sets of local descriptors extracted from two different modalities. Each descriptor $\boldsymbol{x}_{u_x,v_x,t_x} \in X$ represents the feature vector for the spatio-temporal voxel defined by the indices u_x, v_x, t_x, and similarly for the other modality $\boldsymbol{z}_{u_z,v_z,t_z} \in Z$. Let $\boldsymbol{x}$ and $\boldsymbol{z}$ be $N \times 1$ and $M \times 1$ dimensional feature vectors respectively.

Our goal is to develop a fusion algorithm $O = f(X, Z)$ such that spatio-temporal indices are preserved. For every spatio-temporal index from both modalities, we have the output feature vector:

$$\boldsymbol{o}_{u_x,v_x,t_x,u_z,v_z,t_z} = f(\boldsymbol{x}_{u_x,v_x,t_x}, \boldsymbol{z}_{u_z,v_z,t_z}) \tag{6.1}$$

where $\boldsymbol{o}_{u_x,v_x,t_x,u_z,v_z,t_z} \in O$ are the local descriptors of the output. If the input modalities are synchronized in time and space then we will have $(u_x, v_x, t_x) = (u_z, v_z, t_z) = (u, v, t)$. Indices from Eq. 6.1 simplifies into:

$$\boldsymbol{o}_{u,v,t} = f(\boldsymbol{x}_{u,v,t}, \boldsymbol{z}_{u,v,t}) \tag{6.2}$$

Furthermore, if we let modality X to be a spatial signal and modality Z to be a temporal signal. That gives $t_x = 1$ and $u_z = v_z = 1$ and simplifies the Eq. 6.1 into:

$$\boldsymbol{o}_{u,v,t} = f(\boldsymbol{x}_{u,v}, \boldsymbol{z}_{t}) \tag{6.3}$$

Eq. 6.3 defines the local descriptor which is the output of the fusion method. Note that in both Eq. 6.2 and Eq. 6.3, the calculation of $\boldsymbol{o}_{u,v,t}$, depends on the input values at indices u, v, t, which gives an ordered descriptor. Ordered descriptors allow us to exploit the relations between neighboring terms by using methods such as 3D convolutions. The goal is to detect targets using spatial images and temporal seismic sensor data, which

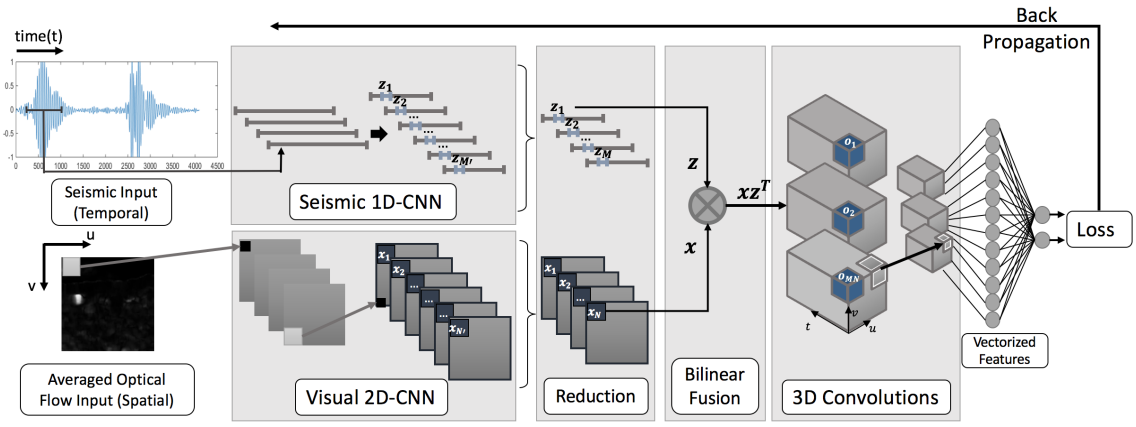


Figure 6.5: Order Preserving Bilinear model: Data from both modalities go through their respective CNN streams. Resulting features are compressed into lower dimensional vectors by sparse feature reduction and then fused by taking outer product at every spatio-temporal index. Since the order is preserved, 3D convolutions are leveraged. Since every module is differentiable, the whole model is trained end-to-end.

fits into the formulation in Eq. 6.3.

Our model is organized into four sub-components as shown in Fig. 6.5: **1)** input sensor signals are processed by dedicated CNNs for each modality (Section 6.4.2); **2)** at each spatial and temporal index, feature vectors are compressed in their depth dimension (Section 6.4.3); **3)** outer product is used in each spatio-temporal index to obtain the bilinear feature vector (Section 6.4.4); and **4)** 3D convolutions are used to leverage neighborhood relations of spatio-temporally ordered terms (Section 6.4.5).

6.4.2 CNN Features

In previous works, CNNs have been shown to extract useful features for variety of tasks on spatial [126], temporal [127] and spatio-temporal [43] modalities. CNNs extract local feature vectors at each spatio-temporal index (u, v, t). The size of the vector depends on the number of filters in the last convolutional layer, i.e., depth of the layer. For each

modality at each index u, v, t we have:

$$\boldsymbol{x'}_{u,v} = [x'_1, x'_2, ...x'_{N'}]^T, \tag{6.4}$$

$$\boldsymbol{z'}_t = [z'_1, z'_2, ...z'_{M'}]^T. \tag{6.5}$$

The prime ($'$) notations refer to the values before feature selection.

6.4.3 Sparse Feature Selection

The proposed fusion method, explained in Section 6.4.4, generates a high dimensional vector. Using high dimensional vectors are computationally challenging and can be prone to overfitting due to increased number of parameters. Within these large number of features, we want to prioritize which feature pairs are more useful (further discussed in Section 6.4.4). Therefore, we implement an efficient way to perform spatio-temporal feature selection by combining sparse 1×1 convolutions with bilinear fusion. Moreover, this method maintains spatio-temporal order. The goal is to compress the input vector to reduce the dimensions from Eq. 6.4. From here on, we generically use the term '**reduction**' to represent both feature selection and dimensionality reduction operations. We define our reduction function $r(.)$ as:

$$r(\boldsymbol{x'}_{u,v}) = \boldsymbol{x}_{u,v} = [x_1, x_2, ..., x_N]^T, \tag{6.6}$$

where $N < N'$ so that we obtain a more compact feature vector and we define the each reduced component x_i as the linear combinations of the original vector:

$$x_i = ReLU(\sum_{k=1}^{N'} w_{ik}^x x'_k) = max(0, \sum_{k=1}^{N'} w_{ik}^x x'_k), \tag{6.7}$$

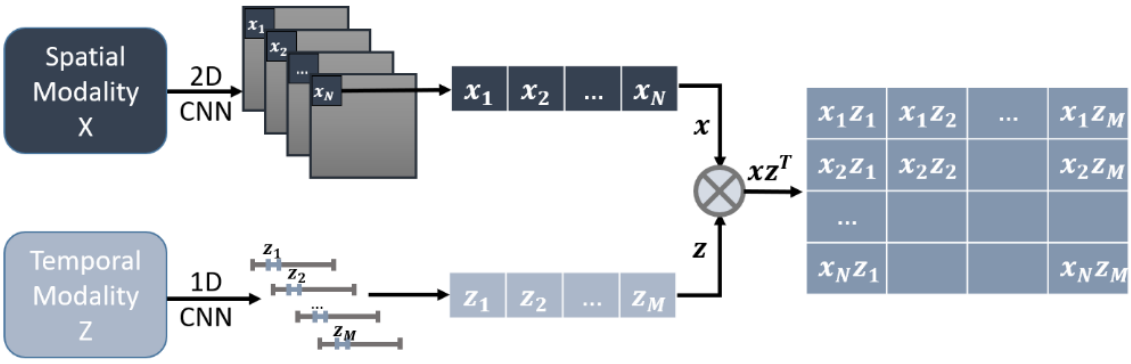


Figure 6.6: Bilinear CNN model for multi-modal fusion. Feature vectors at each spatial and temporal index is extracted using dedicated CNN streams for each modality. Outer product over feature vectors extract the second order pairwise relations matrix between modalities.

where weights w_{ik}^{x} are learned over the training and the norm of the weights are regularized using $L1$ normalization. Compared to $L2$ normalization or without normalization, $L1$ normalization generates a more sparse set of weights which forces the network to 'choose' the features that will be included in the summation. By $L1$ regularization, the weights $|w_{ik}^{x}|$ are mostly close to zero except a few weights that are multiplying essential set of features x'_k. This is similar to LASSO [128, 129] and provides a feature selection operation. Similarly for the second modality, reducing the vector from Eq. 6.5:

$$r(\boldsymbol{z}'_t) = \boldsymbol{z}_t = [z_1, z_2, ...z_M]^T, \tag{6.8}$$

$$z_i = ReLU(\sum_{k=1}^{M'} w_{ik}^{z} z'_k) = max(0, \sum_{k=1}^{M'} w_{ik}^{z} z'_k). \tag{6.9}$$

6.4.4 Order Preserving Bilinear Fusion

Reduced CNN features (Eq. 6.6 and Eq. 6.8) are fed into the fusion layer. At each spatial and temporal index, local feature vectors from both modalities are fused by taking the outer product. The fusion function at each spatio-temporal index $u \in U, v \in V, t \in T$

can be written as:

$$\boldsymbol{o}_{u,v,t} = f(\boldsymbol{x}_{u,v}, \boldsymbol{z}_t) = vectorize(\boldsymbol{x}_{u,v}\boldsymbol{z}_t^T) \tag{6.10}$$

At each index, we have length N vector $\boldsymbol{x_{u,v}}$ and length M vector $\boldsymbol{z_t}$. Outer product between these feature vectors generate the $N \times M$ second order pairwise features matrix:

$$\boldsymbol{x}_{u,v}\boldsymbol{z}_t^T = \begin{bmatrix} x_1z_1 & x_1z_2 & ... & x_1z_M \\ x_2z_1 & x_2z_2 & ... & x_2z_M \\ \vdots & & \ddots & \vdots \\ x_Nz_1 & x_Nz_2 & ... & x_Nz_M \end{bmatrix}. \tag{6.11}$$

We stack the rows together in lexicographical order, i.e., $N \times M$ dimensional matrix into an $MN \times 1$ vector. This gives the fused feature vector at each spatio-temporal index.

$$\begin{aligned} &\boldsymbol{o}_{u,v,t} = [o_1, o_2, ...o_{MN}]^T = \\ &\begin{bmatrix} x_1z_1 & ... & x_1z_M & ... & x_Nz_1 & ... & x_Nz_M \end{bmatrix}^T \end{aligned} \tag{6.12}$$

We repeat this operation for each spatial index u, v and temporal index t and obtain the fused second order feature vector at every combination of indices u, v, t.

Differentiability for Backpropagation

This fusion operation is differentiable for gradient operations and it is end-to-end trainable. In this section we show how the gradient can be backpropagated to each modality stream. Let L denote the cross-entropy loss function. Then by chain rule, we obtain:

$$\frac{\partial L}{\partial \boldsymbol{x}_{u,v}} = \frac{\partial L}{\partial \boldsymbol{o}_{u,v,t}} \frac{\partial \boldsymbol{o}_{u,v,t}}{\partial \boldsymbol{x}_{u,v}} = \frac{\partial L}{\partial \boldsymbol{o}_{u,v,t}} \begin{bmatrix} \frac{\partial o_1}{\partial x_1} & ... & \frac{\partial o_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial o_{MN}}{\partial x_1} & ... & \frac{\partial o_{MN}}{\partial x_N} \end{bmatrix} \tag{6.13}$$

where $\frac{\partial L}{\partial \boldsymbol{o}_{u,v,t}}$ can be calculated using chain rule of derivatives for layers between loss L and the outer product. Each partial derivative in the matrix can be written as:

$$\frac{\partial o_p}{\partial x_r} = \frac{\partial (x_s z_q)}{\partial x_r} \tag{6.14}$$

where $p = 1, .., MN$, $q = 1, .., M$, $r = 1, .., N$ and $s = 1, .., N$. For $s = r$, this simplifies into:

$$\frac{\partial o_p}{\partial x_r} = \frac{\partial (x_r z_q)}{\partial x_r} = z_q \tag{6.15}$$

For $s \neq r$, Eq. 6.14 becomes 0. Gradients before this layer can also be calculated by the regular CNN chain rule. $\frac{\partial L}{\partial \boldsymbol{z}_t}$ can be calculated similarly for the second modality.

Effects of Feature Selection

The outer product generates a high dimensional feature vector at each index $\boldsymbol{o}_{u,v,t}$. To handle the high dimensionality, we pool the convolutional features before the outer product operation by feature selection (Section 6.4.3). When $\sum_{k=1}^{N'} w_{ik}^x x_k' > 0$ and $\sum_{l=1}^{M'} w_{jl}^z z_l' > 0$, multiplying the terms from Eq. 6.7 and Eq. 6.9 yields for each $x_i z_j$ in Eq. 6.12:

$$x_i z_j = \sum_{k=1}^{N'} w_{ik}^x x_k' \times \sum_{l=1}^{M'} w_{jl}^z z_l' = \sum_{k=1}^{N'} \sum_{l=1}^{M'} w_{kl}^{xz} x_k' z_l' \tag{6.16}$$

where each $w_{kl}^{xz} = w_{ik}^x w_{jl}^z$. Otherwise, the $x_i z_j = 0$. This shows that output of reduced fusion operation is linear combinations of the second order interactions of the original feature vectors before the feature selection operation x_k', z_l'.

Weights of 1×1 convolutions w_{ik}^x, w_{jl}^z are trained with $L1$ regularization, hence they are individually sparse (Section 6.4.3). Therefore, this ensures that when multiplied,

the produced set of weights are also sparse and the product $w_{ik}^{x} w_{jl}^{z}$ is non-zero only if corresponding features k, l from each modality x'_k, z'_l are individually important for the task which is similar to sparse representations[130].

6.4.5 3D Convolutions

Since the outer product operation is repeated for every combination of the spatial (u, v) and temporal (t) indices, output of the fusion operation is a spatio-temporal feature tensor as shown in Fig. 6.5. This tensor allows us to use shared weights that stride across spatial and temporal dimensions, i.e., 3D convolutions, to reduce the total number of parameters and chances of overfitting by exploiting spatio-temporal correlations. In the tensor, at every spatio-temporal index, we have a feature vector of length MN which is the output of the outer product between length M vector $\boldsymbol{x}$ and length N vector $\boldsymbol{z}$. In the 3D convolutions, this dimension corresponds to the depth of input. The intuition behind keeping the spatio-temporal order is that certain activations in certain combinations of spatial and temporal indices complement each other. By having all the second order pairs as features at each index, we can find feature pairs that are sufficiently discriminative.

6.5 Implementation Details

In a surveillance setting, viewpoints and conditions vary for cameras and sensors, and surroundings can change the detected signature of the seismic sensors. To take this into account and to make the model generalizable, we split the data such that camera views (angle, background) and seismic sensors that are used in test set are different than the ones in training set. Each person in the field wears a GPS sensor. Using the location information we label the samples as positive when a person is within 15 meters of a seismic sensor. This results in 69483 negative and 16481 positive samples in training set

and 26064 negative and 6440 positive samples in test set.

From the visual input, a 100×100 region that is centered at a seismic sensor location (known a priori) is cropped from each camera frame. From seismic signals we extract our data points as 1 second intervals with 50% overlap. For the video data, we compute optical flow (OF) across the frames representing the same 1 second interval with the seismic signal. We choose to extract OF instead of raw RGB frames as the task is to detect a walking person. This simplifies the visual input and focuses the model on classifying the motion in the scene.

Magnitudes of these OF frames are averaged and used as the input to the proposed method. By averaging OF frames the spatio-temporal modality video is compressed into a spatial representation that encodes the temporal motion information. This approach reduces the noise levels in the optical flow as irrelevant changes across consecutive frames will be filtered out by averaging and it simplifies the computational requirements. This approach is further investigated and compared to LSTMs in Section 6.6.6.

To measure the performance of our methods, we report the precision, recall and F1-score values for the positive class. Recall values measure the detection accuracy whereas Precision measures the rate of false positives. In a data as unbalanced as ours, reporting both recall and precision becomes important. Since the negative class has significantly more samples than the positive class, high accuracy in detecting negative samples might still mean high false positive rates. For example 90% accuracy in negative test samples still means $26064 \times 0.10 = 2606$ false positives which is 40% of the total number of positive samples.

All models are trained using TensorFlow [79] and optimized using ADAM optimizer[77].

6.5.1 Single Modality CNNs

For extracting useful features from both seismic and visual data, we independently train modality specific CNNs for the detection task and analyze their performances.

Since there are no similar works using seismic sensors to be used for transfer learning, a randomly initialized 1-dimensional (temporal) CNN is trained for the seismic modality. This model is comprised of 6 Convolutional layers with a maxpool layer after every 2 layer and 2 fully connected layers for classification.

For the visual modality, we leverage the Inception V3 network architecture explained in [86] and initialize the network with weights that are pretrained for ImageNet [2]. Since this network is trained on RGB images and trained to detect ImageNet-specific features, we use earlier layers instead of the full architecture. Earlier layers in a CNN extract basic features such as edges, corners and these features are more generalizable. In [131] the authors quantified the generality and specificity of the layers and showed that the earlier layers are more generalizable. In [132] an OF CNN for action recognition is initialized using weights from a model trained for ImageNet. In our case, for the OF CNN we use the first five convolutional layers from Inception V3 model and initialize the weights from a ImageNet trained model.

6.5.2 Order Preserving Bilinear Fusion

The proposed approach (Fig. 6.5) consists of two dedicated streams of CNNs for each modality (Section 6.4.2), their corresponding sparse feature selection layers (Section 6.4.3), outer product between outputs of the two streams at each spatio-temporal index to preserve the order (Section 6.6.2), 3D convolutions (Section 6.4.5) and a final fully connected layer for classification. We refer this model as Order Preserving (OP) Bilinear Model.

Architectures used for the modality dedicated CNN streams are the same architectures as the single modality models defined in previous section. This allows us to initialize the model weights with pretrained weights from single modality models. Each CNN stream is followed by sparse feature selection and the fusion is achieved by order preserving outer product operation. Since the proposed outer product fusion is differentiable, as shown in Section 6.4.4, the whole model is fine-tuned in an end-to-end fashion.

6.6 Experiments and Results

In the following sections, we conduct a series of experiments to analyze the performance of each module in our method. First, we report experiments on the single modality CNNs and analyze the effects of dimensionality reduction. Then, we demonstrate the superior performance of the proposed bilinear fusion method compared to single modality models and alternative fusion methods. Furthermore, we compare the order-preserving methods that exploit 3D convolutions with their fully connected counterparts. Finally, we compare our visual approach with a LSTM approach.

6.6.1 Impact of Sparse Feature Reduction

For each modality, two different models are trained. Initial models use convolutional layers followed by fully connected layers. These models are labeled as 'Seismic' and 'Visual' in the tables. Additionally, we train models with the sparse feature selection method explained in Section 6.4.3. We add the feature selection layer between convolutional and fully connected layers. These models are labeled as 'Seismic Reduced' and 'Visual Reduced' in the tables. These models demonstrate the effect of the proposed feature reduction method compared to their regular versions

Table 6.1 shows that sparse feature selection (reduced models) from Section 6.4.3

provide a slight trade-off in performance for computation efficiency for computing bilinear features. In the Visual CNN, the reduction in number of parameters are significant with this reduction method.

Model	Recall	Precision	F1-Score
Seismic	0.90	0.87	0.89
Seismic Reduced	0.89	0.86	0.88
Visual	0.82	0.89	0.86
Visual Reduced	0.78	0.89	0.83
OP-Bilinear Fusion	**0.97**	**0.96**	**0.96**

Table 6.1: Precision, Recall and F1-Score values for single modality models and the proposed fusion method.

6.6.2 Fusion Compared to Single Modalities

Table 6.1 compares the proposed fusion method against single modality models. This table shows that the proposed method achieves modality fusion efficiently and provides the best performance in terms of accuracy (Recall) and false positive rate (Precision) compared to single modality models. Fig. 6.8 compares the method with other select models by plotting Precision-Recall curves. This plot demonstrates that our model is the best performing classifier since OP-Bilinear curve achieves the best Precision-Recall trade-off at every point.

Fig. 6.7 shows 3 sets of data samples. The first set shows the cases where both Visual and Seismic models fail but the fusion model correctly detects the target. In both samples, OF captures a weak motion and seismic sensor captures noise-like signals, but the fusion method detects the person nevertheless. The second set shows the samples where Visual model fail but Seismic and OP-Bilinear models correctly detects the target. Similarly, the third set shows the samples where Seismic model fails but Visual and OP-Bilinear model detects the target. This demonstrates that the fusion model achieves

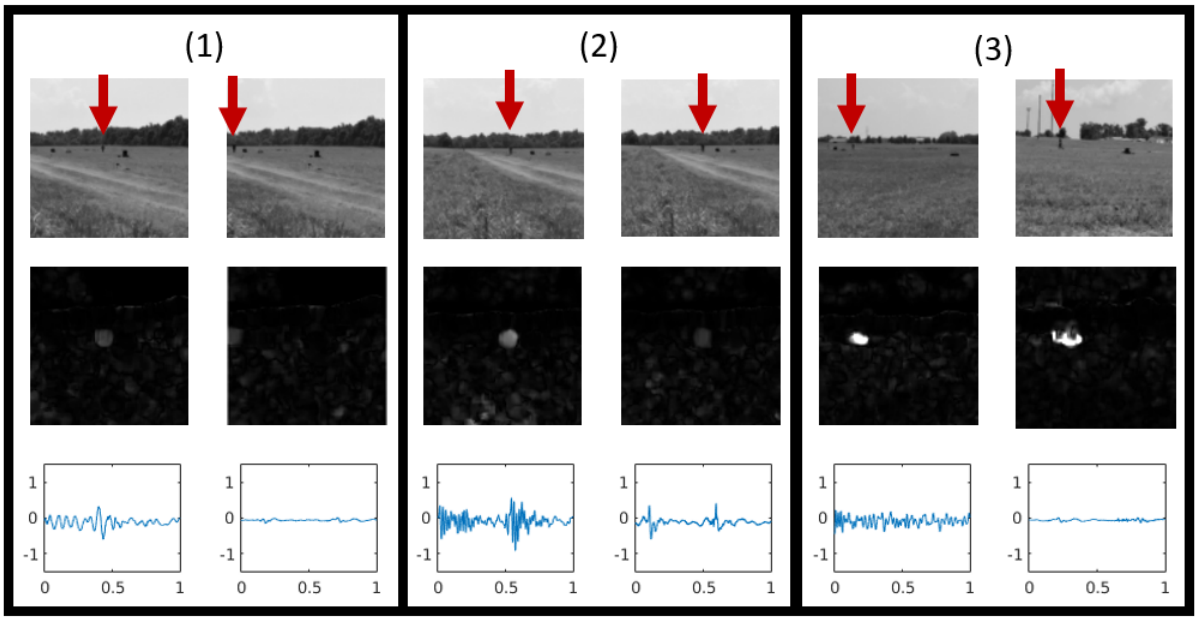


Figure 6.7: Examples of correct detections from the OP-Bilinear Model where single modality models fail. Red arrows indicate the targets. In (1) both Visual and Seismic Model fail to detect, (2) Visual model fails while Seismic model correctly detects the target and (3) Seismic model fails while Visual model correctly detects the target.

robust detection even when the input from a single sensor deteriorates.

We further compare the fusion model to the single modality models. As the distance between the target and the sensors increase, the performance deteriorates. Table 6.2 demonstrates that the proposed OP-Bilinear Fusion model is more robust to distance. The fusion model can effectively incorporate the information from the complementary modality when one modality degrades with range.

6.6.3 Effects of Initialization

In Section 6.4.4, we have derived the gradient for the proposed outer product operation. Since the gradient exists, the whole model is end-to-end trainable. In the previous section, we showed the results of the proposed method by initializing the model with single modality CNN model weights and fine-tuning the whole model. To investigate end-to-end

Distances From **Cameras**(meters)	50-80	80-110	110-140
Visual Reduced	0.96	0.93	0.74
OP-Bilinear Fusion	**0.98**	**0.96**	**0.95**
Distances From **Sensors**(meters)	0-5	5-10	10-15
Seismic Reduced	0.96	0.93	0.80
OP-Bilinear Fusion	**0.99**	**0.97**	**0.93**

Table 6.2: Recall rates for different distances from the cameras and seismic sensors. Even though the performance of OP-Bilinear model also decreases with range, the change is not as significant since it incorporates the information from the complementary modality.

training, we train a model using the same architecture, except the filter weights for the model are randomly initialized. Table 6.3 compares the performance of the end-to-end trained network with the model that is fine-tuned on pre-trained weights. This shows that pre-training achieves a slightly better performance than random initialization.

Model	Recall	Precision	F1-Score
End-to-End OP-Bilinear	0.95	0.95	0.95
OP-Bilinear Fusion	**0.97**	**0.96**	**0.96**

Table 6.3: Precision, Recall and F1-Score values for different initialization methods.

6.6.4 Comparisons with Fusion Methods

We compare our proposed OP-Bilinear Model with multiple late fusion approaches, feature concatenation approaches and state of the art Orderless Bilinear methods.

Average Fusion: We compare our results with a simple confidence score averaging late fusion method. This is a widely used method due to its simplicity [120, 43, 119]. In this method, we take the confidence scores from individually trained models 'Seismic' and 'Visual' from Section 6.5.1 and average them to get the final score for each datapoint. Results are labeled as 'Average Fusion' in Table 6.4.

Dempster Shafer Fusion: We compare our results with a more sophisticated late fusion method, Demster Shafer theory [133]. This theory is a framework for reasoning with uncertainty and generally applicable to sensor fusion models. We implement this model similar to [134]. We assume more uncertainty for visual modality than seismic modality, e.g. 35% versus 15%, due to noise and resolution. The results of this framework are shown in Table 6.4 with label 'Demster Shafer Fusion'.

Compared with these late fusion methods, our proposed fusion method is able to model the relations between the modalities and achieve better performance. Table 6.4 demonstrates that the proposed OP-Bilinear fusion model achieves higher detection rate (Recall) with lower false positive rate (Precision).

Concatenation-Fully Connected: Many multi-modal fusion [117, 118, 135] and feature fusion [120] methods concatenate the feature vectors from CNNs and classify the results using fully connected layer. This simple stacking of feature vectors compresses the spatial or temporal order since the features at every index are stacked into a single vector. Note that such operation does not exploit correlations in the spatial or temporal order. The output of this fusion can be expressed as:

$$\begin{aligned} \boldsymbol{o}_{u,v,t} &= [o_1, o_2, ...o_{M+N}]^T = \\ &\begin{bmatrix} x_1 & ... & x_N & z_1 & ... & z_M \end{bmatrix}^T \end{aligned} \tag{6.17}$$

and vectors at each spatial and temporal indices are also stacked into a vector as:

$$\begin{bmatrix} \boldsymbol{o}_{1,1,1} & ... & \boldsymbol{o}_{u,v,t} & ... & \boldsymbol{o}_{U,V,T} \end{bmatrix}^T \tag{6.18}$$

Results of this model are provided in Table 6.4 and Fig. 6.8 under the label 'Concatenation-FC'. The results show that the OP-Bilinear method achieves better performance than the Concatenation model by extracting bilinear features and preserving order.

Model	Recall	Precision	F1-Score
Average Fusion [43, 120]	0.90	0.92	0.91
Dempster Shafer Fusion [134]	0.93	0.95	0.94
Concatenation-FC [118, 120]	0.91	0.89	0.90
OP-Concatenation	0.93	0.90	0.91
Orderless Bilinear [122]	0.87	0.90	0.88
OP-Bilinear Fusion	**0.97**	**0.96**	**0.96**

Table 6.4: Precision, Recall and F1-Score values for different fusion methods and proposed method. Cited papers use similar (multi-modal or feature) fusion methods to our experimentation models.

Orderless Bilinear Descriptor: Bilinear pooling methods [122, 124, 123, 125] use sum pooling over spatial indices to pool the second order feature tensor into an orderless feature representation. Inspired by this idea, we sum the output of the outer product operation $\boldsymbol{x}_{u,v}\boldsymbol{z}_t^T$ from every spatial and temporal indices.

$$\sum_{u,v,t} \boldsymbol{x}_{u,v}\boldsymbol{z}_t^T = \begin{bmatrix} x_1z_1 & x_1z_2 & \dots & x_1z_M \\ x_2z_1 & x_2z_2 & \dots & x_2z_M \\ \vdots & & \ddots & \vdots \\ x_Nz_1 & x_Nz_2 & \dots & x_Nz_M \end{bmatrix} \tag{6.19}$$

Results of these fusion models can be seen in Table 6.4 and Fig. 6.8. The results demonstrate that the proposed method achieves the highest recall and precision rate among alternative fusion methods. Additionally, we observe that Orderless Bilinear model performs worse than the Concatenation. This is due to the summation approach over all the spatio-temporal indices in the former model loses the information instead of achieving fusion.

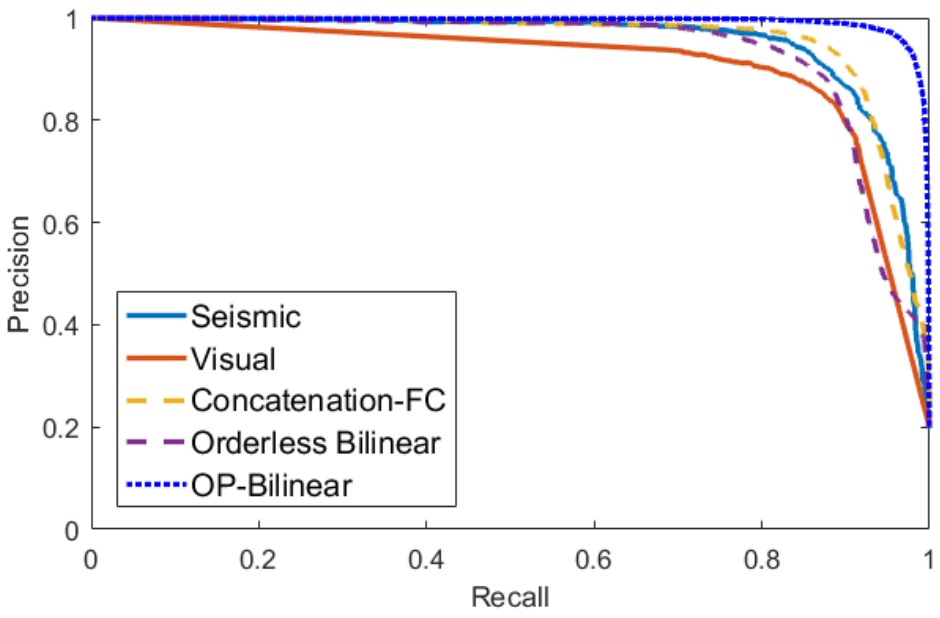


Figure 6.8: Precision-Recall curves show that OP-Bilinear Fusion achieves the best detection rate and fewest false positives.

6.6.5 Impact of 3D Convolutions

In this section we investigate the merits of 3D convolutions. Since the model is order preserving (OP), output of the fusion model is a spatio-temporal tensor. This tensor allows us to leverage 3D convolutions to reduce the total number of parameters and chances of overfitting by exploiting spatio-temporal correlations. We demonstrate this by comparing OP models that exploit 3D convolutions with corresponding fully connected models on two different fusion approaches, i.e., concatenation and bilinear feature descriptors. Table 6.5 demonstrates that models that preserve order achieve superior performance in both fusion approaches.

Order Preserving Concatenation: In this model, we adjust our order preserving approach to concatenation methods. We concatenate the features from each modality at every spatio-temporal index as in Eq. 6.17. However, instead of stacking the vectors

Model	Recall	Precision	F1-Score
Concatenation-FC	0.91	0.89	0.90
OP-Concatenation	0.93	0.90	0.91
Bilinear-FC	0.95	0.75	0.85
OP-Bilinear Fusion	**0.97**	**0.96**	**0.96**

Table 6.5: Precision, Recall and F1-Score values for Order Preserving (OP) fusion methods and their fully connected orderless variants. OP methods exploit 3D convolutions, other methods do not.

further (as in Eq. 6.18), we use these concatenated vectors as spatio-temporal local descriptors with $M + N$ length feature vector $\boldsymbol{o}_{u,v,t}$ at each index (u, v, t). Since the spatio-temporal order of descriptors is preserved this allows us to use 3D convolutions to exploit correlations. Tables 6.4, 6.5 show the results of this model under the label 'OP-Concatenation' and demonstrates that order preserving concatenation performs better than simple concatenation.

Bilinear-Fully Connected: In this model, we replace the 3D convolutions from the model in Section 6.6.2 with fully connected layers and fine-tune the network similarly with pre-trained CNN weights. This effectively removes the weight sharing of 3D convolutions, which removes the order-preserving aspect of the model and makes the model prone to overfitting.

Table 6.5 demonstrates the improvement in performance with preserving order on Bilinear Feature descriptors. OP-Bilinear model results with significantly fewer false positive rates, i.e, much higher precision compared to fully-connected method.

6.6.6 Averaging OF and LSTM Comparison

Our visual input is the magnitudes of OF vectors averaged over a time interval. Extracting OF from low-resolution cameras generate noisy inputs. Additionally, for this application, location and existence of the motion is as important as the evolution of

the motion. Spatial location of the motion captured among subsequent frames does not change drastically and averaging over a short time interval allows OF magnitudes to compress the motion captured while reducing the noise. This generates a low dimensional, compact feature description. However, a more complex and higher dimensional approach is capable of an incrementally better performance. Recurrent Neural Networks (RNNs) and Long-Short Term Memory (LSTMs) models have been shown to achieve good performance on variety of tasks [136, 137, 62]. We compare the performance of our averaged OF model with an LSTM model. In the LSTM model each input frame (OF Magnitude) goes through the convolutional part of the 'Visual' model from Section 6.5.1 and the outputs of the consecutive frames are fed into an LSTM cell similar to Activity Recognition model in [136]. Table 6.6 shows the performance of the LSTM compared to averaged OF visual model. This demonstrates that averaging reduces the dimensionality and has slightly better false positive rates compared to small improvement in detection performance of LSTMs. Additionally, for low-power strategic scenarios, processing every frame through a CNN model may not be possible(which is required in LSTM) whereas taking an average over a time interval and processing only this compact snapshot is more feasible.

Model	Recall	Precision	F1-Score
Visual	0.82	0.89	0.86
LSTM	0.86	0.86	0.86

Table 6.6: Comparison of the visual and LSTM model.

6.7 Conclusions

In this work, we introduced an OP-Bilinear Fusion method to jointly leverage sensor data and imagery. By conducting a series of experiments we analyzed the impact of

each module. We demonstrated that our feature selection algorithm makes the fusion method feasible by effectively reducing dimensionality with only a small tradeoff in single modality detection performance. We showed that our fusion model performs improves performance over models trained on single modalities and demonstrated that the fusion is beneficial. We compared the proposed fusion method with the traditional multi-modal and feature fusion methods and achieved better performance with the proposed method. Finally, we compared our approach of averaging OF frames to a more complicated LSTM approach and showed that by averaging multiple OF frames the sequence information is not lost and the model performs similarly.

The proposed method demonstrates the benefits of retaining the order when using a bilinear operator with video and seismic signals. However, the principle of preserving structural order with bilinear operators may be extended to any combinations of spatial or temporal data sources since the formulation in Eq. 6.1 is generic. Furthermore, by replacing the outer product operation with tensor product operation, method can be expanded for more than two modalities. where tensor product of three feature vectors can be expressed as $\boldsymbol{T} = \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}$ where $T_{i,j,k} = a_i b_j c_k$.

Chapter 7

Discussions

7.1 Discussions

In this thesis, we presented novel detection methods for analyzing videos and modeling actions in various settings. We have introduced and evaluated methods for human/object detection in challenging settings, atomic action detection in short video segments and human-object interaction detection in images. Our novel machine learning models combined relational structure between concepts as attention mechanisms and achieved state of the art results in their tasks. Combining object detectors/trackers with the action detection algorithms, our action detection system is one of the first systems in computer vision area to achieve real-time processing over a video streams to detect actions.

Chapter 2 provides an overview of the relevant tasks, datasets and methods and gives the required background for the following chapters. This chapter starts by describing the re-emergence of neural networks in image recognition tasks with the availability of large-scale datasets and GPUs for processing. This is followed by an overview of datasets and architectures for object detection tasks as this task is closely related to action detection. Finally an overview on previous action detection methods and large-scale datasets is

provided.

Even though object detection and action detection tasks are closely related, a fundamental difference is that action/interaction detection usually requires the analysis of the surrounding context. Chapter 3 addresses this fundamental difference by presenting a novel action detection architecture which models the surrounding context for each detected actor. This is achieved by utilizing an attention mechanism which considers the current actor and the surrounding context instead of looking at the actor individually. Additionally, this chapter showed that the proposed action detection system can transfer to various input settings (e.g.: surveillance videos, autonomous car videos etc.).

Previous detection methods usually follow an end-to-end approach where to whole task is learned from the data. However, when the task can be clearly broken down into multiple sub-tasks, modularity can be a better solution. Chapter 4 addressed this by presenting a modular approach to action detection. This chapter combined the atomic action detector with object detection and tracking modules and analyzed the framework. The framework achieved real-time performance while combining multiple, computationally expensive modules. The framework is tested on various application cases. Building on top of this framework, this chapter presented the complex activity detection pipeline by combining the atomic action detection with a rule based model analyzing spatial relations between humans and objects.

Due to challenges in the annotation process, video action datasets usually don't contain explicitly annotated interactions. Even though interaction labels exist, they do not explicitly annotate the interacted objects or humans. To study such interactions, Chapter 6 focuses on the Human-Object Interaction detection task on images. Building up from the rule-based spatial configurations, this chapter presented a novel neural network model which can utilize the spatial configurations of the human-object pairs and their structural connections. The presented model exploits spatial configurations between hu-

mans and objects to generate attention on each human-object pairs. These attention values quantify the types of relations between the pairs.

Detection of actions inherently depends on the detection of actors/humans. Whenever the human detection fails, further analysis of input videos become challenging. To study human detection in difficult settings, Chapter 7 focused on the surveillance task of detecting humans in a very large area by leveraging multi-modal sensor data. Low resolution visual videos are fused with a low-cost seismic sensor recordings to detect humans walking in the field. Low-resolution videos are recorded from a high resolution camera with a wide field of view (FOV) which is placed close to a power source but far away from the field with targets. This requires visual detection frameworks to search for small (few pixels) objects on a large field. Seismic sensors on the other hand can provide reliable information about their close surroundings and can easily be distributed on a large field. This allowed the data from a seismic sensor to improve the detection of cameras in regions where camera view and sensor range intersects.

7.2 Future Directions

Few-shot learning for action detection: The atomic action detection task has a significant imbalance between the action classes. Frequent classes have hundreds of thousands of instances where as some classes only have a hundred. Few-shot learning methods can help this task as these classes with fewer samples can be learned more effectively if the learning algorithm is designed for those cases.

Temporal Context for action detection: This thesis mostly analyzed spatial context of actions while modeling the temporal scale only on short intervals. However, longer temporal context have essential information about the actions as well. Models that can utilize longer temporal contexts have the potential to improve this task greatly.

Detection of complex activities: Detection of complex activities and analyzing longer temporal context is a challenging task. One major issue in this setting is the lack of datasets to effectively test the models. Collecting and annotating such datasets is challenging due to high cost, privacy issues and lack of interesting actions. Additionally, until recently, atomic action models were lacking in the community as well. Detection of atomic actions is the initial step for detecting complex activities and with such models including our presented model in this thesis, complex understanding of videos can be achieved in the future.

Bibliography

[1] Y. LeCun, *The mnist database of handwritten digits*, *http://yann. lecun. com/exdb/mnist/* (1998).

[2] 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** (2015), no. 3 211–252.

[3] 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*, pp. 740–755, Springer, 2014.

[4] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite*, in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[5] P. Zhu, L. Wen, X. Bian, L. Haibin, and Q. Hu, *Vision meets drones: A challenge*, *arXiv preprint arXiv:1804.07437* (2018).

[6] 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).

[7] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, *Hmdb: a large video database for human motion recognition*, in *2011 International Conference on Computer Vision*, pp. 2556–2563, IEEE, 2011.

[8] 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).

[9] S. Gupta and J. Malik, *Visual semantic role labeling*, *arXiv preprint arXiv:1505.04474* (2015).

[10] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, *et. al.*, *A large-scale benchmark dataset for event recognition in surveillance video*, in *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pp. 3153–3160, IEEE, 2011.

[11] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles, *Activitynet: A large-scale video benchmark for human activity understanding*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

[12] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, *et. al.*, *Ava: A video dataset of spatio-temporally localized atomic visual actions*, in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018.

[13] "Ava: A video dataset of atomic visual action."

[14] N. Wojke, A. Bewley, and D. Paulus, *Simple online and realtime tracking with a deep association metric*, in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[15] N. Wojke and A. Bewley, *Deep cosine metric learning for person re-identification*, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 748–756, IEEE, 2018.

[16] U. Army, *Us army field manual 21-60: Visual signals.*, 1987.

[17] P. BISCHOFF, "The world's most-surveilled cities."

[18] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, *Benchmark analysis of representative deep neural network architectures*, *IEEE Access* **6** (2018) 64270–64277.

[19] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et. al.*, *Speed/accuracy trade-offs for modern convolutional object detectors*, in *IEEE CVPR*, vol. 4, 2017.

[20] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, *arXiv preprint arXiv:1512.03385* (2015).

[21] N. Brew, *An overview of the effectiveness of closed circuit television (cctv) surveillance*. Parliamentary Library, 2005.

[22] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, *International journal of computer vision* **60** (2004), no. 2 91–110.

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

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation applied to handwritten zip code recognition*, *Neural computation* **1** (1989), no. 4 541–551.

[25] K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*, *Biological cybernetics* **36** (1980), no. 4 193–202.

[26] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, *Human pose estimation with iterative error feedback*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4733–4742, 2016.

[27] J. Dai, K. He, and J. Sun, *Instance-aware semantic segmentation via multi-task network cascades*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3150–3158, 2016.

[28] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, *Long-term recurrent convolutional networks for visual recognition and description*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

[29] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, *Decaf: A deep convolutional activation feature for generic visual recognition*, in *International conference on machine learning*, pp. 647–655, 2014.

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

[31] A. Karpathy and L. Fei-Fei, *Deep visual-semantic alignments for generating image descriptions*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[32] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, *Cnn features off-the-shelf: an astounding baseline for recognition*, in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

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

[35] R. Girshick, *Fast r-cnn object detection with caffe*, *Microsoft Research* (2015).

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *Ssd: Single shot multibox detector*, in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *Mobilenetv2: Inverted residuals and linear bottlenecks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[38] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, *Actions as space-time shapes*, in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1395–1402, IEEE, 2005.

[39] 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.

[40] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, *Towards understanding action recognition*, in *International Conf. on Computer Vision (ICCV)*, pp. 3192–3199, Dec., 2013.

[41] P. Mettes, J. C. Van Gemert, and C. G. Snoek, *Spot on: Action localization from pointly-supervised proposals*, in *European conference on computer vision*, pp. 437–453, Springer, 2016.

[42] Y.-W. Chao, Y. Liu, X. Liu, H. Zeng, and J. Deng, *Learning to detect human-object interactions*, in *2018 ieee winter conference on applications of computer vision (wacv)*, pp. 381–389, IEEE, 2018.

[43] K. Simonyan and A. Zisserman, *Two-stream convolutional networks for action recognition in videos*, in *Advances in neural information processing systems*, pp. 568–576, 2014.

[44] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, *Learning spatiotemporal features with 3d convolutional networks*, in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

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

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

[47] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, *arXiv preprint arXiv:1502.03167* (2015).

[48] X. Wang, R. Girshick, A. Gupta, and K. He, *Non-local neural networks*, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[50] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, *Simple online and realtime tracking*, in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, IEEE, 2016.

[51] O. Ulutan, S. Rallapalli, C. Torres, M. Srivatsa, and B. Manjunath, *Actor conditioned attention maps for video action detection*, in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2020.

[52] M. P. Eckstein, S. C. Mack, D. B. Liston, L. Bogush, R. Menzel, and R. J. Krauzlis, *Rethinking human visual attention: Spatial cueing effects and optimality of decisions by honeybees, monkeys and humans*, *Vision research* **85** (2013) 5–19.

[53] J. M. Henderson, C. L. Larson, and D. C. Zhu, *Full scenes produce more activation than close-up scenes and scene-diagnostic objects in parahippocampal and retrosplenial cortex: an fmri study*, *Brain and cognition* **66** (2008), no. 1 40–49.

[54] T. J. Preston, F. Guo, K. Das, B. Giesbrecht, and M. P. Eckstein, *Neural representations of contextual guidance in visual search of real-world scenes*, *Journal of Neuroscience* **33** (2013), no. 18 7846–7855.

[55] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, *Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search.*, *Psychological review* **113** (2006), no. 4 766.

[56] X. Peng and C. Schmid, *Multi-region two-stream r-cnn for action detection*, in *European Conference on Computer Vision*, pp. 744–759, Springer, 2016.

[57] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin, *Deep learning for detecting multiple space-time action tubes in videos*, *arXiv preprint arXiv:1608.01529* (2016).

[58] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin, *Online real-time multiple spatiotemporal action localisation and prediction.*, in *ICCV*, pp. 3657–3666, 2017.

[59] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, and C. Schmid, *Actor-centric relation network*, *arXiv preprint arXiv:1807.10982* (2018).

[60] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, *HMDB: a large video database for human motion recognition*, in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[61] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, *Hollywood in homes: Crowdsourcing data collection for activity understanding*, in *European Conference on Computer Vision*, pp. 510–526, Springer, 2016.

[62] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, *Beyond short snippets: Deep networks for video classification*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694–4702, 2015.

[63] R. Hou, C. Chen, and M. Shah, *Tube convolutional neural network (t-cnn) for action detection in videos*, in *IEEE international conference on computer vision*, 2017.

[64] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, *arXiv preprint arXiv:1409.0473* (2014).

[65] A. M. Rush, S. Chopra, and J. Weston, *A neural attention model for abstractive sentence summarization*, *arXiv preprint arXiv:1509.00685* (2015).

[66] A. Das, H. Agrawal, L. Zitnick, D. Parikh, and D. Batra, *Human attention in visual question answering: Do humans and deep networks look at the same regions?*, *Computer Vision and Image Understanding* **163** (2017) 90–100.

[67] J. Lu, J. Yang, D. Batra, and D. Parikh, *Hierarchical question-image co-attention for visual question answering*, in *Advances In Neural Information Processing Systems*, pp. 289–297, 2016.

[68] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, *Stacked attention networks for image question answering*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 21–29, 2016.

[69] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, *A simple neural network module for relational reasoning*, in *Advances in neural information processing systems*, pp. 4967–4976, 2017.

[70] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, *Relation networks for object detection*, in *Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2018.

[71] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan, *Attentive contexts for object detection*, *IEEE Transactions on Multimedia* **19** (2017), no. 5 944–954.

[72] P. Mettes and C. G. Snoek, *Spatial-aware object embeddings for zero-shot localization and classification of actions*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4443–4452, 2017.

[73] Y. Chao, Y. Liu, X. Liu, H. Zeng, and J. Deng, *Learning to detect human-object interactions*, *arXiv preprint*.

[74] G. Gkioxari, R. Girshick, P. Dollár, and K. He, *Detecting and recognizing human-object interactions*, *arXiv preprint arXiv:1704.07333* (2017).

[75] O. Ulutan, B. S. Riggan, N. M. Nasrabadi, and B. Manjunath, *An order preserving bilinear model for person detection in multi-modal data*, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1160–1169, IEEE, 2018.

[76] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, *Learning transferable architectures for scalable image recognition*, .

[77] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

[78] I. Loshchilov and F. Hutter, *Sgdr: Stochastic gradient descent with warm restarts*, *arXiv preprint arXiv:1608.03983* (2016).

[79] M. Abadi, A. Agarwal, *et. al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.

[80] B. Ghanem, J. C. Niebles, C. Snoek, F. C. Heilbron, H. Alwassel, V. Escorcia, R. Khrisna, S. Buch, and C. D. Dao, *The activitynet large-scale activity recognition challenge 2018 summary*, *arXiv preprint arXiv:1808.03766* (2018).

[81] T. Yao and X. Li, *Yh technologies at activitynet challenge 2018*, *arXiv preprint arXiv:1807.00686* (2018).

[82] J. Jiang, Y. Cao, L. Song, S. Z. Y. Li, Z. Xu, Q. Wu, C. Gan, C. Zhang, and G. Yu, *Human centric spatio-temporal action localization*, .

[83] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, *A better baseline for ava*, *arXiv preprint arXiv:1807.10066* (2018).

[84] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, *Action tubelet detector for spatio-temporal action localization*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4405–4413, 2017.

[85] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, *Learning deep features for discriminative localization*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

[86] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, *arXiv preprint arXiv:1512.00567* (2015).

[87] X. Liu, P. Ghosh, O. Ulutan, B. Manjunath, K. Chan, and R. Govindan, *Caesar: cross-camera complex activity recognition*, in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pp. 232–244, ACM, 2019.

[88] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, *Aggregated residual transformations for deep neural networks*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[89] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, *et. al.*, *Attend, infer, repeat: Fast scene understanding with generative models*, in *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

[90] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, *Semantic understanding of scenes through the ade20k dataset*, *International Journal of Computer Vision* **127** (2019), no. 3 302–321.

[91] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, *Unified perceptual parsing for scene understanding*, in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 418–434, 2018.

[92] H. Fan and J. Zhou, *Stacked latent attention for multimodal reasoning*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1072–1080, 2018.

[93] D. Yu, J. Fu, T. Mei, and Y. Rui, *Multi-level attention networks for visual question answering*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4709–4717, 2017.

[94] H. Nam, J.-W. Ha, and J. Kim, *Dual attention networks for multimodal reasoning and matching*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 299–307, 2017.

[95] D. Teney, P. Anderson, X. He, and A. van den Hengel, *Tips and tricks for visual question answering: Learnings from the 2017 challenge*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4223–4232, 2018.

[96] J. Wu, L. Wang, L. Wang, J. Guo, and G. Wu, *Learning actor relation graphs for group activity recognition*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9964–9974, 2019.

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

[98] Y.-L. Li, S. Zhou, X. Huang, L. Xu, Z. Ma, H.-S. Fang, Y. Wang, and C. Lu, *Transferable interactiveness knowledge for human-object interaction detection*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3585–3594, 2019.

[99] C. Gao, Y. Zou, and J.-B. Huang, *ican: Instance-centric attention network for human-object interaction detection*, in *British Machine Vision Conference*, 2018.

[100] G. Gkioxari, R. Girshick, P. Dollár, and K. He, *Detecting and recognizing human-object interactions*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8359–8367, 2018.

[101] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[102] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah, *The thumos challenge on action recognition for videos in the wild*, *Computer Vision and Image Understanding* **155** (2017) 1–23.

[103] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu, *Learning human-object interactions by graph parsing neural networks*, in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 401–417, 2018.

[104] A. Kolesnikov, A. Kuznetsova, C. Lampert, and V. Ferrari, *Detecting visual relationships using box attention*, in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[105] L. Li, Z. Gan, Y. Cheng, and J. Liu, *Relation-aware graph attention network for visual question answering*, *arXiv preprint arXiv:1903.12314* (2019).

[106] T. Gupta, A. Schwing, and D. Hoiem, *No-frills human-object interaction detection: Factorization, layout encodings, and training techniques*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9677–9685, 2019.

[107] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[108] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size*, *arXiv preprint arXiv:1602.07360* (2016).

[109] S. M. Nabritt, T. Damarla, and G. Chatters, *Personnel and vehicle data collection at aberdeen proving ground (apg) and its distribution for research*, tech. rep., Army Research Lab Adelphi, MD Sensors and Electron Devices Directorate, 2015.

[110] T. Damarla, A. Mehmood, and J. Sabatier, *Detection of people and animals using non-imaging sensors*, in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pp. 1–8, IEEE, 2011.

[111] J. M. Sabatier and A. E. Ekimov, *Range limitation for seismic footstep detection*, in *SPIE Defense and Security Symposium*, pp. 69630V–69630V, International Society for Optics and Photonics, 2008.

[112] S. Bahrampour, A. Ray, S. Sarkar, T. Damarla, and N. M. Nasrabadi, *Performance comparison of feature extraction algorithms for target detection and classification*, *Pattern Recognition Letters* **34** (2013), no. 16 2126–2134.

[113] N. H. Nguyen, N. M. Nasrabadi, and T. D. Tran, *Robust multi-sensor classification via joint sparse representation*, in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pp. 1–8, IEEE, 2011.

[114] R. Damarla and D. Ufford, *Personnel detection using ground sensors*, in *Defense and Security Symposium*, pp. 656205–656205, International Society for Optics and Photonics, 2007.

[115] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, *Multimodal deep learning*, in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[116] B. S. Riggan, C. Reale, and N. M. Nasrabadi, *Coupled auto-associative neural networks for heterogeneous face recognition*, *IEEE Access* **3** (2015) 1620–1632.

[117] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, *Multimodal deep learning for robust rgb-d object recognition*, in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 681–687, IEEE, 2015.

[118] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, *Convolutional-recursive deep learning for 3d object classification.*, in *NIPS*, vol. 3, p. 8, 2012.

[119] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, *Towards good practices for very deep two-stream convnets*, *arXiv preprint arXiv:1507.02159* (2015).

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

[121] J. B. Tenenbaum and W. T. Freeman, *Separating style and content with bilinear models*, *Neural computation* **12** (2000), no. 6 1247–1283.

[122] T.-Y. Lin, A. RoyChowdhury, and S. Maji, *Bilinear cnn models for fine-grained visual recognition*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1449–1457, 2015.

[123] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, *Semantic segmentation with second-order pooling*, in *European Conference on Computer Vision*, pp. 430–443, Springer, 2012.

[124] A. RoyChowdhury, T.-Y. Lin, S. Maji, and E. Learned-Miller, *Face identification with bilinear cnns*, *arXiv preprint arXiv:1506.01342* (2015).

[125] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, *Compact bilinear pooling*, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 2016.

[126] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 2016.

[127] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, *Convolutional neural networks for speech recognition*, *IEEE/ACM Transactions on audio, speech, and language processing* **22** (2014), no. 10 1533–1545.

[128] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68** (2006), no. 1 49–67.

[129] L. Meier, S. Van De Geer, and P. Bühlmann, *The group lasso for logistic regression*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70** (2008), no. 1 53–71.

[130] V. M. Patel and R. Chellappa, *Sparse representations, compressive sensing and dictionaries for pattern recognition*, in *Pattern Recognition (ACPR), 2011 First Asian Conference on*, pp. 325–329, IEEE, 2011.

[131] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, *How transferable are features in deep neural networks?*, in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), pp. 3320–3328. Curran Associates, Inc., 2014.

[132] 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 *European Conference on Computer Vision*, pp. 20–36, Springer, 2016.

[133] A. P. Dempster, *Upper and lower probabilities induced by a multivalued mapping*, *The annals of mathematical statistics* (1967) 325–339.

[134] K. Lee, B. S. Riggan, and S. S. Bhattacharyya, *An accumulative fusion architecture for discriminating people and vehicles using acoustic and seismic signals*, in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[135] J. Wagner, V. Fischer, M. Herman, and S. Behnke, *Multispectral pedestrian detection using deep fusion convolutional neural networks*, in *European Symp. on Artificial Neural Networks (ESANN)*, 2016.

[136] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, *Long-term recurrent convolutional networks for visual recognition and description*, in *CVPR*, 2015.

[137] N. Srivastava, E. Mansimov, and R. Salakhutdinov, *Unsupervised learning of video representations using lstms.*, in *ICML*, pp. 843–852, 2015.