Situate: An Agent-Based System for Situation Recognition

by
Max Henry Quinn

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

Doctor of Philosophy
in
Computer Science

Dissertation Committee:
Melanie Mitchell, Chair
Ameeta Agrawal
Dan Hammerstrom
Feng Liu

Portland State University
2021

Abstract

Computer vision and machine learning systems have improved significantly in recent years, largely based on the development of deep learning systems, leading to impressive performance on object detection tasks. Understanding the content of images is considerably more difficult. Even simple situations, such as "a handshake", "walking the dog", "a game of ping-pong," or "people waiting for a bus", present significant challenges. Each consists of common objects, but are not reliably detectable as a single entity nor through the simple co-occurrence of their parts.

In this dissertation, toward the goal of developing machine learning systems that demonstrate properties associated with understanding, I will describe a novel system for performing visual situation recognition. Given a description of a situation and a small labeled training set, the system, called Situate, learns object appearance models as well as a probabilistic model capturing the situation's expected spatial relationships. Given a new image, Situate uses its learned models and an array of agents to engage in an active search of its input to find the most consistent correspondence between the model of the situation and the content of the image. Each agent develops a possible correspondence between the model and the input, while Situate allocates computational resources to the agents such that promising solutions are developed early, but alternative correspondences are not ignored.

I will compare Situate to a more traditional computer vision approach that relies on the detection of constituent objects of a situation, as well as to a related image-retrieval system based on "scene graphs". I will evaluate each method on the situation recognition task and in the context of image retrieval. The results demonstrate the value of a feedback system between image content and a model of that content.

Acknowledgements

Completing this work would not have been possible without the support of my friends and family. In particular, I thank my partner Deb Healy, my parents Bill Quinn and Suzy Q, and my sister Amy. They have all been supportive and patient beyond reason.

I would also like to thank my advisor, Melanie Mitchell, for her support and for her ability to bring kind and generous students into her fold. In particular, Will Landecker, Mick Thomure, and Jordan Witte have all been dear friends and invaluable colleagues at every step.

Contents

1 Introduction **1**
- 1.1 Background . . . 4
- 1.2 Situate and situation recognition . . . 9
- 1.3 Walk-through of an agent evaluation . . . 14
- 1.4 Related work . . . 17
  - 1.4.1 Methods related to object recognition and localization . . . 18
  - 1.4.2 Tasks related to situation recognition . . . 25

2 Function of a Single Situate Agent **35**
- 2.1 Structures and procedures . . . 35
  - 2.1.1 Implementation and parameter setting . . . 38
- 2.2 Evaluation of the initial Situate agent . . . 47
  - 2.2.1 Sources of improvement . . . 52

3 Improving the Situate Agent **57**
- 3.1 Classifier improvements . . . 57
  - 3.1.1 IOU regression . . . 64
  - 3.1.2 Bounding box regression . . . 66
- 3.2 Support functions . . . 73
  - 3.2.1 Internal support . . . 77
  - 3.2.2 External support . . . 79
  - 3.2.3 Combined internal and external support . . . 88
- 3.3 Evaluation of the updated Situate agent . . . 90

4 Managing Multiple Situate Agents **95**
- 4.1 Multi-Agent Approach . . . 95
  - 4.1.1 Differences from MCTS . . . 103
- 4.2 Evaluation of grounding performance . . . 106

5 Applications and Additional Situations **108**
- 5.1 Situate for retrieval . . . 108
- 5.2 Additional Situations . . . 114

6 Discussion and Future Work **124**

A Additional Experiments **133**
- A.1 Full Situation Localization and Retrieval . . . 133

List of Figures

1.1 Several examples of the "dog-walking" situation. 9
1.2 Several examples of atypical instances of the "dog-walking" situation 13
1.3 The state of a Situate agent during evaluation 15
1.4 The state of a Situate agent during evaluation (continued) 16

2.1 Examples of bounding boxes that localize objects with varying quality 39
2.2 The relationship between classifier response and localization quality . 45
2.3 The distribution of *external support* scores 46
2.4 Localization quality for several situation recognition methods 49
2.5 Significance of differences in localization quality for several methods . 50
2.6 Localization quality for several lesioned versions of Situate 54
2.7 Significance of differences in localization quality for lesioned Situate . 56

3.1 Examples of low quality Workspaces generated by Situate agents . . . 59
3.2 Correlation between classifier output and ground truth IOU scores . . 61
3.3 The expected value of increases in IOU as a function of differences in SVM classification scores 63
3.4 The probability of an increase in IOU as a function of differences in SVM classification scores 64
3.5 The probability of an increase in IOU as a function of differences in estimated IOU 65
3.6 The expected value of increases in IOU as a function of differences in estimated IOU 65
3.7 The difference in IOU before and after bounding box regression using training data with varying ground truth IOU 68
3.8 The probability and expected value of changes in IOU using two versions of bounding box regression 69
3.9 The probability and expected value of changes in IOU using multiple bounding box regression models in combination 70
3.10 Object localization quality for SVM and IOU estimation versions of the Situate agent 72
3.11 Significance of the differences in localization quality between classification methods 72
3.12 Significance of the differences in localization quality between classification methods (continued) 73
3.13 Estimating odds ratios using the AUROC of the IOU estimator . . . 80
3.14 The relationship between predicted localization scores using contextual evidence and actual localization quality 87

3.15 The relationship between situation localization quality and the number of calls to the classifier for the original and updated Situate agent . . 91
3.16 The object localization quality for the two versions of the Situate agent and Faster-RCNN on the dog-walking situation 92
3.17 Examples of Workspaces with high confidence that did not effectively localize the objects of a situation that was present in the image . . . 93
3.18 Examples of Workspaces with low confidence of a situation that was present in the image 94

4.1 Characteristic errors made by Situate agents 97
4.2 Example of Situates multi-agent evaluation of an image 101
4.3 Object grounding quality comparison between Faster-RCNN, a single Situate agent, and the multi-agent implementation of Situate 107

5.1 ROC and PR curves for image retrieval of the dog-walking situation using Situate, Faster-RCNN and IRSG 111
5.2 ROC and PR curves for image retrieval of the dog-walking situation using Situate, Faster-RCNN and IRSG with challenging negative images 113
5.3 Images for which Faster-RCNN produced high situation scores for the ping-pong situation 116
5.4 Images for which Situate produced high situation scores for the ping-pong situation 117
5.5 Images for which Faster-RCNN produced high situation scores for the handshaking situation 118
5.6 Images for which Situate produced high situation scores for the handshaking situation 119
5.7 Localization quality for Situate and Faster-RCNN on the ping-pong situation 120
5.8 Localization quality for Situate and Faster-RCNN on the handshaking situation 120
5.9 ROC curves for image retrieval using Situate and Faster-RCNN on the ping-pong and handshaking situations 121
5.10 Failures to correctly ground the handshaking situation by Faster-RCNN and Situate 122
5.11 Failures to correctly ground the ping-pong situation by Faster-RCNN and Situate 123

A.1 Localization quality of the full situation bounding box for the dog-walking situation 134
A.2 Highest scoring instances of the full situation detected by a full-situation localization system 135
A.3 High scoring bounding boxes for negative instances of the dog-walking situation generated by the full situation localization system 136

A.4 Low scoring bounding boxes for positive instances of the dog-walking situation generated by the full situation localization system 137

List of Tables

3.1 A comparison of bounding box sampling methods, scoring methods, and the discriminative capacity of density values 83

4.1 Psuedocode for Situate's agent management method. 105

Chapter 1

Introduction

This dissertation addresses the problem of "visual situation recognition". This task requires a system to identify several objects in an image that have a particular relationship to one another. An illustrative example is "dog-walking", wherein an image contains a person walking a dog, a dog, a leash, and clutter, such as buildings, trees, cars, or other people. A correct solution to the problem consists of a binary indicator of whether or not an input contains the situation, and labels and sufficiently precise specifications of the image regions that contain the objects of the situation. An example of the situation recognition task can be found in Figure 1.1.

Despite significant strides in machine learning systems in recent years, situation recognition is an example of a problem that remains difficult, and is illustrative of several challenges that I anticipate will become increasingly significant as the field develops in coming years. My proposed work is meant to explore possible tools for addressing some of these anticipated challenges and to use the problem of visual situation recognition as a test bed.

- *Machine learning systems will need to provide evidence used to make decisions.* The broad adoption of machine learning systems has come quickly, but there remains a long standing concern over the opaqueness with which decisions are made and how to understand errors. For critical tasks, which range from control systems for vehicles to removing users from a social media platform, depending on the priorities of the user, understanding how decisions are made makes it

possible for users of machine learning systems to take responsibility for those decisions, as well as to identify the source of errors and to address them. Situation recognition emphasizes approaches that address this concern by requiring that the responses from a system provide supporting evidence for a decision by identifying constituent parts, including parts that are difficult to identify without the use of context and inference (such as the leash object in a dog-walking situation).

- *The decision tasks assigned to machine learning systems will become more complex.* Instead of identifying individual objects, the relationships between objects will become increasingly important. Current approaches to recognition rely on massive data sets on which to train, the existence of which cannot be relied upon when the object or situation of interest is defined by complex or rare interactions between objects. For problems that will be important moving forward, we may need to leverage understanding interactions rather than just observing instances during the learning phase. The task of situation recognition requires solutions that take relationships into account, as much of the clutter in an input may be made up of object types that can be relevant to the situation, but are not involved for the particular input. For example, Figure 1.1 shows examples of the dog-walking situation where the clutter includes other people that are not involved in the dog-walking situation.

- *Machine learning systems will need to provide richer responses.* As it stands, the outputs of classifiers are usually trained to provide binary responses, indicating inclusion or non-inclusion of an instance in a category. Outside of training, the actual responses are generally presented as probabilities, where values close to zero or one are easy to interpret, but middling values have an ambiguity. When a human responds that they are unsure, we generally expect that they

can provide additional information, either with respect to what was confusing or what additional information would make the decisions clearer. This lets us know how to proceed with whatever task required the decision. If a person can't read a sign, it might be because it's too far away, or it might be because it's in a language that the reader does not know. These situations are different, and likely require a different response. However, classifiers don't respond with explanatory information. There are many reasons for failed classification, such as significant noise in an input or an input being from outside of known categories.

The main contribution of this dissertation is the demonstration of a prototype system, called *Situate*, that is meant to solve problems like situation recognition, where relationships between objects matter, and responses must include a record of the evidence used to make a decision. Situate is a system designed to be extended such that it can integrate structured, prior knowledge with patterns learned from data, and will eventually provide conditional responses with caveats of the form "this is an instance of the query given the replacement of condition A with condition B". That is, it will be able to generalize from known situations to related, but previously unobserved, situations. For example, if Situate knows the situation "dog-walking" to consist of a dog, leash, and a person walking, but also can identify people riding bicycles, a possible response to a "dog-walking" situation query might include something like "yes, this is an instance of the query given the replacement of a person walking with a person riding a bike".

To build an understanding of an input, Situate utilizes a collection of agents; each representing a hypothesis regarding the content of the input and a series of resulting predictions. The agents' predictions can be evaluated using classification systems and the findings can be compared against expectations about the situation that is

contained in a graph and informed by labeled training data. Finally, the hypotheses can develop into a structure that functions as both a decision and an explanation. By utilizing multiple agents, Situate can compare multiple interpretations of an input and respond with the one that it is most confident in.

1.1 Background

Much of machine learning research and attention in the past several years has focused on "deep learning," roughly understood as the use of multi-layered artificial neural networks for a variety of discrimination tasks. The success of deep learning systems is somewhat surprising because these models are essentially the same as those proposed as early as the 1960s, and studied with fervor during the 1980s [23]. The difference between current models and their earlier counterparts is largely a) the ability to take advantage of GPUs for massively parallel floating point operations, b) the use of an activation function that makes the networks especially compatible with GPUs, c) the use of a stochastic training mechanism called "dropout," which trains a subset of the network's nodes during each iteration of a long training process, and d) the availability of massive data sets upon which to train. These changes are relatively simple (in terms of algorithms), but have led to a profound increase in effectiveness.

However, at their core, these networks are feed-forward functions that map from an input to highly constrained outputs. This structure applies a fixed set of linear pattern detection functions and simple non-linear activation functions that work well for recognition tasks, but struggle with what we would recognize as reasoning tasks. When considering inputs of variable length, neural networks can struggle as their topology defines how much of the input can be processed at one time. When the difficulty of input instances vary, neural network use their full architecture for every input. Furthermore, because feed-forward networks are trained using error back-

propagation, because the topology of the network needs to be able to process the most difficult of its likely inputs, and because back propagation needs more data as the network increases in complexity, very large data sets and training periods are the norm.

The limitations of feed-forward networks have not prevented researchers from utilizing deep networks in solving interesting reasoning problems. This is usually done by a) somewhat awkwardly adjusting the networks to encode more complex outputs, or b) using the deep network as a component in an assembly of systems. In both cases, the systems tend to be very specialized. A significant example of how deep networks are used in systems that need to make complex decisions is AlphaGo, the first artificial intelligence system to defeat a high-level human Go player. This system used deep learning, but not by mapping board states to moves likely to be made by a good player. Selecting moves in Go is particularly difficult because there are a large number of legal moves at any time, and players are able to consider many moves in the future while playing. Players are able to do this because the vast majority of moves are clearly not viable, so can be eliminated from their consideration; and frequently the game falls into a sequence where many turns are easily predicted before the board state again requires serious consideration.

A purely feed-forward neural network approach would require large collections of network nodes to consider possible moves for a particular board state, some large number of layers to make an evaluation of that board state, and then the structure repeated to consider subsequent moves that might follow. Looking ahead more moves would be expressed in the depth of the network. Feed-forward networks have a constant evaluation time, which is a function of the number of nodes in the network, which means the vast majority of computation time would be spent on moves and sequences that should be easily rejected.

Instead of a purely feed-forward architecture, AlphaGo's deep network classification of board-states can be interpreted as "generally good" or "generally bad". The space of possible moves is explored using Monte Carlo Tree Search, a stochastic system that uses the approximation of board-state quality to focus computational resources in promising directions and deemphasize less promising directions [3, 32]. This combination of specialized classification systems and additional structure addressed the particular limitation of purely feed-forward networks for planning long sequences of moves.

Combining the classification ability of deep networks with additional structure is also the approach used by Situate for the problem of situation recognition. In addition to wanting to navigate complex hypothesis spaces that make situation understanding challenging, we would like Situate to have properties that will be important for addressing the aforementioned challenges for machine learning. Some of the properties that we hope to foster include:

Understandable decisions

One of the ways that a classifier can fail is to provide a confident but wrong answer. Most classifiers operate as a "black-box", producing a classification without an interpretable indication of how the classification was made or what evidence the system used to support the erroneous classification. This makes it difficult to trust classifications and makes it difficult to know what to change to improve the system. If systems expose their decision making process in an understandable way, humans will be able to address the problem, by correcting properties of the training set that led to errors, enforcing explicit rules, or accepting the error as something that does not require explicit correction.

Meaningful confidence reporting

Another way that classifiers can fail is to provide an inconclusive response.

Users often interpret these responses as "low confidence," but the contributing factors to a low confidence decision are left unclear. The system may have interpreted an input instance as an edge case between known categories, may have seen an instance as a poor example of a known category, or the instance may have been from an unknown category. What a human or system does with the instance may depend on these factors, but the information is not typically available. Classifiers that provide additional information about the final state of a decision can address this limitation.

The ability to integrate human prior knowledge

Often classification is a combination of strict rules and something more like subjective decision making. For example, harassment in social media posts might be recognized by the inclusion of any of a set of specific words or sequences, but something more like sentiment analysis may also indicate harassment. When humans want to provide this sort of information to a machine learning system, it is often done through curation of massive data sets. This makes it difficult to know if the rules that we hope the machine learning systems are learning are actually being learned as strict, rather than just approximated in *most* situations.

Consider the problem of recognizing the dog-walking situation. It requires an enormous amount of data to train classifiers to recognize dogs and people. However, once the object of detection is something more complex, such as the relationship between dogs and people in an image, it is less clear that available data sets are large enough to allow for a network to learn a) to detect people, b) to detect dogs, c) to detect leashes, and d) to encapsulate the variance in configurations that we recognize as "dog-walking". Even if the data is available, the time spent re-learning constituent objects is largely wasted, as classifiers for

these objects exist.

An ideal system would allow for the efficiency and reliability of rule specification, but with access to the classification ability of neural network-based systems. We could specify that dogs and people are both present and recognizable via classifiers trained on those object types specifically, are near one another, are usually walking in the same direction, and have a leash between them. This could be expressed in a concept graph requiring little change to the object classifiers involved. By allowing a human to define the structure of a situation, the machine learning system is able to bypass what would likely be a long retraining process.

A flexible trade-off between run-time and response quality

When using a machine learning system to make a decision, the completion of the decision making process is binary, complete or incomplete. If provided with more run time to make a particular decision, the system does not improve its response (by doing something like ruling out additional alternative explanations for observations), and if less time is provided, it will not be able to provide a partial response (such as a low confidence, preliminary classification). Ideally, a system would have a running estimate of its response that would become more confident over time, allowing the time spent on a problem and the confidence of the response to be variable depending on the importance of the decision being made.

Flexible adherence to classification criteria

Although the response to a classification task may be negative, it is a valuable property of human cognition that we are able to see that small deviations from our prototype would allow for inclusion of an instance into a category. We interpret with flexibility, and can communicate that information. For example,

Figure 1.1: Several examples of the "dog-walking" situation. Each image contains one dog-walker, one dog, and a leash connecting the two. Clutter is made up of other people, cars, trees, and other everyday objects. (Best viewed in color)

consider Figure 1.2. If asked if these images depict people walking dogs, an optimal response might be akin to "No, but they are close in the following way..." or "Yes, given the following relaxations...".

Situate takes an agent-based approach to solving complex problems. Much of its architecture is based on the "Copycat" program [16], a system developed by Melanie Mitchell and Douglas Hofstadter that made analogies between abstract, symbolic "situations". Situate uses neural networks to interface real data and a symbolic space. It also has similarities to Monte Carlo Tree Search, which shifts the distribution of computational resources toward partial solutions based on their estimated quality. The architecture of Situate helps avoid the difficult encoding problems associated with genetic algorithms, and is compatible with desirable features described above (such as flexible decision making and the inclusion of prior knowledge) that are not a natural part of Monte Carlo Tree Search.

1.2 Situate and situation recognition

Situate is an *agent-based* system for recognizing visual situations, where *situation* refers broadly to a named concept that relates to one another a collection of objects and their properties. An *agent* refers to a structure that contains information about the input that influences how the agent operates. Situate uses multiple agents that

each store different information about the input and therefore behave differently from other agents. Situate allocates computational resources to the agents based on the quality of their findings. This allows Situate to consider multiple interpretations of an input, and to provide insight into the final decision by making available information gathered by other agents, thus providing a record of alternative interpretations that were rejected.

Visual situation recognition is a task where an image is provided to a system that generates a numerical score indicating its confidence that the situation is depicted in the image, as well as providing the evidence that was used to satisfy the situation requirements.

The *situation recognition* problem is: given an input and a situation definition, determine if the content of the input satisfies the situation definition. Furthermore, provide a *situation grounding*, a structure indicating the evidence that appropriately supports a positive decision. For visual situation recognition, the input is an image and the situation grounding consists of bounding boxes that localize the constituent objects of the situation.

The situation definition is a graph that can specify: included objects, properties of objects, and relationships between objects. In the case of "dog-walking", the simplest situation specification requires that there be a person, a dog, and a leash, and that the person and dog be connected by the leash.

The evidence for the decision consists of a set of bounding boxes that localize the constituent objects and satisfy a quality threshold defined by a ratio between intersection and union of the bounding box predicted by the system and a human-defined ground truth bounding box. More precisely, for ground truth bounding box B_{gt} and a predicted bounding box B_p,

$$IOU(B_{gt}, B_p) = \frac{area(B_{gt} \cap B_p)}{area(B_{gt} \cup B_p)}.$$

An object that is correctly labeled and is specified by a bounding box with an IOU score with the ground truth box over .5 is considered sufficiently detected. [a]

[a] The .5 intersection over union threshold is used commonly in localization tasks. For example: [29].

The primary example situation used while developing Situate was the visual situation "dog-walking", wherein a dog and a person are connected by a leash. It has been a useful example situation as it demonstrates several of the challenges that make

situation detection distinct from simple object detection, including frequent distractor objects that are similar to one of the situation objects (people) and an object that is difficult to recognize without context (leash). Figure 1.1 shows examples of the dog-walking situation.

Situation recognition, in itself, is a challenging task, requiring the recognition of constituent objects, their properties, and relationships. The variability in properties and the combinations of objects that make up relationships leads to a large number of possible interpretations of any input, only a few of which may lead to the correct detection of the situation of interest. Figure 2.1 shows an example of the dog-walking situation, as well as a sufficient detection and several insufficient detections of the situation. There are several ways in which a detection can fail, including the detection of an object that might have the same label as an object in the situation, but which is not actually involved in the situation (as can be seen in Figure 2.1d).

Situation recognition is meant to be a problem that cautiously approaches issues of *understanding* in machine learning. Some of the properties we associated with understanding a problem and an input include the ability to provide evidence supporting a response, the confidence the system has in that response, and the ability to provide flexible responses. For example, we would like Situate to leverage the similarity between the dog-walking situation and some closely related situations, such as the non-prototypical instances of "dog-walking" shown in Figure 1.2, and to express what relaxations of the situation definition are required to accept the input as an instance of a particular situation.

Situate approaches the problem by trying to construct solutions, rather than evaluate all possible solutions. It does this by finding relevant objects and using information about the situation to direct the search for related objects and properties that can be assembled into a known situation. This approach allows Situate to expand

Figure 1.2: Several examples of atypical instances of the "dog-walking" situation, where the situation specification does not strictly apply. For each of these images, we would like Situate to recognize the similarity to "dog-walking" and to be able to indicate what relaxations would allow for the instance to be identified as such. The strength of the required relaxations (dog-walker → dog, dog-walker → person on bike, dog → two dogs, dog → cheetah) should influence how close the input is to being seen as an instance of "dog-walking". (Best viewed in color)

on findings quickly, making it possible to find positive cases quickly, and to avoid exhaustive searches.

Situate uses a pool of agents, each having a state (called the *Workspace*) that contains its beliefs about the input, a method for generating hypotheses about the input (called the *situation model*), and access to methods for evaluating hypotheses (such as a set of image classifiers). Each agent tries to build a solution based on the information it has, and Situate allocates computational resources to those agents based on how promising their partial findings are, and how much progress they are making. Agents that are productive and have promising findings are allocated additional resources early, hopefully finding a solution quickly. Agents that do not continue to make progress are deemphasized in the resource allocation process. In the case of visual situation recognition, the most computationally expensive action is a call to an object classifier. A single call to a classifier and any necessary updates to an agent's Workspace make up a single update iteration. For this reason, *computational resources*, *calls to a classifier*, and iterations are used somewhat interchangeably.[1]

[1]In practice, there are differences between *iterations*, *calls to the classifier*, and *computational resources*. Each *iteration* consists of a call to the classifier and *potentially* an update to the agent's

1.3 Walk-through of an agent evaluation

Situate manages multiple agents, each of which have their own state, have their own sampling priorities, and may produce different situation groundings. The method for allocating resources to agents is discussed later in chapter 4. Before describing the structures and methods of Situate in detail, it can be helpful to see what it does in practice by seeing what a single agent does. Figures 1.3 and 1.4 show the state of a Situate agent at several points during its evaluation of an input. It shows objects detected, how it focuses on specific image regions as its Workspace develops, and an example solution.

Workspace and associated structures, but also may not involve any update. Additionally, Situate agents keep a record of recent calls to the classifier so that, if a call is sufficiently similar to an earlier call, the results of the earlier call are returned instead. This reduces the computational cost of calls to the classifier, but are still counted as iterations. The assumption that calls to the classifier make up the majority of the computational cost associated with an iteration is true given the classifier being used and the cost of updating the model of the situation when the Workspace changes, but may not be true if different classifiers and models of the situation were involved.

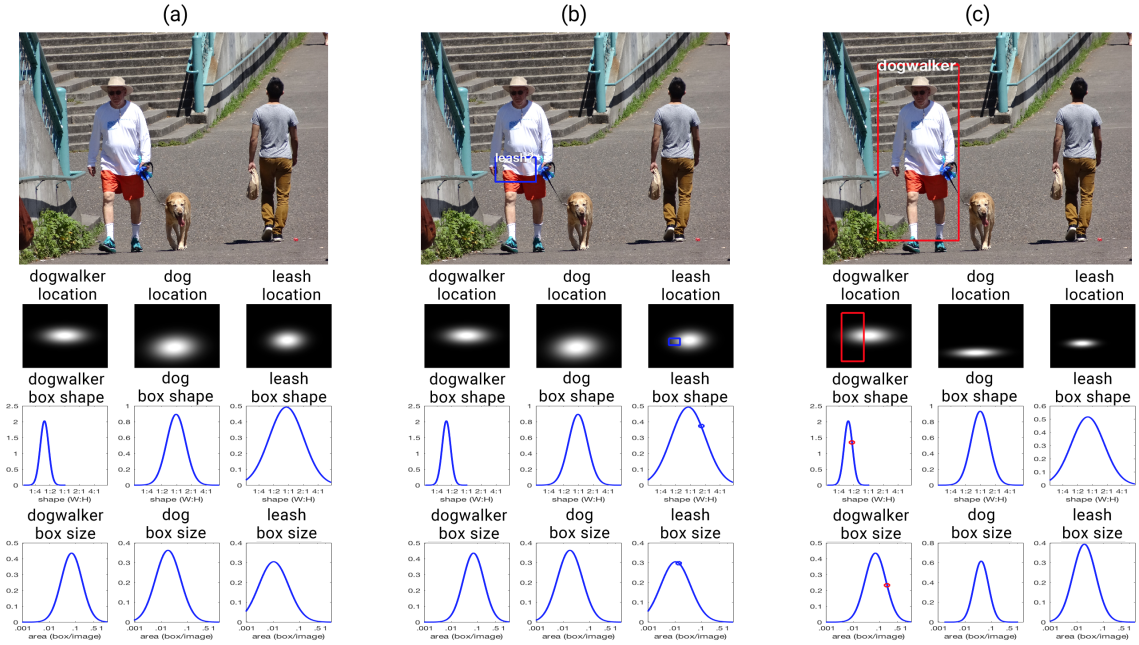


Figure 1.3: Column (a) shows an input image and the prior distributions for the objects in the dog-walking situation. The prior distributions include the location distribution (here represented as a gray-scale image with white indicating regions where the center of the bounding box is likely to occur), shape distribution, and size distribution for each of the objects in the situation.

Situate makes predictions by sampling bounding box parameters from the situation model. Column (b) shows a sampled bounding box for the leash object. The sample appears in blue on the image and spatial location distribution. The sample's shape and size are represented with a blue dot on the shape and size distribution curves. The sampled image region is sent to the appropriate classifier for evaluation.

If the classifier response is over a threshold, the object is added to the agent's Workspace. Column (c) shows a red bounding box for the dog-walker class that has been added to the Workspace. The bounding box appears on the image, the spatial location distribution, and the shape and size distributions. With an addition to the Workspace, the situation model updates its expectations about the other objects in the situation. Notice that the expected locations for the dog and leash bounding boxes have narrowed. (Best viewed in color)

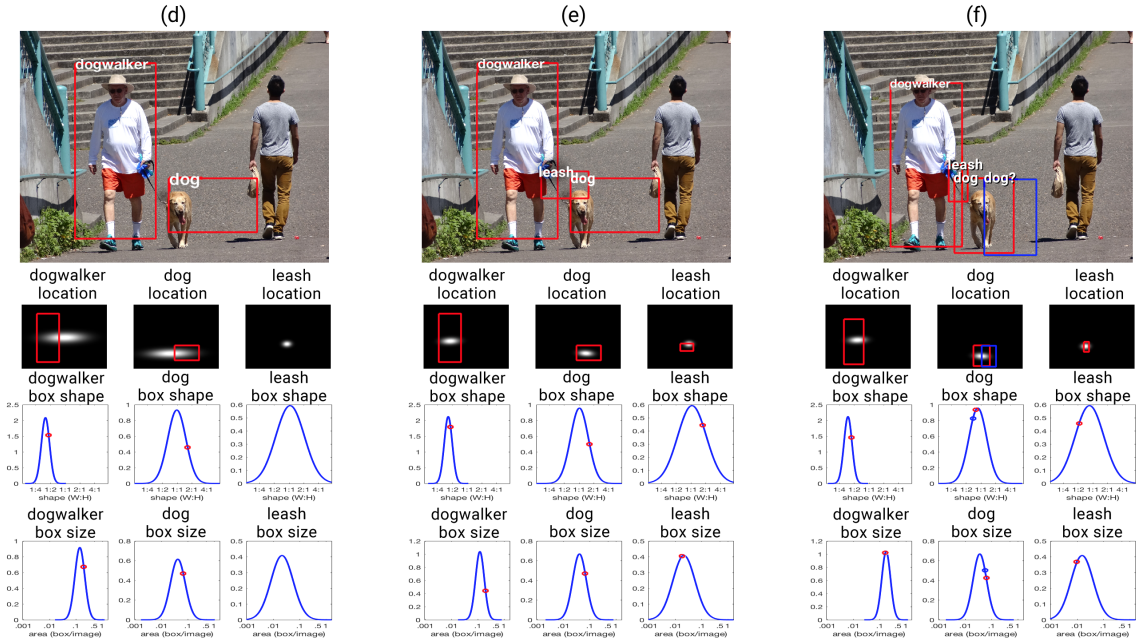


Figure 1.4: Columns (d) and (e) show a later state in the evaluation as more objects have been found and the situation model has further narrowed down on likely parameters of bounding boxes for objects in the image.

Situate continues to sample from these distributions repeatedly. As the distributions have narrowed, the samples are more likely to be small deviations from what has already been added to the Workspace. This allows the contents of the Workspace to be updated with higher quality bounding boxes for the constituent objects. This higher quality is a function of both the confidence of the classifier and consistency between objects (in terms of the likelihood of their bounding box parameters with respect to the situation model). Column (f) shows the final state of the Workspace. Each of the objects of the situation appear to have been localized quite well. The dog-walker and leash have been localized sufficiently to satisfy the .5 IOU threshold requirement, but the dog object has not been (as the bounding box is a bit too large and produced an IOU of .39 with the ground truth bounding box). This example result highlights the difficulty of the situation recognition task even for relatively simple examples of the situation.(Best viewed in color)

1.4 Related work

The situation recognition task is related to common vision tasks in machine learning. Like image classification, in which images are labeled with a binary indicator of whether or not a particular object type is present in an image, situation recognition is concerned with mapping the content of images from pixels to something more symbolic. However, situation recognition is also concerned with the relationships between the objects represented in the image. Automatic image captioning, a task in which natural language descriptions of the contents of an image are generated, is also interested in both the content and the relationships between objects in images. However, Situation recognition differs from image captioning in that it limits the problem to symbolic relationships that are expressed graphically, rather than intermingling the problem with natural language, its generation, and its evaluation.

Some computer visions tasks overlap substantially with situation recognition, but are often quite a bit more specialized, which can make the tasks more immediately applicable to problems that exist for the fields that generated them, but also can lead to methods that are hard to generalize and apply elsewhere. For example, the action recognition task tries to identify the action that a person is engaging in, often relating to other objects, or being defined by body position. The actions to be recognized could be expressed as a situation recognition graph relating a person's body parts to one another and to other objects, but because the action recognition task is almost always related to the human body, solutions often tend toward modeling the human body quite specifically.

There are several tasks and methods that relate to the situation recognition task and to Situate in ways that I believe are worth noting. Below, I discuss several such tasks and methods, and their similarities to situation recognition and Situate.

1.4.1 Methods related to object recognition and localization

Object localization

For the *object localization* [9] task, the input consists of a label and an image that may or may not contain an object to which the label would be appropriately applied. A correct output specifies the location of the object in the image. The structure of the specification can vary, but commonly it is a bounding box that contains the object with a sufficient overlap with a human-provided ground-truth bounding box. A common measure of quality (intersection over union) and threshold for sufficient localization (.5) matches the standard for correct evidence in the visual situation recognition task as defined.

The difference between situation recognition and repeating the object localization task for each object in a situation is that situation recognition requires the selection of the correct instances of objects in the situation, which is indicated by the relationships between objects.

Like much of the work related to semantics in images, attention in computer vision, and automatic image labeling; Situate uses object localization as a starting point, but includes additional selection criteria and search mechanisms that are task specific, such as in [19].

The usefulness of a purely object localization approach to detecting the entirety of a situation, represented as the minimal bounding box that contains all of the objects of the situation, is briefly evaluated in chapter A.1, although the evaluation metrics are discussed in chapters 2.1.1 and 5.

Region Convolutional Neural Network (R-CNN)

A successful and commonly used method for object localization is based on creating a large set of potential bounding boxes for objects and then sending

image regions specified by those bounding boxes to an object classifier. The set of bounding boxes may be the result of a complex preprocessing step, or may be from a large set of predetermined bounding boxes that are not a function of the image content. The system then returns the bounding boxes associated with the highest classifier confidences. There is variation between these systems in the form of a) the method used for proposing bounding boxes and b) the classifier used to evaluate the associated image regions. R-CNN established this pairing of mechanisms and was briefly the gold standard for image localization [12]. It combined an existing proposal mechanism called "selective search" [34] with an existing deep neural network for classification [21], as well as a *bounding box regression* system, which uses convolutional neural network features to tune the parameters of the bounding box to more accurately localize a detected object.

Subsequent projects integrated the method for generating a collection of bounding box proposals and the method for classifying those boxes by using the same deep convolutional network for both phases. Faster-RCNN [28] does this by applying a deep convolutional network to an input image, but stopping before features from everywhere in the image have been combined and classified, providing a set of useful features that retain some region specificity. A set of regions are defined and evaluated with respect to the feature activations that lie within their receptive fields. The features are used to generate an *objectness* score. These scores are used to select boxes likely to contain objects. The rest of the classification process is then applied to those boxes that have high scores. This produces a confidence value for each of the boxes for one of the known object types.

After the bounding boxes have been classified and scored, R-CNN and Faster-RCNN use bounding box regression to tune the parameters of the box for the

selected object type. Bounding box regression, as described in [12], and in turn referencing *P. Felzenszwalb, et al.* [10], uses a series of regressors that predict the difference between a proposed bounding box b_a and the ground truth bounding box b_{gt} for an object in terms of several parameters. The regression targets are these differences, expressed as follows:

$$\Delta_x = (x_{gt} - x_a)/w_a$$
$$\Delta_y = (y_{gt} - y_a)/h_a$$
$$\Delta_w = log(w_{gt}/w_a)$$
$$\Delta_h = log(h_{gt}/h_a).$$

Where x_g indicates the x position of box b_g, x_a indicates the x position of box b_a, and so on. Given these differences, an improved bounding box can be constructed as follows:

$$b^*_x = x_a + w_a\Delta_x$$
$$b^*_y = y_a + h_a\Delta_y$$
$$b^*_w = w_a e^{\Delta_w}$$
$$b^*_h = h_a e^{\Delta_h}.$$

To predict the Δ values, a linear regressor is trained using ridge regression [15] to map from the convolutional neural network (CNN) image features extracted from the image region defined by box b_a to each of the regression targets. Data used to train the regressors consists of bounding boxes that are similar to ground truth bounding boxes, but differ slightly in their parameterization. Only boxes with an intersection over union score greater than .6 are used for training, which

is to say, boxes that are enclosing their objects of interest fairly well.

After proposed bounding boxes are adjusted, R-CNN does not re-compute the classification score for the proposed boxes, meaning the bounding box regression process is only intended to make small adjustments to the width and height of the bounding boxes, rather than changes significantly change the shape or location. The bounding box regression used in Faster-RCNN differs from the above described procedure in minor ways, such as having a larger set of regressors from which to select depending on the box's shape, size, and the initial object classification.

R-CNN and its descendants return as many instances of recognized objects as surpass a confidence threshold. To use R-CNN methods for situation recognition, each constituent object needs to be searched for and an instance of the object must be selected. For comparison purposes, the selection criteria I used was to select the most confidently detected instance for each constituent object of the situation. Then, the geometric mean of those confidences are combined to produce a single decision value called a situation score.

Overfeat

Overfeat introduced the integration of localization and classification into a single, end-to-end system [31]. Images are fed into a single, deep convolutional network and a large number of bounding boxes are produced. The system uses the accumulation of bounding boxes to localize objects, rather than by culling the large set down to a few bounding boxes to evaluate. Overfeat also performs positive classification on multiple object categories at once. By classifying many object types, confusion between object types can be reduced. The intuition is that the correct classification often produces higher confidence than an incorrect classification. This allows Overfeat to bypass a "background" class, but

requires a very large amount of labeled training data. Essentially, Overfeat needs to have a model for each object type that might appear in the input to avoid false detections.

R-CNN and variants are used more often than Overfeat (in terms of citations and other systems that use one or the other as a pre-processing step), this basic pipeline remains the state of the art for object localization with convolutional neural networks. An input image is passed through a system that produces bounding boxes (often the same CNN used for classification). These boxes are reconciled in some way that limits how many of them are considered. Each region is given a classification score.

Active search

An alternative to the feed-forward approach to detection is *active search*, which is the sequential analysis of inputs. When the input is a single image, there is an analysis loop that investigates a region, results from that investigation are stored, the stored information is used to select the next region for analysis, and the loop continues until some stopping criteria is met. Situate fits firmly into the active search category.

Much of the active search literature is associated with robotics research, as the loop of active search fits naturally with the updating state of the robot and updating inputs to its sensors. Active search has regained attention in the general computer vision field in recent years due to the expected benefits of integrating contextual information into vision tasks, as well as expanding general interest in computer vision for robotics. The performance of active search systems is often considered in terms of the number of regions evaluated before some sort of decision can be made [1, 13].

The above methods utilize variations on a "belief map," where locations on the

map represent the current belief that the object of interest is located there. After evaluating a proposal, the map is updated. Updates are not just the evaluation of the location searched, but regions around that location. For example, if the system is looking for doors on buildings, and it detects features consistent with a window, it may increase the belief that there may be doors next to or below the detected window. The maximum over the map is used to select the next location to be evaluated. The evaluation takes into account the image features of the window, as well as "context," which uses previously observed regions in the image and estimated displacement vectors between those regions and possible locations for the target object. Gonzalez-Garcia et al. modeled the estimated displacement vectors for the object of interest during training using a random forest [1, 13].

In addition to using active search with related objects to locate objects of interest, active search has been used to improve the localization quality of bounding boxes. Caicedo and Lazebnik show that a reinforcement learning approach can be used to update a bounding box for an object [2]. Evaluation of active search methods are often reported in terms of detection accuracy for an object of interest as a function the number of regions analyzed, a method that we also use to evaluate Situate.

Like Overfeat, Situate avoids using a "background" class, but does so in a way that is specific to the situation recognition task. Situate is trained with knowledge of a small number of object types. Background objects can then be confused for objects of interest, as they cannot be attributed to their correct classes. However, if they do not have contextually supporting objects, they are rejected as not a part of the situation of interest. There is a benefit to this approach in that the amount of training data required for the situation recognition task specifically is dramatically reduced when

compared to Overfeat-like systems (as only relevant objects are learned). Instead of the 1000 classes used in Overfeat, Situate uses only the objects included in the Situation of interest (although nothing prohibits the inclusion of additional object types for a more thorough labeling).

Like the active search methods discussed, Situate uses a method for accumulating context that informs what the system pursues. However, Situate integrates the situation structure when deciding where to "look". That is to say, the contextual evidence that Situate will leverage is provided by a user, or generated as a part of a separate process, rather than from purely bottom up evidence. This is not always an advantage, as one might want to specify only the target of the search and have no input on the process, but separating the two sources of information has advantages.

- First, if there is limited training data for the whole situation, the components can be learned and the relations specified. For example, atypical situations that specify objects that can be identified, and relations that are common between other objects, can be used to define a search. Something like a person walking a cat where cat replaces dog, as people walking cats is uncommon, but people walking dogs is relatively common. Essentially, this opens up the system to explicit knowledge transfer.

- Second, context can be learned separately and used as a known situation, but the known situations that are reasonable for the system to use can be audited, as they will not be obfuscated by being embedded within a neural network. This issue is of particular importance as the methods by which machine learning systems make decisions are being increasingly scrutinized by those affected by those decisions [2].

[2]Metz, Cade "We Teach A.I. Systems Everything, Including Our Biases", *The New York Times*, Nov. 11, 2019

1.4.2 Tasks related to situation recognition

Methods for recognizing and localizing individual objects have coalesced into a relatively consistent pipeline, meaning that much of the work related to Situate is related in one of two ways: first, by similarities in either the definition of tasks beyond simple object recognition and localization; or second, with respect to the approach to integrating object detection into a more complex system.

In identifying what makes these systems distinct and interesting, the important features seem to be: the construction of tasks, the applications for which systems were originally defined, and their applicability to closely related tasks. Below are several tasks that are motivated by an interest in multi-component classification or situation and context based tasks. I believe that some of these tasks are essentially subsets of situation recognition as I've defined it, as they can be expressed in terms of objects and a simple graph of relationships.

Action recognition

Action and pose recognition have been tasks of interest for some time, often as part of a pipeline that includes motion capture, pose estimation, and action recognition. Much of the research in action recognition uses video inputs and makes significant use of matching video sequences in frequency space. However, a subset of action recognition tasks, static action recognition, uses still images and is more similar to the goals of Situate. Static action recognition is generally associated with estimating the pose of the human body specifically, and associating it with actions such as walking, running, or athletic activities [14].

Recent work in static action recognition has been focused on applying a number of well known deep learning models to existing data sets. L. Wang et al. found state of the art results by combining a deep network that focuses on the person

of interest with a network that processed the full scene. The networks were combined by using a linear combination of the scores from each network [36].

We believe our formulation of situation recognition can account for action recognition in still images, as body position can be expressed in terms of a graph of body parts and relationships between them, but have not directly applied Situate to traditional action recognition tasks. We would be interested in exploring it in the future. The approach of combining networks that process the input at different scales and are linearly combined was shown to be useful in identifying actions in images, but the approach does not seem to be generally applicable in understanding more complex combinations of objects and relationships between them. However, some of the more traditional approaches of estimating pose, which involves identifying the components of the body and then classifying the configuration, is actually quite similar to our general approach toward situation recognition.

Referring expressions

A referring expression is a phrase that specifies a particular instance of an object in an image. Usually, the object is not unique based on simply a label, so requires additional information to uniquely identify it. For example, given an image that contains two people, one of whom is walking a dog and the other is sitting on a bench, referring expressions of interest might be "the person walking the dog" or "the person sitting on the bench".

The referring expression task generally goes in the direction of expression to localization. That is, given a referring expression, localize the object of the expression. The general shape of solutions to these problems involves identifying objects with labels that occur in the expression, then identifying relational expressions, and then finding objects that most strongly represent the expression.

The identification of objects is performed as in other object localization methods I've discussed, using one of the standard convolutional neural networks.

To score the relational term, there may be separate networks trained for the term when associating two arbitrary objects, and when associating objects that are frequently associated by the term, depending on how frequent the relation and objects are in the training corpus. In either case, the smallest region of the image that contains both of the objects being considered is cropped and sent to the appropriate relational classifier. The confidence of this classifier, as well as the confidences of the component classifiers, are combined to generate a final score for the object set.

Possible pairs of objects in the image are scored, and the highest scoring pair is returned as the predicted subject of the referring expression.

Referring expression tasks are interesting and share a lot of the challenges that are interesting about our formulation of situation recognition. Both include multiple objects and relations. However, the free-form nature of referring expressions led it to intersect with natural language processing, and the enormous number of possible relations has meant that the highest performing methods report low accuracy, even when restricted to very simple expressions that include only two objects[25].

Additionally, and unfortunately, it appears that the semantics of the expressions have less influence on the final model accuracy than one would hope. Cirik et al. [5] investigated the functionality of several state-of-the-art referring expression recognition systems by providing original and randomly permuted referring expressions. The permuted expressions caused the systems to select the correct object only slightly less frequently. For example, the top scoring system selected the correct object for 68% of trials with a permuted referring expression rather

than 71% of the time with the original expression. In fact, when the entire referring expression is removed and a system returns objects simply in order of general salience, the object of the referring expression is identified in the top two responses in 73% of trials. This led the authors to conclude that there is significant photographic bias in the standard datasets (such as Google-RefExp [24]).

Our goal with the situation recognition task is to step away from the complexity of the referring expression task, to restrict the problem to clearly defined relationships that we express in graphical form, and to include the support that led to the response in the evaluation, making it easier to verify that the system is doing what we think, and to allow us to diagnose the behavior of the system.

Graph-based Image Retrieval

Image Retrieval using Scene Graphs [19] describes a system (which I will refer to as IRSG) for retrieving images from a database using a *scene-graph* as the query. The scene-graph contains objects (such as "man"), attributes of objects (such as "tall"), and relationships between objects (such as man "holding" leash). When a query graph is submitted, IRSG tries to ground that graph to images in its database. The images that produces the strongest groundings are returned.

The system that constructs the grounding for an image consists of visual classifiers for objects and traits, and Gaussian mixture models for relationships. The training data consists of a large set of images with exhaustive labeling of objects, traits, and relationships. A neural network is trained to recognize each object type and trait. A Guassian mixture model (GMM) is trained for each relationship in an object-agnostic manner, as well as for frequently occurring object-relationship-object sets. For example, there may be visual classifiers for "man" and "horse" and a GMM for "man-riding-horse", but no GMM for

"horse-riding-man" (although there is a GMM for "x-riding-y" for arbitrary objects "x" and "y").

The grounding is constructed and scored by 1) generating bounding boxes for objects in the image using Faster-RCNN, 2) scoring those boxes with respect to the trait classifiers, and 3) scoring possible groundings of the graph using a conditional random field (which integrates the object score from Faster-RCNN, the trait score, and the likelihoods from the relevant Gaussian mixture models).

This system is comparable to Situate in its intention and is the most applicable to the situation recognition task. The most notable difference between Situate and IRSG is that Situate is a dynamic system. Image regions that may never be considered by IRSG (for being insufficiently object-like) can receive attention in Situate due to their relationship to detected objects. This particular dynamic is observed in the initial implementation of Situate, and may account for differences in performance between Situate and IRSG observed in chapter 5.1.

Since the introduction of scene graphs, there has been continued research interest in the topic. Image data sets that include scene graphs that can be used for training as well as for question-answering tasks have been developed [17], and scene graphs have been used in combination with generative vision systems to modify images via modification of their scene graph [18].

Automatic Image Captioning

A task that has received significant attention in the past several years is automatic image captioning, wherein an appropriate natural language description of the the content of an image is generated. The task has an obvious application in improving accessibility for people with limited vision, potential applications for search, and holds a special appeal due to its similarity to the Turing test.

The basic structure of captioning systems are generally similar to one another. Given an image, a visual classification system identifies some combination of objects, object traits, object relationships, and context. Those findings are stored in an intermediate representation. Then, natural language sentences are generated that correspond to the intermediate representation. The intermediate representation might contain elements that explicitly correspond to named objects and traits or may be a more abstract space that is generated using human generated captions [37].

Evaluating the quality of captions generated by different automated systems presents challenges that are familiar in other speech generation fields, such as chat-bots and language translation systems. Human evaluation of the quality of automatically generated text can ascribe more understanding than is actually present in the output. The long recognized Eliza effect [16] seems to predispose humans toward seeing intelligence and complexity where there is little.

Automatic methods for scoring automatically generated captions are adapted from language translation systems, such as the BLEU score. These methods score by comparing the output of the automated system to a reference text generated by humans. This score consists of counting the number of n-grams shared between the captions [26]. Limitations generally relate to the lack of a notion of semantic similarity between words, which can lead to captions with lucky sequences of filler words outscoring semantically good translations that differ in specific vocabulary [3].

However, in the case of image captioning, additional difficulties arise. Simply reproducing captions from training images based on their intermediate repre-

[3]Sellam, Thibault "Evaluating Natural Language Generation with BLEURT", *Google AI Blog* May 26, 2020 `https://ai.googleblog.com/2020/05/evaluating-natural-language-generation.html`

sentation similarity to a query image competes with complex, recurrent neural network-based language models with respect to BLEU scores. Other language models can produce more unique captions (that are not found in the training data), and are evaluated more highly by human evaluators [8].

What I take from this is that, although automatic image captioning shares some of the goals of the Situate project with respect to understanding images, the reality is that evaluating what a captioning system has discovered about an image via its caption output is complicated by the opacity of convolutional neural networks and by the information (and biases) that might be embedded in language models. A structured intermediate representation, like Situate's Workspace, could help to disentangle some of those issues for captioning systems.

Integration of symbolic knowledge and classification

In recent years there has been increasing recognition that reasoning at the symbolic level is important for machine learning, but also that it may not develop naturally from deep neural networks as they are usually constructed. To what degree symbolic reasoning requires entirely new structures rather than being something that can be coaxed out of neural networks is an active area of research. Situate sits on a far end of this spectrum, with explicit graphs that represent object types, and Workspaces that store parameters of those objects that are determined to be of interest (that is, the bounding box parameterizations). On the other end of the spectrum is research into network dynamics that might be more interpretable and might develop such structures based on cost functions alone. If a particular group of nodes in a neural network can be identified as being related to a particular object type, then it might be possible for a network to begin operating on instances of objects and their parameterizations.

Of particular interest are Capsule Neural Networks[30]. These networks use

collections of nodes, called capsules, which recognize an object as being of a type that the capsule is sensitive to, as well as determining a parameterization for the instance. The terms of the parameterization are not explicit, but can be transmitted to other capsules. Capsule Neural Networks are designed to be trained end to end, which is often a desirable property but means they are slow to train and lack an explicit correspondence between parts of the network and object types.

Many of the above tasks can be expressed in terms of a specific graph and a constrained set of images, but the approaches represented often rely on the specifics of those graphs and (I think unintentionally) on the biases that are present in their most common data sets. By being a bit more general in the construction of the task and by including evaluation of the support used to generate decisions, the situation recognition task emphasizes finding solutions that include the desirable properties discussed in chapter 1.

Of particular interest is the issue of *interpretability*. Interest in interpretability in machine learning seems to be a perennial topic, often regaining attention after major advances in the capabilities of machine learning systems, and a renewed concern over the obfuscated machinations of those systems.

When biomimetic computer vision was at its peak interest, just before the deep-learning revolution, Thomure et al. found that core elements of one of the most studied networks could be replaced with random sub-networks that had undergone no training of any kind, with little loss in efficacy [33]. Landecker et al. found that the regions of an image that were actually contributing to decisions were often totally non-overlapping with the region of the image that a reasonable person would consider critical [22]. These works demonstrated that our assumptions about what is happening inside of networks is often not correct, nor are our intuitions about what is

happening during training, despite resulting in interesting tasks being accomplished relatively well.

Amid the deep learning revolution, there is similar interest in understanding what is happening inside of the most effective systems on particular tasks. A popular approach to understanding visual classification networks has been to construct pseudo-inverse images that maximally activate a classifier. These have produced interesting results, but do not address the issue of how decisions are actually being made in the other direction.

An alternative approach to understanding how a system works, rather than a post-hoc analysis of a black box, is through imposing constraints on the operation of the system. This approach is similar to the general concept of regularization, which is the use of penalty terms during optimization procedures that bias solutions toward those that demonstrate a heuristic notion of how the system *should* work. For example, the idea that a good intermediate representation for a network will contain most of the information necessary for reconstructing an input while using less data (presumably by dropping noise and irrelevant information) motivates the sparsity constraints used when training auto-encoders. Similar relationships between regularization expressions and intentional bias can be found in support vector machines and regularized regression techniques.

The requirement in the situation recognition task that systems generate a decision as well as a set of bounding boxes and relationships that specifies constituent objects is conceptually similar to regularization, as it is a constraint that directs systems toward more desirable solutions, but differs in that it is a constraint on the formulation of the task rather than a classifier training heuristic. This expresses that solutions will need to remain consistent with some prior expectations regarding reasonable approaches that will be trusted by users.

By defining the task broadly, taking an image and a graph as inputs, situation recognition can encapsulate several interesting, related tasks, but does not lend itself to the highly specialized solutions that do not generalize.

In the next chapter, I will describe the functions involved in a single situate agent, describe how the agent develops a solution to the situation problem, describe how it was initially implemented, and evaluate the performance of that agent in identify the objects that make up a situation when presented with a positive instance of the situation. This chapter will demonstrate that Situate agents find solutions to Situation recognition tasks and that the situation model contributes to finding those solutions, but will also demonstrate that there limitations to the agent, but that those limitations are interpretable due to the structured results generated by the situation recognition task.

Chapter 3 describes and demonstrates improvements that I made to the initial Situate agent in several ways, including: improvements to the object classifier, the bounding box generation process, and the support functions. Despite those improvements, chapter 3 also demonstrates the failure states of a single agent searching for the correspondence between the situation structure and an input image, failure states that are largely corrected when multiple agents evaluate an image jointly.

Chapter 4 describes how Situate manages multiple agents, including how resources are allocated amongst them and how solutions are selected when agents find multiple possible solutions. This chapter also evaluates this full, multi-agent version of Situate using the same tasks as previous chapters, and compares it against a greedy method using Faster-RCNN boxes and against IRSG.

Chapters 5 and 6 evaluate Situate on image retrieval tasks, evaluate Situate on additional situations, and discuss future work.

Chapter 2

Function of a Single Situate Agent

In this chapter I describe the functions associated with a single Situate agent. Situate uses a multiple agents when evaluating an instance of the situation recognition task, where each agent gathers information, adjusts its expectations based on findings, and develops a possible solution to the situation recognition problem based on those expectations independent of one another. Multiple agents are involved because initial findings can lead agents in unproductive directions and because there may be multiple reasonable solutions to the instance of the situation recognition task. Selecting from among multiple possible solutions, as well as allocating resources among agents, is handled by Situate. This leaves the agent to focus on constructing solutions to the situation recognition task at hand.

2.1 Structures and procedures

A Situate agent develops a solution to the situation recognition task by adding information to (and occasionally modifying information in) a structure called its *Workspace*. The agent's Workspace is a list containing specifications and labels for bounding boxes that the agent believes are relevant to the situation recognition task being performed.

Information is added to an agent's Workspace by generating and evaluating hypotheses regarding elements of the situation. These hypotheses are something like "there is a dog in a bounding box of dimensions b centered at location l." When a

hypothesis is supported by the evaluation, information is added to the Workspace. For situation recognition in images, the hypotheses consist of an object label for one of the situation objects, a location coordinate for the center of the object, and a width and height for a bounding box. The evaluation is performed by sending the region of the image defined by the bounding box to an image classifier. The classifier responds with its confidence that the object is present in the specified location. If that confidence is sufficiently high, the bounding box specification, label, and classifier confidence are added to the Workspace. In figures 1.3 and 1.4, the hypotheses are depicted as blue bounding boxes overlaying the input image with the hypothesized object type in white text.

Agents generate their hypotheses via sampling from a *situation model*. The *situation model* is initialized with prior distributions over the parameters defining the situation. As Situate runs, the situation model is conditioned based on what has been found and added to the Workspace. For the visual situation recognition task, the situation model relates the bounding box parameters for all objects in the situation. Figures 1.3 and 1.4 do not show the full situation model distribution, as it is high dimensional, but do show the distribution projected onto the x and y center coordinate dimensions for each object type (where white indicates high likelihood and black indicates low likelihood), as well as onto the bounding box shape and size dimensions, respectively. Situate models the shape of a bounding box in terms of its log aspect ratio, and the size of the bounding box as the log ratio of the bounding box area to the image area.

To generate a hypothesis for evaluation, the conditioned situation model is marginalized so as to sample parameters related to a single object of the situation, and a sample is drawn. This generates a hypothesis (bounding box) about the input that can be evaluated by a classifier.

To evaluate the hypothesis, the classifier is used to produce a value called *internal support*. This value indicates how much the hypothesis is supported by the evidence provided by the specific parameters that make up the hypothesis in isolation. *External support* is then calculated, which expresses how compatible the hypothesis is with information already stored in the Workspace and is a function of the likelihood of the sample given the situation model. For visual situation recognition, this is the likelihood of a particular bounding box location, shape, and size given the other known objects. The basic form of the external support function is something of the form

$$external\_support(hypothesis, Workspace) = f(P(hypothesis|Workspace))$$

where f is a function with the range $(0, 1)$.

The internal and external support are combined to produce the *total support*. The total support is a function of the internal support and the external support with a range of $(0, 1)$. The total support function is meant to express the confidence in a particular situation object given both the direct evidence and the contextual evidence.

If the total support for a hypothesis surpasses the predefined threshold, the hypothesis is considered supported and the Workspace is updated to reflect the updated beliefs about the input. The updated Workspace causes the situation model to update with new conditioning information, which in turn changes the hypotheses generated during subsequent iterations. This is represented in figure 1.3(c), where a person is added to the Workspace, and the expected locations of the dog and leash are updated to reflect the conditioned situation models for each. Figure 1.3(d) shows the addition of a second object to the Workspace, and the updated situation models. Notice that, once several objects are detected, the expectations become very focused.

This process is repeated until a termination condition is met. The termination

condition may allow for additional iteration after all objects have been detected, allowing for refinement or substantial changes in the Workspace (as depicted in figures 1.4(e) and (f)). However, once that condition is met, Situate returns 1) the bounding boxes and labels from the Workspace of the agent in which it has the most confidence (i.e., the situation grounding), and 2) the *situation support*, a single value indicating how confident Situate is that the returned Workspace meets the situation definition. How Situate manages multiple agents is discussed in chapter 4.

2.1.1 Implementation and parameter setting

Situate's image classifiers and situation model both require training, and the methods used to combine them use a fair number of parameters. Although an end to end training procedure with a single optimization target is a valuable feature in a classification system, Situate's structure means that trained components can be assembled without retraining each component. This means that components can be understood and trusted in isolation, supporting the goals of *understandable decisions* and *the integration of prior knowledge*.

During early exploration with Situate, I found that poor parameterizations can lead Situate to failure states, making it impossible to evaluate the system as a whole or the quality of individual components. Therefore, the components were combined up from the bottom up, starting with the classifier, then the situation model, then parameters that combine the classifier and situation model to make decisions, and finally the logic of how to allocate resources to agents and termination conditions.

Data and validation set

During the exploration process, I used the Portland Simple Dog-walking dataset, which includes a set of 500 natural images of people walking dogs. Each of these images contains one dog, one dog-walker, and one leash connecting a person to

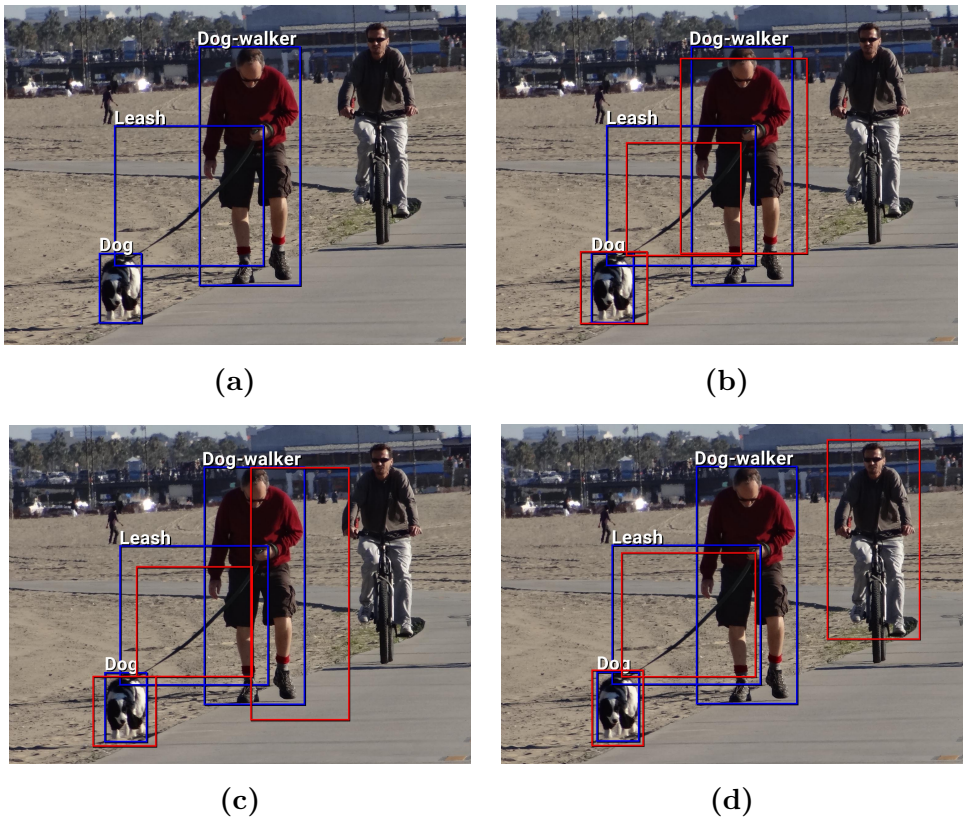


Figure 2.1: Figure 2.1a shows an example of a dog walking situation with human generated ground truth bounding boxes (in blue) from the Portland Simple Dog-walking dataset. The boxes tightly enclose the dog, the person walking the dog, and a leash connecting them. A successful detection requires an appropriate *situation grounding*, that is, a correct localization and labeling of each constituent object. Figure 2.1b shows an example of bounding boxes (in red) that would be considered a successful situation detection, as each bounding box has a label that matches the ground truth label and an intersection over union ratio (IOU) with the ground truth box greater than or equal to .5. Figure 2.1c shows a set of bounding boxes (in red) that represent failed situation detections, as the dog-walker bounding box has an IOU of .33 with the ground truth box, which is under the required threshold of .5. Although there are several people in the image, detecting a person that is not walking the dog (as in figure 2.1d) also would result in a failed detection. The challenge of finding tight bounding boxes, as well as the correct objects when several similar objects are present in the image, contribute to the challenge that the situation recognition task poses to object localization systems that do not account for the relationships between objects. (Best viewed in color)

the dog, and may have distractor objects, including other people. Each image has an associated label file that includes human supplied coordinates for each of the situation objects. The coordinates tightly crop objects with axis-aligned bounding boxes.

From the 500 images, 100 images were reserved for testing at a later time. They were not included in the exploratory phase or parameter setting experiments. From the remaining 400 images, 100 were used as a validation set, used to compare parameterizations. This left 300 images for training. This is far fewer training examples than are generally used for training a system top to bottom. However, this has proven sufficient for Situate as it uses an assemblage of pre-trained neural networks (which are "tuned" with the available data, but do not need to be fully trained) and low-complexity models for the situation model.

Classifiers

The classifier at the core of Situate is a combination of an off-the-shelf convolutional neural network (CNN) for feature extraction and a traditional classifier. Many CNNs, including the one used in Situate, are trained discriminatively to recognize a large number of objects. However, much of the resulting neural network structure is tuned to the statistics of natural images in general, with the discrimination between objects occurring at a late stage (high layer) in the network. This means that the networks do not need to be fully re-trained for every object type to be effective, instead allowing for tuning to our particular collection of objects of interest, where "tuning" here means training a new classifier to map the state of a late layer of the CNN to a class indicator value. Essentially, we replace the existing final layers of the network with a new classifier.

I compared networks described in [4] combined with a linear support vector machine (SVM) for object recognition. I used implementations of the networks

found in MatConvNet [35]. For each of the object types in the dog walking situation, I extracted the tightly cropped object of interest from each of the training images, as well as image crops that did not contain any of the situation objects but were of similar shape and size to the cropped objects of interest. These images served as a background category. An SVM was trained to distinguish between each of the object categories and the background category (that is, one SVM per object category). The trained network was then applied to similarly generated crops from the validation image set. For each image crop, distance to the margin was used as a decision variable and the area under the receiver operating characteristic curve (AUROC) value was calculated. The AUROC is a single value that summarizes the efficacy of a classifier and will be described in more detail later.

I found that most of the convolutional neural networks produced similar discrimination quality for our small evaluation set, so I used the network that was simplest and fastest. This turned out to be the network based on AlexNet [21], the same network at the core of R-CNN and that kicked off the intense interest in deep networks for computer vision several years ago. AlexNet was trained to recognize 1,000 object types from the ImageNet data set. The network is an assemblage of convolutions, local pooling, and eventually a densely connected neural network with many hundreds of thousands of neurons. I replaced the final layers of the neural network, which make the final classification decisions, with alternative classifiers that were trained on data for the situation recognition task. I discriminatively trained an SVM for each of the constituent objects of the situation using 300 positive training examples and 300 negative training examples from the Portland Simple Dog-walking dataset. The positive training examples were tightly cropped instances of the target object. The negative

examples were regions from the same images, that contained none of the objects of interest, with crops that matched the size and shape of the positive examples. I trained an SVM for each of the object types. I used 100 additional images from the Portland Simple Dog-walking dataset to generate a similar set of positive and negative examples, which I then used to estimate the quality of the resulting classifiers. The dog-walker classifier had an AUROC of .98, the dog classifier had an AUROC of .98, and the leash classifier had an AUROC of .90. These each indicate a reliable classifier for the task of discriminating between well cropped instances of their target objects and background images, although the dog and dog-walker classifiers are substantially better than the leash classifier.

This method for building the classifiers is re-evaluated in chapter 3.2.1.

Situation model

The situation model relates the parameterizations of situation objects to one another. I experimented with possible divisions of parameters of objects (such as one distribution that relates object sizes and another that relates object locations), however, I eventually settled on a single high dimensional normal distribution, as it allowed the system to find the meaningful correlations without too much interference. Then, the situation model for the dogwalking situation, which consists of three objects, is:

$$D_{situation} = N(x_{dog}, y_{dog}, p_{dog}, q_{dog}, x_{dog-walker}, y_{dog-walker},$$

$$p_{dog-walker}, q_{dog-walker}, x_{leash}, y_{leash}, p_{leash}, q_{leash}),$$

where N is a Gaussian distribution, x_{object} is the center coordinate of the bounding box for an object on the x-axis of the image, y_{object} is the center coordinate

of the bounding box for an object on the y-axis, p_{object} is the log ratio of the area of the bounding box to the area of the image, p_{object} is the log aspect ratio of bounding box for an object, and q_{object} is the log ratio between the area of the bounding box and the area of the image. The x and y positions both appear appropriately modeled by a Gaussian distribution, as do the log area and aspect ratios of each object.

During evaluation, the input is searched by sampling parameters for a bounding box from the situation model. Bounding box parameters are sampled for a single object, so the situation distribution is conditioned based on detected objects represented in the Workspace, and parameters for other objects are marginalized out. An example of this process can be found in figure 1.3.

The simplicity of a Gaussian distribution and the simple relationships that can be captured is both a benefit to the model and a detriment. We assume that there may be many training examples for constituent objects of a situation, but relatively few examples of the full situation, a simple model that can capture the trends in the situational information is preferable. However, there are also relationships between parameters that are not served well by the simplicity. For example, a normal model cannot recognize that the edges of one object might always be contained within the bounds of other objects.

Internal support function

Internal support relates output of an image classification system to other information available to Situate for deciding what should be added to the Workspace.

The initial construction was intended to approximate the probability that the IOU between a proposed bounding box and a ground truth bounding box was

over .5. That is

$$S_{internal}(c(box, im)) \approx P(IOU_{gt}(box, im) > .5)$$

for an internal support function $S_{internal}$ and the classifier output for an image crop $c(box, im)$. I used a standard SVM as the classifier, with Platt scaling to convert the distance to the margin (a unitless output from the SVM) to a probabilistic value indicating the likelihood that the bounding box has an IOU greater than .5.

Figure 2.2 shows the relationship between the classifiers predicted probability of an IOU greater than .5 and the actual probability. Although the predictions are actually fairly poor, the resulting relationship is monotonic and smooth, which is encouraging.

External support function

External support indicates the level of agreement between a situation object and the other objects in the Workspace. This agreement is based on the density of the box parameterization with respect to the situation model conditioned on the current set of objects in the Workspace.

My approach to scaling is discussed more in a later chapter, but the basic approach involves moving from the range $(0, \text{inf})$ of the raw densities to the range $(0, 1)$ by using a wrapping function. The wrapping function is the combination of a *log* function, a logistic function, and appropriate scaling parameters. Then, the external support function y of the raw density x and the parameter vector $\mathbf{t}$ is

$$y(x|\mathbf{t}) = t_0 + \frac{t_1}{1 + e^{-t_2 \times (x - t_3)}}.$$

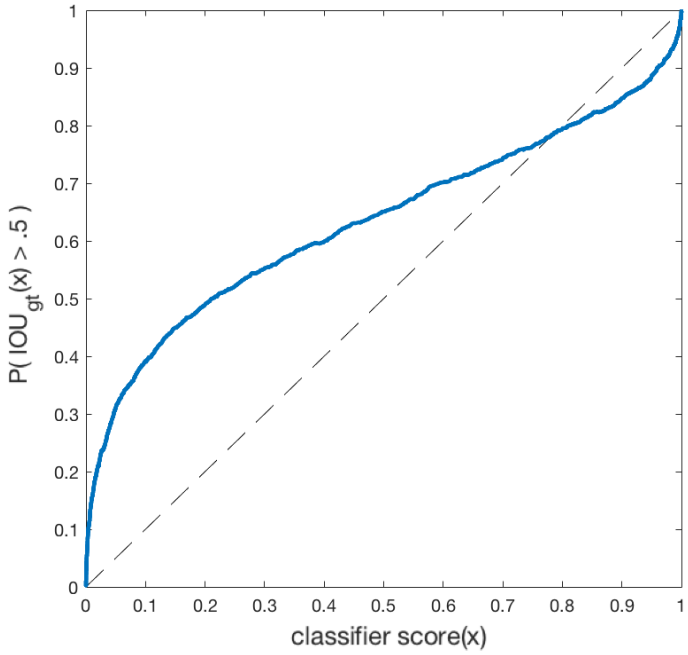


Figure 2.2: The above plot shows the relationship between the SVM classifiers predicted probability of an IOU greater than .5 and the actual probability using a set of validation crops.

The parameters were initially set using bootstrapping. After a solid guess on parameters, letting Situate run on a few training images, and recording sample densities that occurred, I found values for $\mathbf{t}$ that minimized the square error between the above function and the empirical cumulative distribution function of the density values. That is, the vector $\mathbf{t}$ was set such that $y(x_i) = \frac{|\{x|x \in X, x < x_i\}|}{|X|}$. Figure 2.3 shows a visual representation of this process and the result.

Total support function

Total support combines internal and external support for a particular object for the purpose of making a final decision about what that object should be included in the Workspace. Total support could be a prediction of the IOU score for proposed boxes, or it could be the probability that a bounding box, given a Workspace, has an IOU greater than our threshold of .5. As the internal

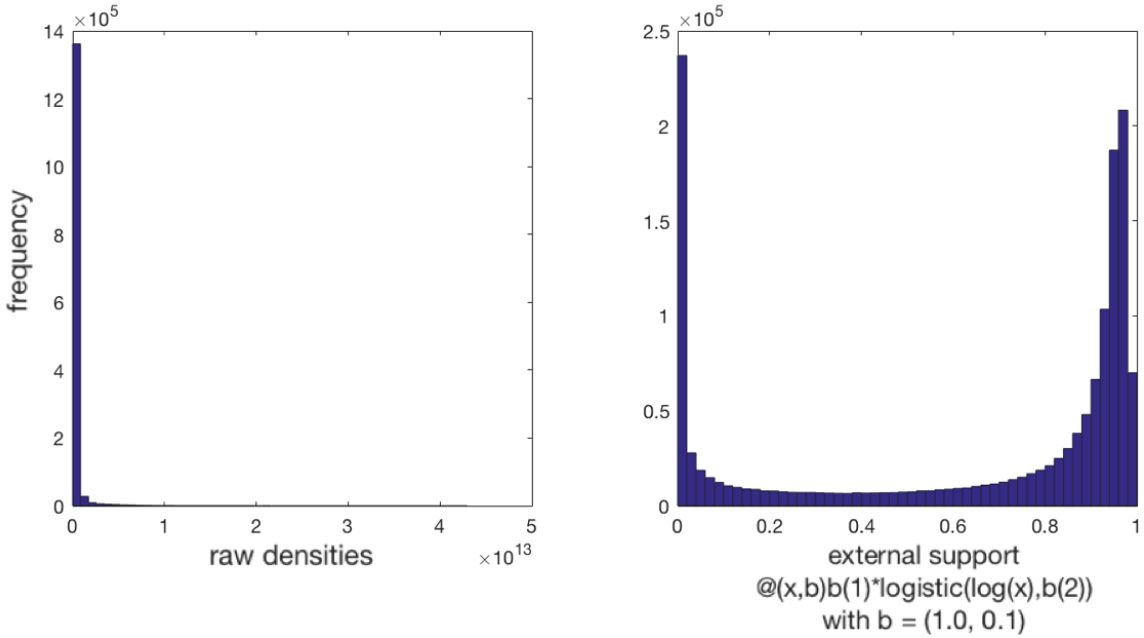


Figure 2.3: The distribution of raw densities from the situation model are condensed near zero and have a long tail of high values associated with a distribution conditioned on several objects in the Workspace. The wrapping function constrains the range to [0,1], making the values more interpretable and easier to use as a component in the total support function.

and external support scores are bounded in [0,1], it seemed reasonable to start with linear combinations of the internal support score, external support score, a bias term, and a mixing term. That is,

$$S_{total}(s_{internal}, s_{external}) = t_0 + t_1 s_{internal} + t_2 s_{external} + t_3 s_{internal} s_{external}$$

where $s_{internal}$ and $s_{external}$ are the internal and external support scores, respectively.For these initial evaluations of the Situate agent, I set the bias and mixing terms to 0 and the internal and external support weights to .5. Alternative formulations are discussed in a later chapter.

Situation score

Situation score is a single value that expresses Situate's confidence that a

Workspace contains the situation of interest. It can be thresholded to produce a binary response, or used to order confidence of a number of images for purposes of retrieval. I used the geometric mean of total support values based on the probabilistic interpretation of total support.

2.2 Evaluation of the initial Situate agent

I constructed the following experiments to verify that the Situate agents' method of sampling regions of an image and updating expectations would lead to appropriate solutions to situation recognition problems. My specific hypotheses were:

- The situation model will lead to a higher number of situation detections than an otherwise equivalent system that uses uniform, static distributions to sample bounding boxes (i.e., a control method).
- Situate agents will more accurately localize constituent objects of the situation (on the basis of IOU) than the control method.
- Localizations of situation objects will occur using fewer resources (in terms of calls to the visual classifier) than the control method.

To evaluate the above hypotheses, I compared several modified versions of the Situate agent and a method based on Faster-RCNN.

Situate agent is a single Situate agent as described above. The agent was run for 300 iterations with no other stopping condition.

Control is a heavily lesioned version of the Situate agent that did not use a situation model at all. It sampled bounding boxes using a uniform distribution over log shape (the box width over the box height) and size ratios (the box area over the

image area). The center location for the bounding box was sampled uniformly from image regions for which the bounding box would lie entirely within the image. The underlying visual classifiers were unchanged. The training data was otherwise used only to define upper and lower bounds for box parameters (without setting different bounds for individual object types). This version was also run for 300 iterations.

Situate - stop is a variant of the Situate agent that had an early stopping condition. As soon as all situation objects had been added to the Workspace with a total support value greater than or equal to .5, the system stopped searching the image and returned the Workspace. This variant was included to establish some expectations about appropriate resource allocations and to determine how stopping early might affect the resulting situation groundings.

Faster-RCNN method uses Faster-RCNN to identify the highest confidence individual bounding boxes for each object type. This method was included to represent a straight-forward approach that uses common deep learning tools. It does not have a complex attentional system nor does it use information about the relationship between objects available in the training data. When using Faster-RCNN to perform situation recognition tasks in this paper, I refer to it as the *Faster-RCNN method*, but it should be noted that this is a simple application that uses the results of Faster-RCNN, and is not a part of the original specification of Faster-RCNN.

Faster-RCNN works by using a fixed collection of boxes of several sizes and shapes, laid out in a grid pattern over an input image. Image features are extracted from the image using a deep convolutional neural network. The features within the bounds of each box are used to evaluate the *objectness* of the contents. Boxes with a sufficiently high objectness score are classified using a

multi-class classifier, which use the already extracted neural network features as inputs. The bounding boxes are then adjusted using a *bounding box regression* system, which maps the parameters of the box, the class of the object, and image features to modified bounding box parameters. Finally, non-max suppression is applied, wherein bounding boxes that overlap significantly with other boxes assigned to the same object class are removed. There is no notion of iterations in this model, as the entire image is evaluated.

Each method was run on 100 images from the Portland Simple Dog-walking dataset that were not involved in the training or parameter setting experiments, or for the fine tuning of Faster-RCNN. I did not run any of the methods on images that did not contain the situation of interest. The initial evaluation only considered how well each method was able to localize the objects of a situation when that situation is present.

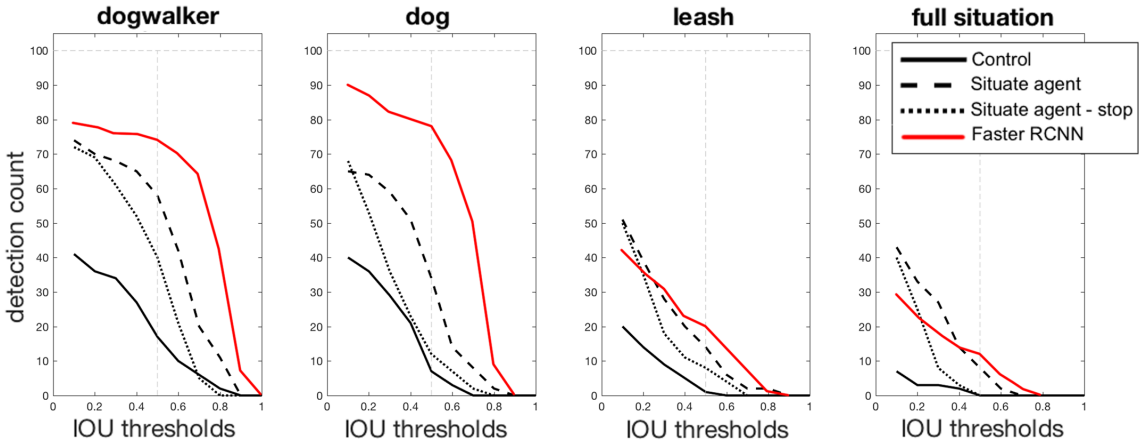


Figure 2.4: Each plot shows the number of instances of objects that were detected at different IOU thresholds. For example, the first plot shows that the control method using faster R-CNN localized 80% of the dog-walkers in the data set with an IOU value of at least .1, and approximately 50% with an IOU of at least .5. We can see that this control method localizes several objects better than the Situate agent, and that the "leash" object is substantially more difficult to localize than other objects for each method. The final plot shows the number of images for which all objects were localized at the specified threshold. (Best viewed in color)

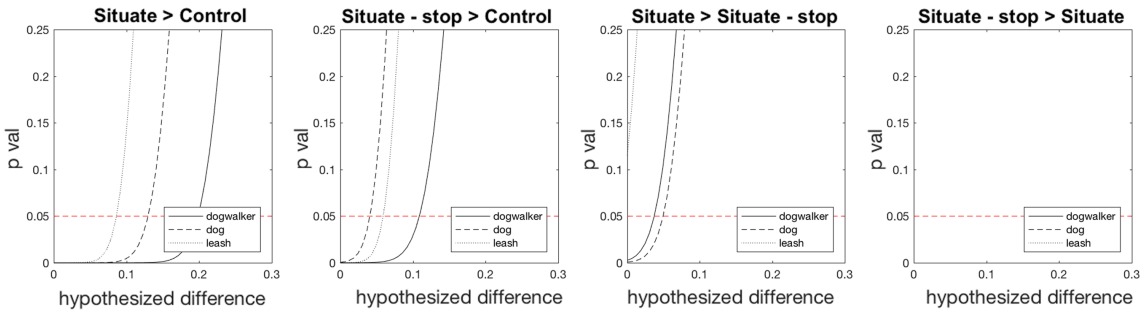


Figure 2.5: Each of the above plots show the the significance of a hypothesized difference in localization quality between two methods and for each object type use the 1-sided t-test. For example, the first plot shows the confidence that Situate was able to localize objects better than the control method. The x-axis shows the hypothesized improvement in localization provided by Situate and the y-axis shows the probability that the observed difference was actually a result of chance (i.e., the null hypothesis that Situate is not better than the control).
I found that both of the Situate agents (with and without early stopping) are performing significantly better than the control. The difference in average IOU values between the Situate agent and the control for dog-walker, dog, and leash objects that were supported at the .05 confidence level were 0.2, 0.13, and .08, respectively. The difference in performance between the Situate agents with and without the stopping condition is approximately .05 at the 95% confidence level for the dog and dog-walker objects, but is not supported for the leash object.

Figure 2.4 shows the localization results for the initial version of the Situate agent, the agent with the early stopping condition, the control, and the Faster-RCNN method. Figure 2.5 shows the significance of the differences between variants of the Situate agent.

- The Faster-RCNN method is substantially better at localizing several of the situation objects than the Situate agent.

- Comparing the Situate agent to the sampling-based control method, we can see that the inclusion of the situation model improves individual object localization for all object types. With a 95% confidence, the effect sizes, in terms of expected IOU, were approximately .1 for the dog and leash objects, and .2 for the dog-

walker object.

- The leash object was far more difficult for each method to localize, and was a clear limiting factor for the overall situation detection quality.

- Situate agents with an early stopping condition still perform better than the control, but the quality degrades substantially when compared to Situate agents allowed to run for a higher number of iterations. Stopping early did hurt performance. However, the mean number of evaluations made by the agent with the stopping condition was 174.6, substantially fewer than the 300 evaluation steps made by the agent without the early stopping condition.

  Also of note (although not visible in the figures) is that the Situate agents without the early stopping condition continued to make changes to their Workspace up until the 300 iteration limit. This suggests that there might be a benefit to increasing the number of iterations available, but also suggests that the agent is taking a long time to settle on a stable Workspace and that no longer making updates to the Workspace may not be a reliable indicator that the agent is finished with its work. I'll touch on this issue in the next chapter where I work to improve the agent.

The Faster-RCNN method outperformed the Situate agent in localizing several of the object types. Factors that may have contributed to this include:

- Faster-RCNN evaluates all image regions. When a Situate agent identifies a region that is sufficiently similar to the object of interest, the system narrows its attention. This may lead to some regions remaining unsearched.

- Faster-RCNN returns the best scoring region even if that region is identified as a poor representation of the object of interest. Situate only adds regions to

a workspace if they exceed a confidence threshold, both with respect internal and total support. In the positive-only evaluation used here, this may benefit Faster-RCNN, but may also lead to a higher false positive rate in other settings. This will indeed be the case in later analyses.

- Faster-RCNN uses *bounding box regression*, a method used to refine bounding boxes using intermediate features from its convolutional neural network. Situate relies on its iterative sampling method to find similar bounding boxes to those in the Workspace. Although bounding box IOU scores and classifier scores are reasonably assumed to be positively correlated in a general sense, the variance in that relationship is not clear, making the reliance on sampling to improve bounding box quality suspect.

The uniform sampling control method did not perform as well as I expected. I assumed that, given many iterations, reasonable bounding boxes would be found in the image for each object of interest. That did not appear to be the case, and suggested a few possible problems.

- Many more iterations may be required for uniform sampling to produce good localization.
- The thresholds on classifier scores may have been set too high, leading to false rejections.
- The fall-back behavior for Situate should evaluate all regions of the image. Relying on repeated sampling is a poor way of doing that.

2.2.1 Sources of improvement

Figure 2.5 shows that the situation model contributed to higher detection quality over the control. However, it is not clear what, specifically, about the situation model was

accountable for the improvement. It may have been a) the more tuned box parameter model, b) the focus on specific regions of the image after conditioning occurs, or c) the contribution of external support to classifying bounding box proposals.

To clarify how the situation model led to improved results, I ran an additional experiment with a variety of lesioned versions of the Situate agent.

Situate and Control remain as described above.

Situate - internal support only functioned as described above, but did not use *external support* in the *total support* calculation. The situation model was still used to direct sampling locations, and was conditioned on Workspace entries as usual.

Situate - no conditioning used only the marginalized version of the situation model. The box parameters and locations were based on the training data, but when an object was added to the Workspace, the distributions did not change.

Figure 2.6 shows results from the modified Situate agents. The agent from *Situate - no conditioning* used the modeled box parameters from the training data, including size, shape, and location, but does not update based on objects added to the Workspace. Figure 2.7 shows that this version performed significantly better than the control for each object type, with effect sizes at the 95% confidence level of approximately .04, .09, and .17 for the leash, dog, and dog-walker objects, respectively. This demonstrates the significant benefit to sampling from a tuned distribution over the control.

Adding the situation model back in, but only using internal support for classification, produces *Situate - internal support only.* Figure 2.7 shows that this version was no better at detecting the dog-walker object than *Situate - no conditioning.* Digging

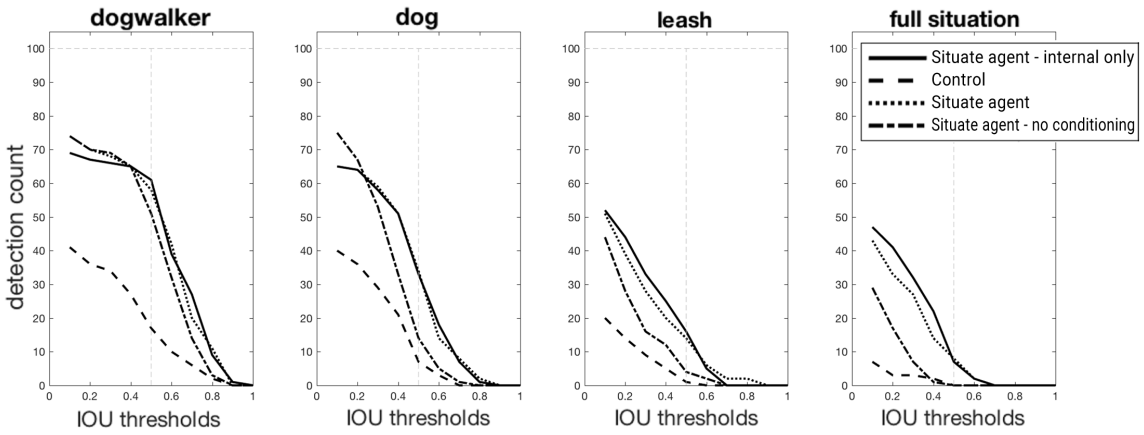


Figure 2.6: These versions of the Situate agent were run for 300 iterations each and differed with respect to how the situation model was utilized. *Situate* and *control* are as they were in the previous experiment. *Situate - internal support only* used a total support function that only uses the classifier and not the external support. *Situate - no conditioning* used the situation model to generate boxes with parameters based on the training data, but the relationships between objects had no influence.

into the sequence of detections a bit, I found that the dog-walker was the most common object to be detected first by a wide margin, which means that the dog-walker object was usually detected prior to conditioning. Figure 2.7 shows that there may be a benefit for the dog object (although it is just shy of statistical significance). For the leash object, the benefit was significant, although the effect size was small.

There was no significant difference between the *Situate* agent and the lesioned *Situate - internal support only* agent. This indicates that the external support value was not making a meaningful contribution from a classification perspective. Example workspaces in figure 3.1c suggest why this may have been. The scaling of the external support function led to a very limited range of values that rarely would have been the deciding factor in meeting the threshold for admission to the Workspace.

Taken together, it appears that the situation model's contributions came from the modeling of bounding box parameters and from the conditioning of the situation model for finding some objects, but that there was not a clear benefit associated with

the external support value used by the individual Situate agent to make classification decisions.

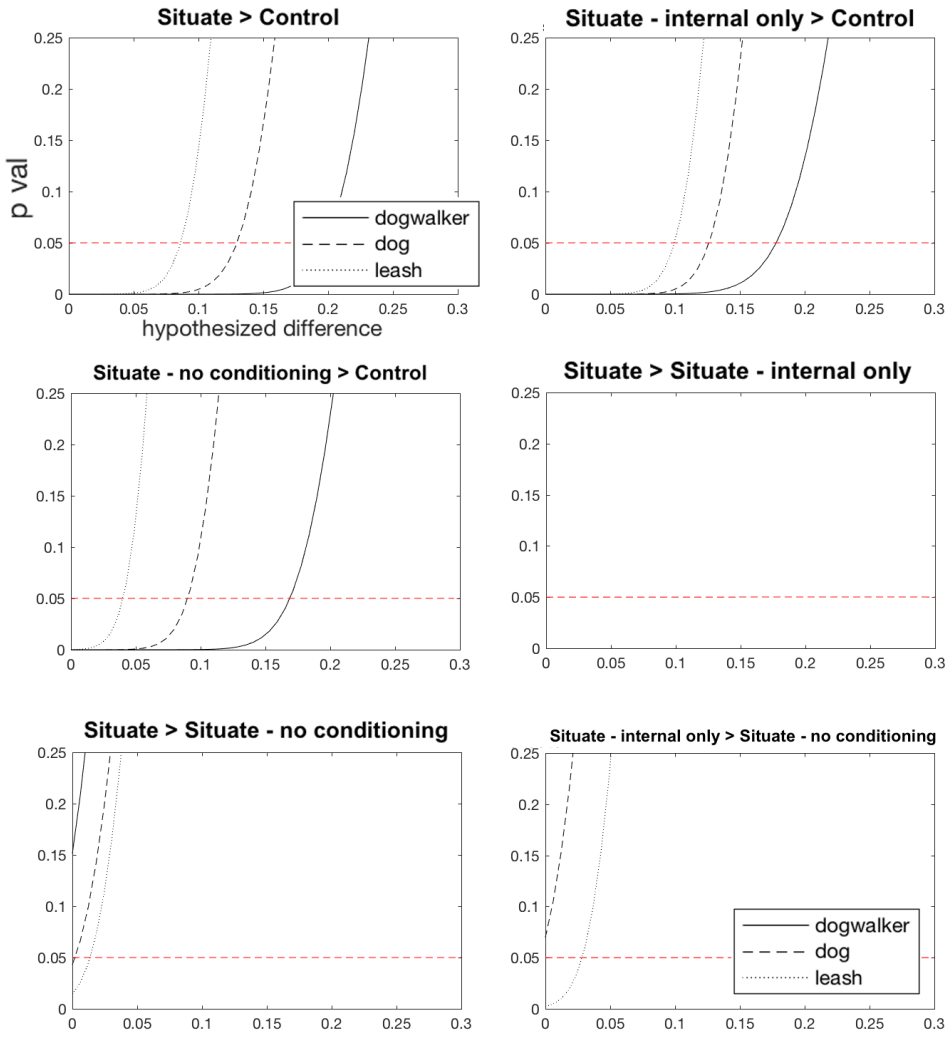


Figure 2.7: These plots show the significance levels of differences in the mean intersection over union scores between methods for several objects of interest. I used the 1-sided t-test to compare Situate and each lesion against the control, as well as several additional comparisons. Situate and each lesioned version performed significantly better than the control, but the effect size was small for the lesion with no conditioning from the situation model. The test between *Situate* and *Situate - internal support only* showed no significant drop in performance for any object type. The bottom row, middle and right column plots show us that removing the situation structure does not apparently hurt detection of the dog-walker, is on the cusp of significance for the dog, and is significant for the leash object.

Chapter 3

Improving the Situate Agent

In this chapter I continue to focus on the individual Situate agent and discuss improvements made to most of its core components, including: the visual classifier at the core of the agent, the method by which bounding boxes are iteratively improved to more accurately localize situation objects, and the support functions that are used to integrate visual and contextual information.

In the previous chapter, there was evidence supporting the hypothesis that the situation model appropriately directed the Situate agent toward regions likely to contain objects of interest, but it also demonstrated that the exhaustive method of searching for situation objects using Faster-RCNN was still more effective than the single Situate agent, even at localizing an object that should benefit from contextual information. However, reviewing the final Workspaces generated by a Situate agent revealed common failure modes, modes which motivated most of the changes described in the rest of this chapter. At the end of this chapter, I will compare the initial Situate agent, as described in the previous chapter and henceforth referred to as Situate agent v1, to the improved Situate agent, described in this chapter and referred to as Situate agent v2, and to the Faster-RCNN method.

3.1 Classifier improvements

Figure 3.1 shows a number of final Workspaces, generated by the Situate agent v1, that failed to fully localize the objects of the situation in a variety of instructive ways.

Those Workspaces demonstrate how relying on the Situate agent's iterative sampling procedure to detect objects, improve bounding boxes, and select from multiple possible objects of a situation was unreasonable and should have some more explicit procedures involved. In this section I expand on the Situate agents' performance of the following tasks:

- estimating the true IOU value (IOU_{gt}) of a bounding box using regression rather than using binary classification to predict whether the IOU is greater than .5
- improving object localization quality by using an explicit method for making adjustments, and
- selecting the correct objects of interest in an image by improving the external support function.

When implementing the Situate agent v1, I made several assumptions about the iterative sampling procedure's ability to detect objects and improve bounding boxes. Those assumptions included:

- An increase in classifier score would lead to an increase in the actual intersection over union between a proposed bounding box for an object and the ground truth bounding box. That is, $f(y) > f(x) \implies IOU_{gt}(y) > IOU_{gt}(x)$ for bounding box proposals x and y and image classifier function f.
- A bounding box with a high density with respect to the situation model would be more likely to be a correct localization than a bounding box with a low density. That is,

$$d(y) > d(x) \implies p(IOU_{gt}(y) > .5) > p(IOU_{gt}(x) > .5)$$

for probability density function d.

Figure 3.1: Each of these Workspaces were generated by Situate agents for images that contained the dog-walking situation. To the right of each figure are the support values associated with each constituent object (internal, external, and total), as well as the ground truth IOU score for each bounding box (gt). Several categories of error are apparent.

- In figures 3.1a and 3.1b the classifiers have high confidence in the localizations, but the objects are only roughly localized. Depending on how noisy the classifier scores are, there may be little room for bounding boxes with higher IOU scores to produce meaningfully higher classifier scores.

- Figure 3.1c shows a failure to select the correct instance of an object. The external support function should bias the agent toward selecting a person near the dog, but the external support value is very high, suggesting a problem with scaling the score.

- The Workspace in figure 3.1d shows an example of an unreliable classifier. Diagonal lines or edges are commonly confused for the leash object, which is a tendency that Situate should account for.

(Best viewed in color)

- The sampling space of potential bounding boxes would include all possible bounding boxes, meaning that, eventually, the correct objects of the situation would win out over incorrect objects of the situation.

Given these assumptions, the method by which the Situate agent v1 would build a situation grounding in its Workspace was as follows:

- Starting from an empty Workspace, a bounding box proposal produces a classifier output that is sufficient to add an object a to the Workspace.
- The situation model is updated for other objects (say, b and c), but not updated for object a.
- Once objects b or c (or both) are added to the Workspace, the situation model for object a is updated, focusing the agent's attention on to a narrow parameter space for object a, leading the agent to sample a better parameterization for the bounding box for object a, which would replace it in the Workspace.

The replacement of parameters for object a relied on $f(y) > f(x) \implies IOU_{gt}(y) > IOU_{gt}(x)$.

Recall that my selection for the classifier in the Situate agent v1 was a combination of a neural network for feature extraction and an SVM for classification. To train the SVM, image regions for the dog-walker class, as well as *distractor regions* that did not contain the dog-walker, were extracted from 300 images from the Portland Simple Dog-walking dataset. This produced 300 positive crops and 1200 distractor crops. Image features were extracted from those image regions using the previously discussed convolutional neural network [4]. A standard linear support vector machine (SVM) was trained to separate the positive feature vectors from the distractor vectors. I used *Platt scaling* to convert the SVM margins to probabilities [27]. The resulting classifier was able to distinguish crops containing dog-walkers from distractor crops

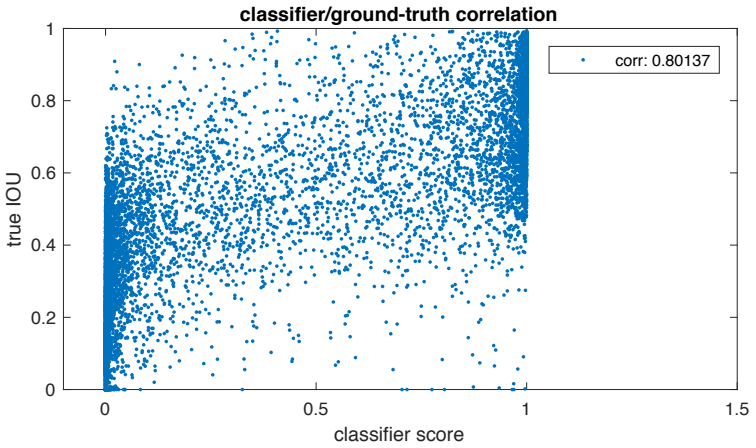


Figure 3.2: The output of the SVM classifier correlated with the ground-truth IOU values of bounding boxes fairly well, but the scores tend to be either close to zero or one. This creates a "signal to noise" problem when trying to determine if an increase in classifier output score is likely indicative of an increase in IOU score.

taken from a 100 image validation set with an area under the ROC curve of .94, which is to say, the neural network and classifier combination was able to perform the discrimination task with few errors.

In order to see how well the trained SVM classifier could predict IOU scores, I applied the classifier to image regions with intermediate IOU values. These crops were extracted from validation images by varying the shape, size, and location of ground truth boxes. This set of bounding boxes was then sub-sampled to create a set with uniformly distributed ground truth IOU scores. I applied the trained classifier to these crops. The correlation coefficient between the ground truth IOU score and the classifier output was $\approx .80$. Figure 3.2 shows a scatter plot with the IOU scores and the classifier scores for each crop from the validation images. Although the trend is clear, it also shows that small changes in the classifier output do not reliably indicate a corresponding change in the IOU score, suggesting that while $f(y) > f(x) \implies IOU_{gt}(y) > IOU_{gt}(x)$ is true in the aggregate, it may not be reliably true for any particular y and x.

Next, to get a sense of how reliable the assumption is that $f(y) > f(x) \implies IOU_{gt}(y) > IOU_{gt}(x)$ for any single instance of an object, I took a large number of bounding boxes around a human in a single image from the validation set. I constructed a set of bounding boxes X such that the entries were of uniform ground truth IOU. Then, for each individual $x \in X$, I found the probability that another crop $y \in X$ with a higher classifier score also has a higher ground truth score. That is, $P(IOU_{gt}(y) > IOU_{gt}(x) | f(y) > f(x))$ for each x in X and for all $y \in (X \setminus x)$. Figure 3.3(a) shows a scatter plot of the ground truth IOU for each x and its associated probability $P(IOU_{gt}(y) > IOU_{gt}(x) | f(y) > f(x))$.

Averaging the probability of improvement for xs with similar classifier scores (shown in red in the figure) initially suggested that improvement was quite reliable, but this obfuscated a problem. Given a reasonably good localization of an object, there are many bounding boxes that are significantly worse than the current localization, a few that are a minor improvement, and vanishingly few are a significant improvement. For instance, for a number of arbitrarily selected individual images in our training set, a correctly sized and shaped bounding box for the dog-walker, evaluated in uniform steps over the image, has 0 IOU in 85% of cases and an IOU less than .25 in about 95% of cases. This means that, when calculating the probability of improvement in figure 3.3(a), this distribution was biased toward large improvements.

Small, iterative adjustments provide a more reliable improvement. Figures 3.3(b) and (c) show the probability of an increase in IOU given a small increases in the classifier score. The samples in figure 3.3(b) use score increase limited to within .1 (of the unitless IOU scores). The plot shows that the probability of an increase in IOU is relatively low (usually below .55). When considering score increases within .2 of a proposal (figure 3.3(c)), the probability of improvement increased a small amount.

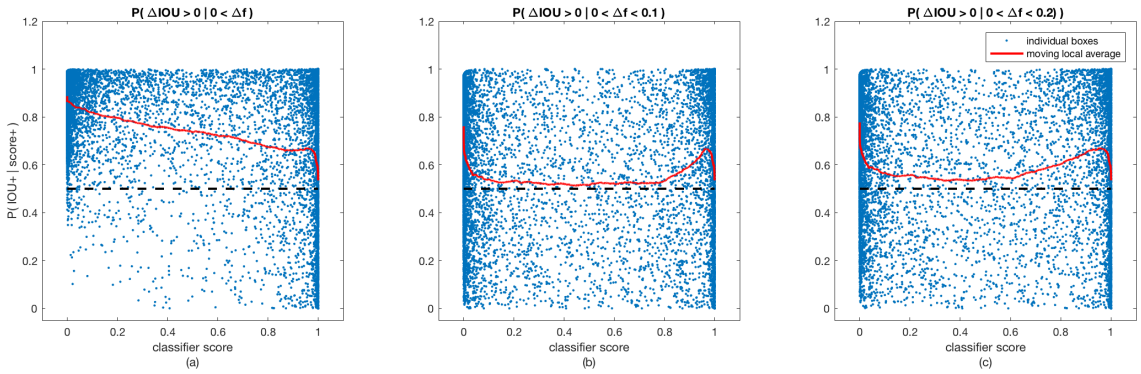


Figure 3.3: The probability that a new bounding box with a higher SVM classifier score actually has a higher IOU value depends on the difference in those scores. (a) shows a scatter plot of the SVM classification score versus the probability that a bounding box with a higher classification score has higher ground truth IOU score, i.e., $P(\Delta IOU > 0 | \Delta f > 0)$. This plot assumes that the expected IOU score of a resampled bounding box is uniformly distributed over $[0, 1]$. Under this assumption, resampling has a good chance of finding a bounding box with a higher estimated IOU. In reality, most bounding boxes will be worse than the current bounding box, and differences in the estimated IOU will be small. For example, if we have a bounding box with an estimated IOU of .2, it is doubtful that our first resampled bounding box will have an IOU of .9. It is more likely that it will have an estimated IOU of .1 or .3. Subfigures (b) and (c) show us the likelihood of an improved IOU score given changes in classifier score less than .1 and less than .2, respectively. In these cases, the probability of improvement in IOU is low or negligible. (Best viewed in color)

Figure 3.4 considers the same issue from the perspective of the expected value of the change in IOU given the same changes in the classifier scores (rather than the probability of an improvement, irrespective of magnitude). The expected change was near zero for even moderate changes in classifier output.

These figures demonstrate is that treating binary classifiers as if they are performing regression is foolish. Although the classifiers performed well on the discrimination task and the correlation between classifier output and IOU was reasonable, using the classifier output to rank individual bounding boxes was a misapplication of the SVM, and that $f(y) > f(x) \implies IOU_{gt}(y) > IOU_{gt}(x)$ is not correct for many x and y.

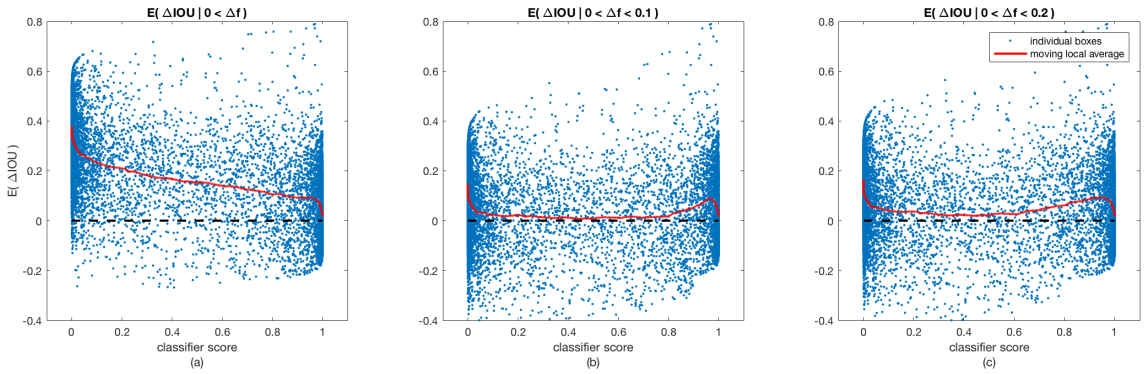


Figure 3.4: Similar to what we saw in figure 3.3, the expected value of replacing a bounding box with a bounding box with a higher SVM score produces, on average, a small increase in actual IOU. (Best viewed in color)

3.1.1 IOU regression

Because Situate agents are using classifier scores for direct quality comparisons, it would be wise to use something that explicitly estimates the metric of interest. For the Situate agent v2, I replaced the SVM classifier with a ridge regression model that estimates the IOU of a bounding box proposal with the underlying object of interest. To do this, I used a more robust training set that included boxes at all IOU values (specifically, the same box generating method used to evaluate the SVM response to intermediate IOU scores). The true IOU scores were used as the target output, rather than the binary indicator of whether or not the IOU score was greater than .5

With this method, the correlation between the IOU estimate and the actual IOU for crops from the validation set was .85 (versus .8 with the SVM). To evaluate the degree to which $f(y) > f(x) \implies IOU_{gt}(y) > IOU_{gt}(x)$ is a reliable claim with the new model, I applied the same analysis as was used to evaluate the SVM. The scatter plots in figure 3.5, again, show the classifier score (now an explicit estimate of the IOU score) plotted against the probability that an image region that generated a higher score also had a higher actual IOU score. Figure 3.5(a) used all bounding boxes with a higher estimated IOU than the bounding box of interest to calculate the

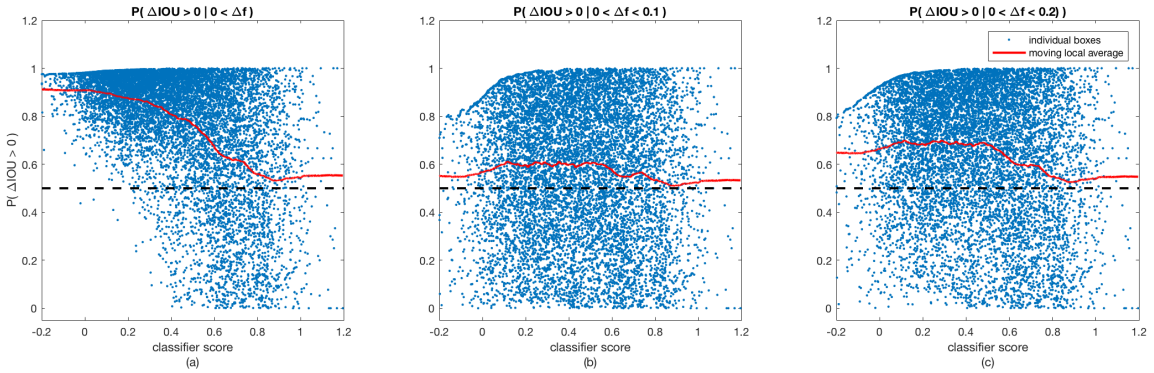


Figure 3.5: Replacing the SVM with the IOU estimator produces relatively similar results to using the SVM, but with generally higher probability of an improvement given the same difference in scores. Some of this effect may be related to the challenge of scaling outputs from the SVM classifier. (Best viewed in color)

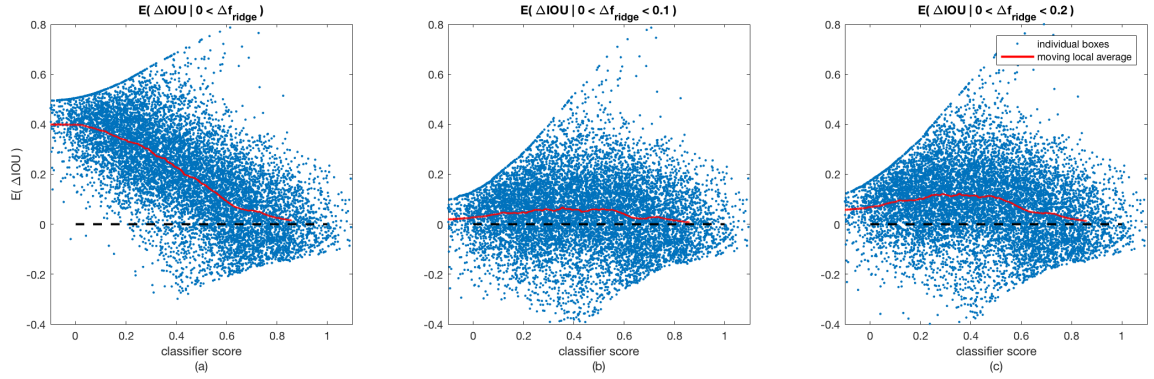


Figure 3.6: The IOU estimator is also associated with a higher expected change in ground truth IOU given the same increase in scores. Again, some of this effect is related to scaling, but in practice, Situate will be making replacement decisions based on differences in scores, so more consistency in the relationship between classifier score and ground truth IOU should make that easier. (Best viewed in color)

probability. Figures 3.5(b) and (c) used bounding boxes with estimated changes in IOU between 0 and .1 and between 0 and .2 greater than an existing bounding box to calculate the probabilities of improvement. Figures 3.5(b) and (c) showed that, for bounding boxes estimated to have had a low to moderate initial IOU score (between .1 and .6 or so), a higher estimated IOU had between a .6 and .7 chance of leading to a higher actual IOU. Figure 3.5 showed the expected value of the change in IOU.

3.1.2 Bounding box regression

The Situate agent v1 used repeated sampling to improve the localization quality of detected objects. This approach was appealing largely based on its simplicity, but adding some complexity to the agent behavior improves localization quality significantly.

After the Situate agent v2 has applied the CNN to an image region, the output of the CNN is used to estimate the IOU of its bounding box with the underlying object (as described above). That output of the CNN (specifically, the network activations of the layer immediately prior to the classification layer) remains useful in estimating a correction to the bounding box that will more accurately enclose the object in the image. This process, called *Bounding box regression*, uses the output of the CNN with the current shape and location of a bounding box to predict the parameters of the correct bounding box. It does this by using four independent regressions from the intermediate representation from the CNN and bounding box parameters to the following values:

$$\Delta x = \frac{B_x - \hat{B}_x}{\hat{B}_w}$$

$$\Delta y = \frac{B_y - \hat{B}_y}{\hat{B}_h}$$

$$\Delta w = log(B_w/\hat{B}_w)$$

$$\Delta h = log(B_h/\hat{B}_h)$$

where $B = (x, y, w, h)$ is a bounding box, x, y are the coordinates of the center of the bounding box, and w, h are its width and height. The terms express the differences between the current bounding box $\hat{B}$ and the ground truth box B. The differences between center points are scaled by the bounding box dimensions to keep

each term independent of image size, and the ratios are passed through a *log* function to normalize them, making them more amenable to modeling. Again, I used ridge regression to build the individual models. The bounding box regression models are applied to get $\Delta^*x, \Delta^*y, \Delta^*w, \Delta^*h$, a set of estimates that can be used to invert the above functions to get B^* , an approximation of the ground truth bounding box B. That is, B^* is defined as:

$$B^*_x = \Delta^*x \times \hat{B}_w + \hat{B}_x$$

$$B^*_y = \Delta^*y \times \hat{B}_h + \hat{B}_y$$

$$B^*_w = e^{(\Delta^*w)} \times \hat{B}_w$$

$$B^*_h = e^{(\Delta^*h)} \times \hat{B}_h$$

Previous work used this method with training data restricted to bounding boxes with a high ground truth IOU[28]. The authors of Faster-RCNN used training data restricted to bounding boxes with an IOU greater than .6, and was only applied to boxes that already had a high confidence of containing an object of interest. Thus, the adjustments were generally small and the resulting bounding boxes were not re-evaluated by a classifier to update the confidence value of the classification.

The decision to use bounding box regression for only small adjustments in Faster-RCNN, as described in [28], was due to Faster-RCNN's use of a dense covering of all image regions with multiple bounding box sizes and shapes, leading to a high similarity between a ground truth bounding box and its most similar entry in the initial search. However, in Situate, bounding box proposals are generated using a sampling procedure, so the bounding boxes will more often need significant improvements. Noting this, I updated the bounding box regression method to make more substantial adjustments. I built bounding box regression models trained with data

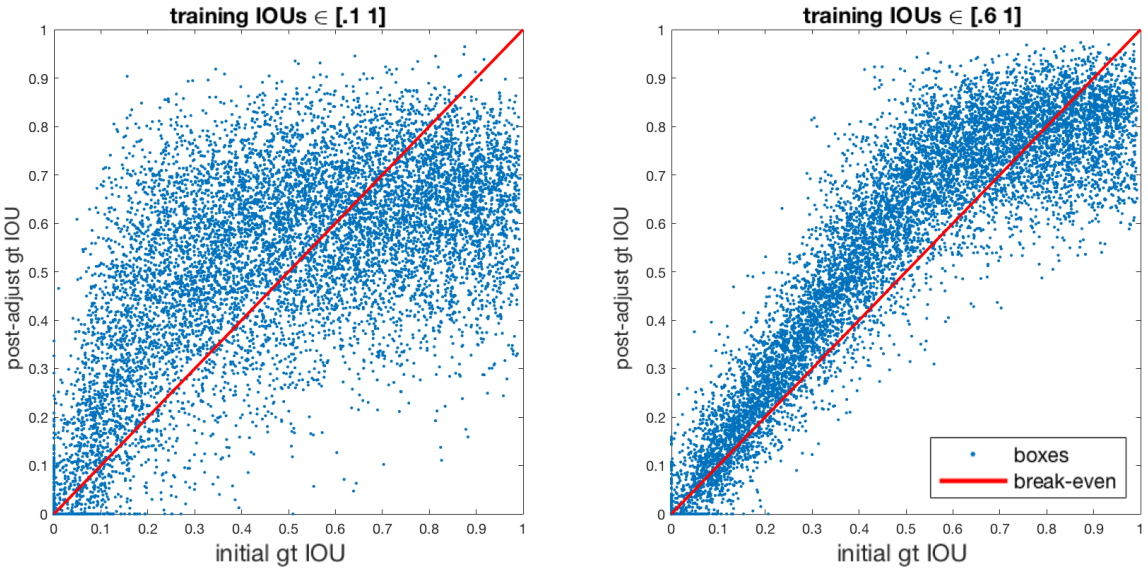


Figure 3.7: Each point represents the ground truth IOU of a bounding box before applying bounding box regression (x-axis) and after (y-axis). Points above the red line represent an improvement in IOU after applying the method. The left plot shows the result of applying the regression after having been trained on bounding boxes with initial IOU scores greater than .1. The right plot shows the results after training the model with bounding boxes with IOU scores greater than .6. The magnitude (and direction) of change is clearly related to the IOU of the input. (Best viewed in color)

from several ranges of IOU values, including IOU values greater than .1 and IOU values greater than .6. Figure 3.7 shows scatter plots of the ground truth IOUs of a set of box proposals and the result of applying the bounding box adjustment procedure described above.

The left plot in figure 3.7 shows changes in the quality of bounding boxes using the bounding box regression system trained with crops of uniform IOU from .1 to 1. It shows that quite a few low-IOU proposals were greatly improved by applying the procedure, as well as a majority of high-IOU proposals that were made substantially worse. The right plot shows results from the system after being trained with bounding boxes from the narrower range of IOUs from .6 to 1. The associated improvements were modest, but the adjustments rarely led to a reduction in localization quality. Figure 3.8 shows the probability of improvement and the expected value of the change

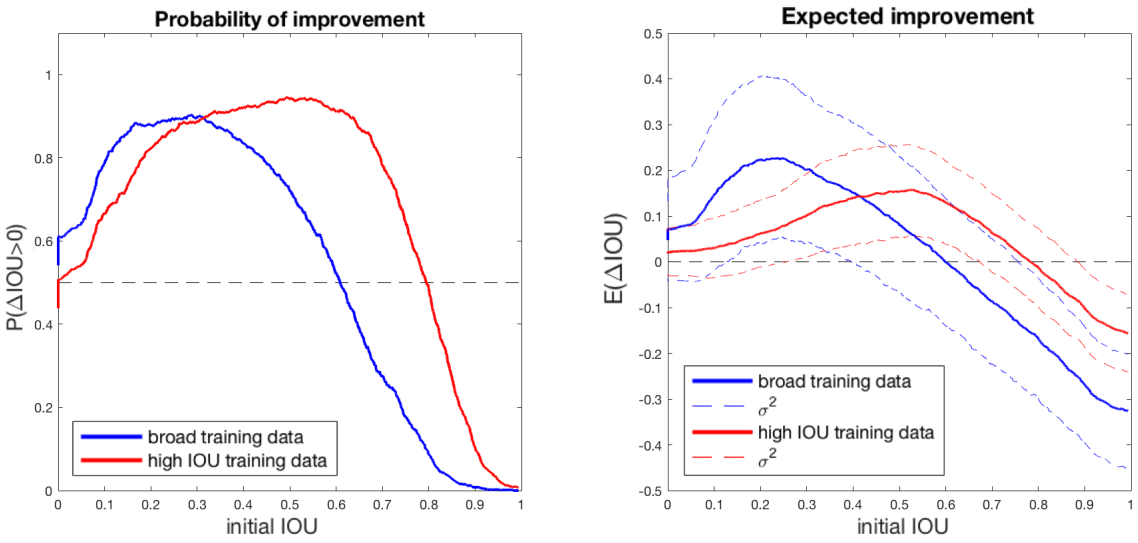


Figure 3.8: The left figure shows the probability of an improved IOU score when bounding box regression is applied to bounding boxes with different initial IOU scores. The right figure shows the expected value of the change in IOU after applying bounding box regression. Bounding box regression models were trained with broad training data (with initial IOU scores ranging from .1 up to 1) and with narrower training data (with initial IOU scores ranging from .6 to 1). Although the probability of improving an initially poor bounding box is similar for using both training methods, the expected value is much greater when using the model trained on a broader range of training data. When applied to already good bounding boxes, the method with the broad training set resulted in a poor outcomes. (Best viewed in color)

in IOU for both models.

The model trained on a broad collection of training images provided a more substantial benefit when applied to bounding boxes with a low IOU, and the more narrowly trained model provided a more substantial benefit when applied to higher IOU bounding boxes. Conveniently, our IOU estimator does a good job of identifying which category a bounding box proposal falls into, so I constructed a combined system that applies each model selectively.

Figure 3.9 shows the result of selectively applying one of the bounding box regression models based on the estimated IOU of the input. I refer to the resulting model as the *bounding box adjustment two-tone model*. The threshold at which the models are selected is based on the optimal threshold for the training data. If the

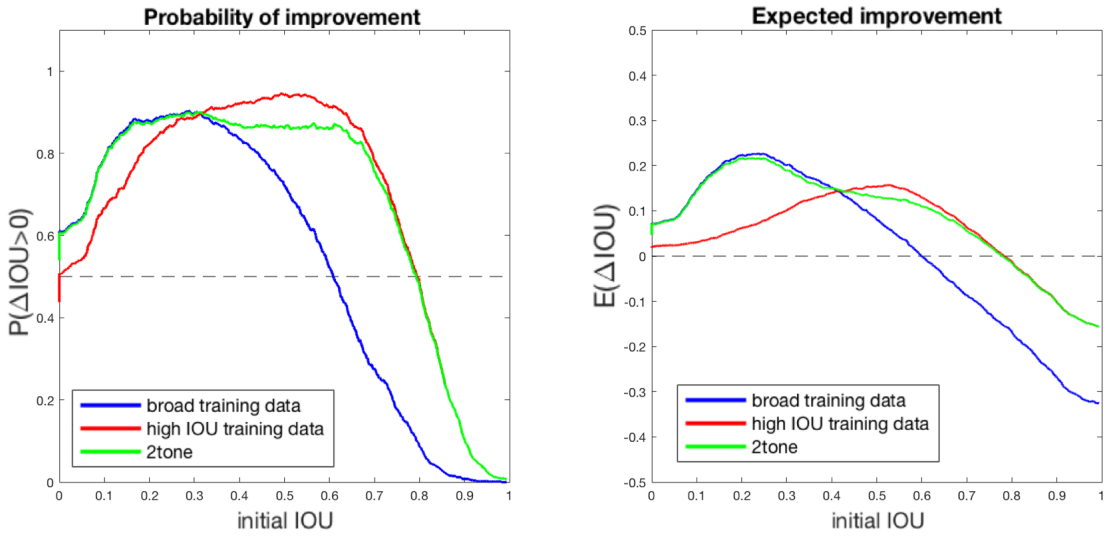


Figure 3.9: The hybrid method of applying bounding box regression, where one of the two bounding box regression models is used based on the estimated IOU of the initial bounding box, performs similarly to the better of the two individual models at each starting IOU value. The difference results from the error in estimation of the actual initial IOU value, leading to the occasional selection of the sub-optimal bounding box regression model. (Best viewed in color)

two-tone model was able to correctly identify whether the object had been localized with an IOU above or below the threshold, the performance of the two-toned model would track the maximum performance of the two sub-model exactly. The two-toned model's performance is slightly below this only due to the occasional over or under estimate of the IOU of bounding boxes that are near the threshold. However, the two-toned model has a higher expected improvement per evaluation than either individual model.

The bounding box adjustment logic is activated by the Situate agent v2 when a sample has an internal support value over a fixed threshold. The bounding box adjustment model is applied to the qualifying bounding box and the resulting bounding box is re-evaluated by the Situate agent.

Initial experiments with this bounding box adjustment logic showed an occasional looping behavior, where a bounding box would satisfy the adjustment threshold,

Situate would generate an updated box, the updated box would have little or no improvement in estimated IOU, but would again satisfy the bounding box adjustment logic and lead to a sequence of unproductive update attempts. I corrected the behavior by including a generation counter. When bounding box adjustment logic is called, the counter is compared to a fixed limit. If the counter is under the limit, a new bounding box is generated, it inherits the generation count from the origin box and increments it. If it is equal to or greater than the limit, no adjusted bounding box is generated. I set the limit to five generations for all following experiments.

Figure 3.10 shows that bounding box regression produced a clear improvement in localization for the dog-walker and dog objects. The IOU regression classifier seemed to do little to improve over the SVM classifier on its own, but IOU regression, in conjunction with the bounding box regression system, may have had a minor benefit over the SVM with bounding box regression. There was little to no improvement on leash objects for any method, suggesting that tuning the bounding box quality was not the primary issue for that object type. Figures 3.11 and 3.12 show paired t-tests that compare the result of running a Situate agent with the SVM classifier and the IOU estimator, with and without bounding box regression.

The SVM classifier and the IOU estimator both improve on the SVM classifier alone to a significant degree for the dog and dog-walker objects. The difference in quality between the IOU estimator with bounding box adjustment and the SVM classifier with bounding box adjustment are less clear. I used the combination of the IOU estimator and bounding box regression system for subsequent experiments.

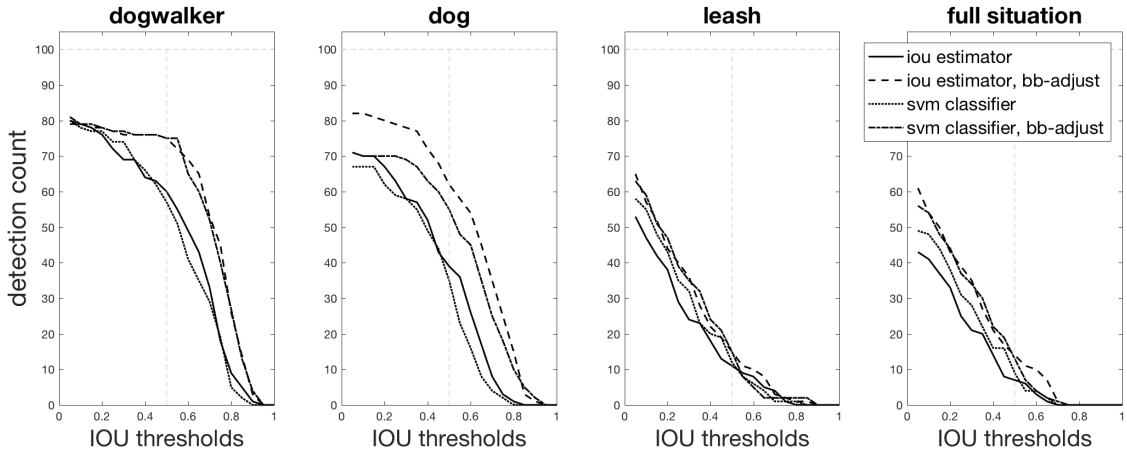


Figure 3.10: The above curves show the object localization quality for a Situate agent using the SVM classifier, the IOU estimator, and each when bounding box regression was included.

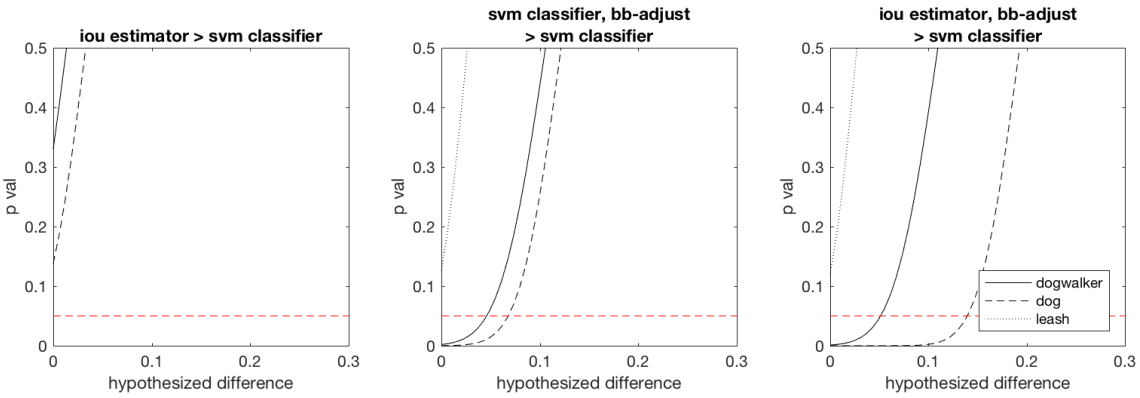


Figure 3.11: The above curves show the significance of differences between combinations of classifiers and the bounding box adjustment method at different hypothesized differences in the ground truth IOU of localized objects using paired t-tests. These three curves show the significance of improvements over the SVM classifier used in the Situate agent v1. The IOU estimator alone does not have a significant improvement for any of the object types. The SVM classifier with bounding box adjustments improves over the SVM alone to a small degree for the dog and dog-walker objects, but not for the leash. The IOU estimator with bounding box adjustment improves over the SVM for the dog-walker object, and do a greater degree for the dog object.

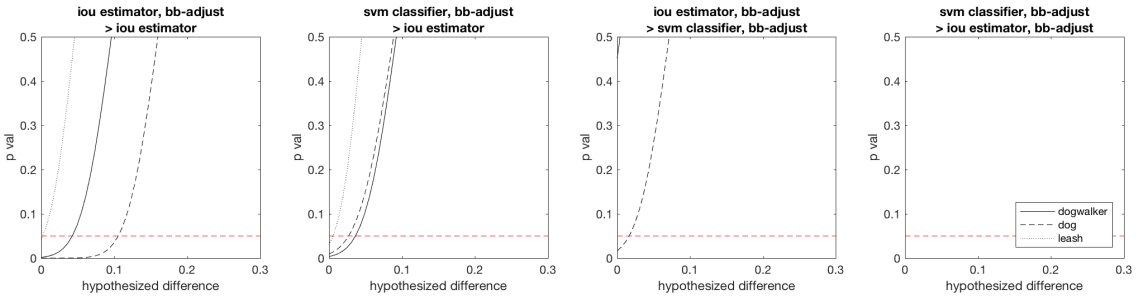


Figure 3.12: The above curves show the significance of differences between select combinations of the IOU estimator and the SVM classifier with and without bounding box adjustment. The IOU estimator with bounding box adjustment improves most substantially over the IOU estimator alone for the dog object. The SVM with bounding box adjustment improves on the IOU estimator alone for each object type, but for very small hypothesized differences. The IOU estimator with bounding box adjustment improves on the SVM classifier with bounding box adjustment with low confidence for only the dog object.

3.2 Support functions

Previously I described Situate's internal support function, which scales the output of its classifiers to estimate the quality of proposed bounding boxes, and external support function, which gives a single value that expresses how compatible proposed bounding boxes are with Situate's model of the situation. Having support functions that produce interpretable values helps a user diagnose possible mistakes being made by agents and lets agents prioritize the correct objects to include in Workspaces. To these ends, the support functions should do the following:

- apply useful scaling to the values that come from the classifier,
- emphasize the selection of the correct objects in the Workspace, and
- integrate this information into a single, interpretable value.

Considering the reliability of classifiers and trust in general makes it natural to consider the relevant Situate functions from a Bayesian perspective. It may clarify the

specific differences between Situate and Bayesian networks, as well as providing some scaffolding upon which to improve the internal and external support functions used by the Situate agent v1. Defining some terms will help reasoning through the situation recognition problem:

- b_x are the parameters for a bounding box with object label x.
- c_x is the output of the classifier trained to detect object x applied to the region defined by b_x.
- l_x is the proposition that the bounding box b_x has localized its object of interest with IOU greater than .5.
- b_y, c_y, and l_y are similarly defined for object y.

Several important correspondences are relatively clear. For a situation consisting of objects x and y:

$$\text{Situation score (for a situation consisting of objects } x, y) \sim p(l_x, l_y | c_x, c_y, b_x, b_y)$$

$$\text{Total support for object } x \sim p(l_x | b_x, c_x, l_y, b_y, c_y).$$

That is, the notion of *situation score* is similar to the probability that objects x and y have been successfully localized given the parameters of their bounding boxes and the associated classifier scores. *Total support* (for an object in the Workspace) is similar to the probability of the correct localization of an object given the rest of the information in the Workspace, and assuming that all other included objects are correctly localized.

When working on the support functions for Situate, I made a few assumptions regarding how Situate uses its classifiers, the prior belief that an image contains the

situation of interest, and the space of possible situation groundings for an image. Those assumptions include:

- c_x is conditionally independent of b_x, y, b_y, and c_y given l_x. That is to say, other objects, their bounding box parameters, and other classifier outputs do not influence the output of a classifier applied to an image region. If the presence of the object is accepted, the other parameters are not affected by the visual classifier output.

- For situation detection, priors for propositions l_x and l_y are low, but not vanishingly so. The prior expectations for l_x and l_y are not the prior for the object being present in an image in the data set, but for the probability that an arbitrary bounding box has a sufficient overlap with the ground truth bounding box for its object of interest. From the perspective of an individual agent, the situation is assumed to be present in the image.

- The background distribution of all possible box parameters b_x as approximately uniform with respect to location, aspect ratio, and area ratio. The actual set of all bounding boxes over an image is not quite uniform in this way, but once very tall and very wide boxes are removed, and very small and very large boxes are removed, it is not a bad approximation. The conditional distributions for $p(b_x|l_x)$ are assumed to be and are modeled as normal distributions.

Looking at the total support function for objects in the Workspace, and given the above assumptions, we see that our division of total support into a combination of *internal support* and *external support* comes out naturally.

$$p(l_x|b_x, c_x, l_y, b_y, c_y) =$$

$$= \frac{p(l_x, b_x, c_x, l_y, b_y, c_y)}{p(b_x, c_x, l_y, b_y, c_y)} \quad \text{(by definition)}$$

$$= \frac{p(c_x|l_x, b_x, l_y, b_y, c_y)p(l_x|b_x, l_y, b_y, c_y)p(b_x, l_y, b_y, c_y)}{p(c_x|b_x, l_y, b_y, c_y)p(b_x, l_y, b_y, c_y)} \quad \text{(factoring)}$$

$$= \frac{p(c_x|l_x, b_x, l_y, b_y, c_y)p(l_x|b_x, l_y, b_y, c_y)}{p(c_x|b_x, l_y, b_y, c_y)} \quad \text{(canceling)}$$

$$= \frac{p(c_x|l_x, b_x, l_y, b_y, c_y)p(l_x|b_x, l_y, b_y, c_y)}{p(c_x|l_x, b_x, l_y, b_y, c_y)p(l_x) + p(c_x|\neg l_x, b_x, l_y, b_y, c_y)p(\neg l_x)}$$

(marginalizing over l_x)

$$= \frac{p(c_x|l_x)p(l_x|b_x, l_y, b_y, c_y)}{p(c_x|l_x)p(l_x) + p(c_x|\neg l_x)p(\neg l_x)} \quad \text{(by conditional independence of } c_x \text{ given } l_x\text{)}$$

$$= \frac{p(c_x|l_x)p(l_x|b_x, y, b_y, c_y)}{p(c_x)} \quad \text{(recombining } p(c_x)\text{)}$$

$$= \frac{p(l_x|c_x)p(l_x|b_x, l_y, b_y, c_y)}{p(l_x)} \quad \text{(applying Bayes' rule)}$$

The $p(l_x|b_x, y, b_y)$ term tracks our notion of external support, as does the $p(l_x|c_x)$ term for our notion of internal support.

3.2.1 Internal support

We saw in the evaluation of the Situate agent v1 that detecting the location of the leash in an image was quite a bit more difficult than detecting the dog or dog-walker 2.4. We also saw that the classifier for the leash object was less able to discriminate leash crops fro background crops than the dog-walker and dog classifiers based on the difference in their AUROC values 2.1.1. This indicated to me that it would be worth being explicit about how much we trust the classifier and integrating that level of trust into the behavior of the Situate agent.

Starting from the term $p(l_x|c_x)$, we can rearrange terms a bit to find $p(c_x|\neg l_x)/p(c_x|l_x)$.

$$p(l_x|c_x) = \frac{p(c_x|l_x)p(l_x)}{p(c_x)} \qquad \text{(Bayes' rule)}$$

where

$$p(c_x) = p(c_x|l_x)p(l_x) + p(c_x|\neg l_x)p(\neg l_x). \qquad \text{(marginalizing)}$$

Then

$$p(l_x|c_x) = \frac{p(c_x|l_x)p(l_x)}{p(c_x|l_x)p(l_x) + p(c_x|\neg l_x)p(\neg l_x)}$$

$$= \left(1 + \frac{p(c_x|\neg l_x)}{p(c_x|l_x)}\frac{p(\neg l_x)}{p(l_x)}\right)^{-1}.$$

The term $p(c_x|\neg l_x)/p(c_x|l_x)$, an odds ratio, expresses the relationship between the classifier's output when the target is present and when it is not present. Notably, the term is independent of $p(l_x)$, which means that it is a property of the classifier unrelated to how often the target of classification is present in a particular data set, making it similar to the ROC curve analysis commonly used to evaluate classifiers, rather than a precision/recall analysis.

The AUROC value that summarizes the ROC curve can not be used to perfectly reproduce an ROC curve, but given a few assumptions, such as the normality of distributions of decision values for positive inputs and for negative inputs, and equal variances between those distributions, you can come close. Similarly, the AUROC value can be used to approximate the odds ratio above. The odds ratio can be approximated with a function of the form:

$$p(l_x|c_x) \approx \left(1 + e^{f(c_x, AUROC, log(\frac{p(\neg l_x)}{p(l_x)})}\right)^{-1}.$$

Figure 3.13 shows the log odds and $p(l_x|c_x)$ curves generated by using the odds ratio $p(c_x|\neg l_x)/p(c_x|l_x)$ (with both probabilities based on Normal distributions fitted to data gathered during classifier training) and curves generated using an approximation that uses the AUROC of the classifier. The approximation leads to similar $p(l_x|c_x)$ predictions. The notable exception is that the *dog* and *dog-walker* estimates are estimated to be the same, as they have essentially the same AUROC values, where the $p(l_x|c_x)$ should be higher for dogs than for dog-walkers at the same classifier confidence. I expect that what is being obfuscated here is that the dog-walker classifier is highly activated by positive instances of dog-walkers, but is also activated by the frequent background humans in the negative data, whereas the dog classifier may be less activated by positive instances of dogs and has fewer confusing background image regions. That over-estimate of the likelihood of a localization of a dog-walker should

be corrected in Situate with the use of *external support*.

3.2.2 External support

External support is intended to support two properties of Situate. The first is the selection of the correct object when there are multiple instances of an object involved in a situation. For example, selecting the person walking the dog when there are several people present. The second is to provide supporting information when the classifier is known to be unreliable for a particular object type. For example, the leash object, which appears as a simple line and is often confused with background regions of images, can be identified more reliably given contextual objects.

During the evaluation experiments for Situate agent v1, it was not clear that external support value was accomplishing these goals. Some of the ambiguity came from the scaling of the support values, which were almost always near zero or near one. The original function that scaled external support was parameterized to minimize the error between the external support score and the empirical cumulative distribution function of the density values. I intended for this to create an intuitive relationship between the external support values and the underlying statistics. However, in practice, there is little difference between an external support score in the 10th percentile of external support scores and in the 50th percentile, as all values in this range provide little discriminative information. The support values that did provide discriminative information were very near the 100th percentile.

To get a clearer view of whether or not there was useful information being obfuscated by the poor scaling of the external support scores, I performed an additional, small investigation. I sampled a large number of bounding box proposals for each training image in the Portland Simple Dog-walking dataset using several different sampling methods. The methods included a) a uniform model, b) Situate's per-object

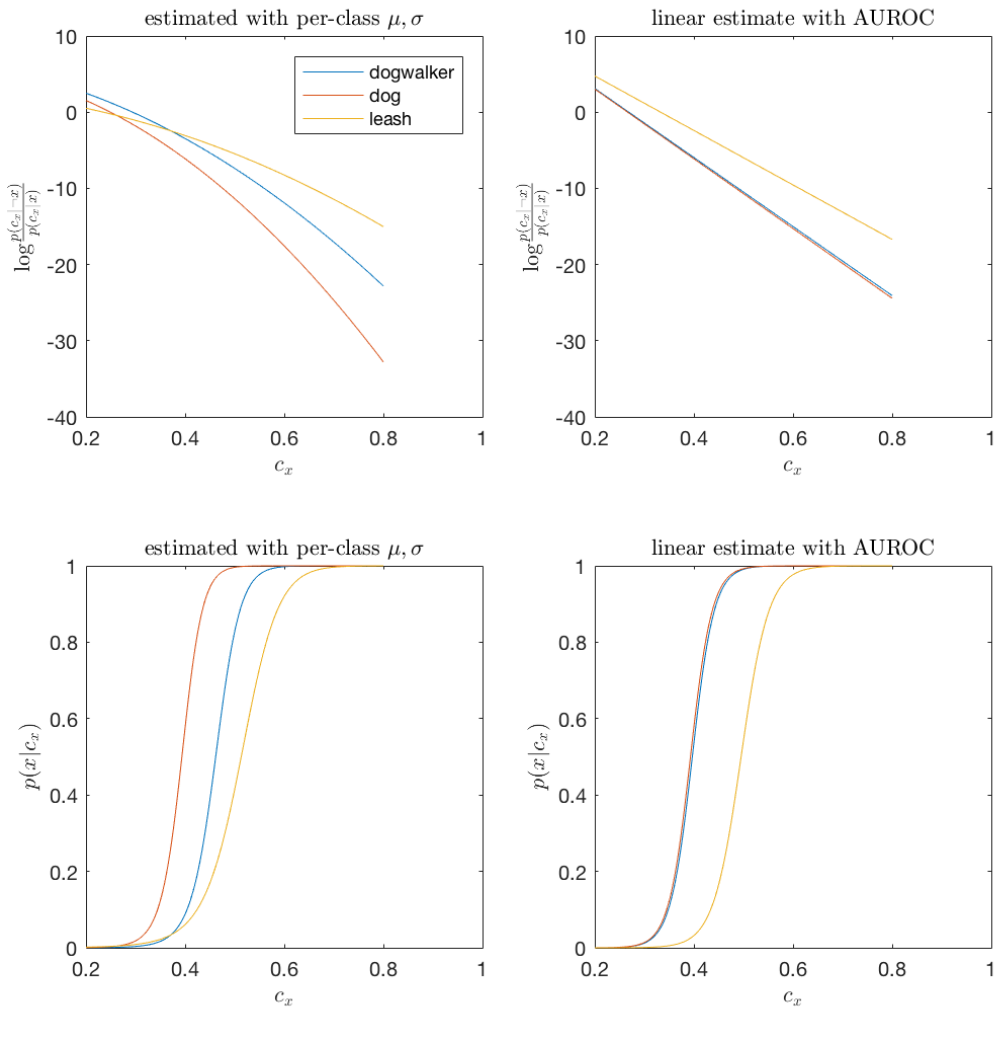


Figure 3.13: The upper left plot shows the log odds ratios for each object class when $p(c_x|l_x)$ and $p(c_x|\neg l_x)$ are modeled as normal distributions using the training data for the respective objects. These ratios produce the probability estimates in the lower left plot. The plot in the upper right quadrant shows the results of estimating the odds ratios as a linear function of the AUROC score for each classifier. The lower right plot shows the result of using the function to estimate probabilities of correct localization. This process shows that these probabilities can be estimated reasonably well using a simple function that takes only the AUROC. (Best viewed in color)

normal model of bounding box parameters, and c) the conditioned situation model using the ground truth for two of the situation objects to condition for the third. For each of those collections of samples, I calculated the density of each sample with respect to the other models. For example, I calculated the densities for each sample drawn from the uniform distribution given the normal model and the conditioned normal model. Table 3.1 shows results from that experiment. From it I drew several conclusions.

- **The situation model provides good discrimination power when we assume uniform density over all possible bounding boxes.** If we suppose a uniform sampling of bounding boxes, where all possible bounding boxes are given equal likelihood, then the density values with respect to our situation models will indeed provide a good signal for identifying bounding boxes that are likely to have a high ground-truth IOU. We gather this from the reasonably good AUROC values in the "uniform sampling, normal density" and "uniform sampling, conditional density" rows of table 3.1.

- **If more information is available to the situation model, the average quality of samples improves.** The goal of Situate is not to process all possible bounding boxes and eliminate the chaff. Situate uses the situation model to focus its sampling such that the samples it does draw are likely to be good. We see that the situation model contributes to accomplishing this goal by comparing the mean IOU and $p(IOU(x) > .5)$ for uniform sampling, normal sampling, and conditioned sampling rows. As the information available to the situation model increases from none, to object specific, to conditioned on other objects, the average IOU increases substantially for each object type. Conveniently, this analysis also gives us concrete values for $p(l_x)$ for each object and sampling method.

- **When the same model is used to draw samples and to score them, there is little discriminatory power.** That is, if Situate is drawing samples from a particular distribution, say, the unconditioned normal distribution for dogs, then scoring those samples with external support based on that same unconditioned normal distribution for dogs does not provide useful information. This, unfortunately, nearly nullifies the benefit of using density as a discriminative signal *(* when the samples being compared are from the same distribution). Rows "normal sampling, normal density" and "conditional sampling, conditional density" show that the AUROC values drop to nearly the guessing rate of .5.

- **When there is a mixture model used in sampling and evaluation, the density has some discriminitive power.** In practice, the distributions used in the situation model change during the evaluation of an image, so the sampling sources are mixed, and the function with which Situate is evaluating external support updates. A rough approximation of this can be seen in row "combined normal and conditional methods", where a mix of sampling sources and density calculations are combined, and the AUROC indicates reasonably good discriminative power. Samples drawn during an actual run of a Situate agent produce a similar distribution.

These statistics demonstrate that there is useful, discriminative information in the density values when the situation model used to evaluate them has more information available to it than was available when the sampling occurred. However, the AUROC scores indicate that the discriminative potential is limited. An alternative scaling mechanism should at least help Situate to utilize the discriminative potential that is there and make the external support scores more interpretable. Toward that end, I looked into the Bayesian correlate of external support, $p(l_x|b_x, l_y, b_y, c_y)$. Before that,

	dog-walker	dog	leash	all
uniform sampling, normal density				
mean IOU	0.051	0.023	0.022	0.032
$p\big(IOU(x) > .5\big)$	0.004	0.002	0.002	0.003
AUROC	0.859	0.781	0.898	0.822
normal sampling, normal density				
mean IOU	0.107	0.040	0.039	0.062
$p\big(IOU(x) > .5\big)$	0.037	0.006	0.003	0.015
AUROC	0.596	0.653	0.641	0.683
normal sampling, conditional density				
AUROC	0.838	0.943	0.934	0.882
conditional sampling, conditional density				
mean IOU	0.320	0.234	0.193	0.249
$p\big(IOU(x) > .5\big)$	0.249	0.088	0.053	0.130
AUROC	0.588	0.636	0.657	0.669
combined normal and conditional methods				
mean IOU	0.146	0.084	0.073	0.101
$p\big(IOU(x) > .5\big)$	0.082	0.025	0.015	0.041
AUROC	0.821	0.855	0.857	0.847

Table 3.1: Several methods of sampling box proposals and scoring them based on their density are compared in the table above. Bounding boxes were sampled using Situate's "uniform" method, its unconditioned normal model, and its conditioned normal model (where the conditioning used ground truth parameters for the other objects in the scene). For example, to generate bounding boxes for the dog object, samples were drawn from the uniform model, the normal distribution over dog bounding box parameters, and for a normal model conditioned with the ground truth parameters for the dog-walker and leash objects in the image. For each sampling method, the average IOU of the samples with the ground truth object of interest were recorded, as was the probability of samples having an IOU greater than .5. The density of each sample was calculated with respect to several distributions. For each sampling method and density calculation pair, the AUROC value was calculated to characterize the pair's capacity to discriminate between bounding boxes with a ground truth IOU above or below the .5 threshold.
There are two notable patterns. First, more complex sampling models lead to higher mean IOU for the sampled boxes, which is to say, the better models do indeed lead to better samples. Second, the AUROC values are high when the complexity of the density model is higher than the complexity of the sampling model, but is low when the density model and sampling model are the same. Together, these suggest that using a more complex model improves the quality of the samples, but that the density of the samples with respect to that model are going to provide minimal discriminative value.

an additional assumption:

- The probability that a particular object has been localized is independent of what other objects have been identified in the image. That is, $p(l_x|l_y) = p(l_x)$. In the past, computer vision systems have used terms like $p(l_x|l_y)$ as a co-occurrence term that tracks how often different objects occur in an image together [11]. Co-occurrence terms have been shown to be useful for correcting object labels, but they are not something currently used in Situate. However, the probabilities of object localizations do change when the *parameterizations* of bounding boxes are considered.

$$
\begin{aligned}
& p(x|b_x, y, b_y, c_y) \\
& \quad = \frac{p(l_x, b_x, l_y, b_y, c_y)}{p(b_x, l_y, b_y, c_y)} && \text{(definition)} \\
& \quad = \frac{p(b_x|l_x, l_y, b_y, c_y)p(l_x, l_y, b_y, c_y)}{p(b_x, l_y, b_y, c_y)} && \text{(factoring)} \\
& \quad = \frac{p(b_x|l_x, l_y, b_y, c_y)p(l_x)p(l_y, b_y, c_y)}{p(b_x, l_y, b_y, c_y)} && \text{(independence of } l_x, l_y\text{)} \\
& \quad = \frac{p(b_x|l_x, l_y, b_y, c_y)p(l_x)p(l_y, b_y, c_y)}{p(b_x|l_y, b_y, c_y)p(l_y, b_y, c_y)} && \text{(factoring)} \\
& \quad = \frac{p(b_x|l_x, l_y, b_y, c_y)p(l_x)}{p(b_x|l_y, b_y, c_y)} && \text{(canceling)} \\
& \quad = \frac{p(b_x|l_x, l_y, b_y, c_y)p(l_x)}{p(b_x|l_x, l_y, b_y, c_y)p(l_x) + p(b_x|\neg l_x, l_y, b_y, c_y)p(\neg l_x)} && \text{(marginalizing over } l_x\text{)} \\
& \quad = \frac{1}{1 + \frac{p(b_x|\neg l_x, l_y, b_y, c_y)}{p(b_x|l_x, l_y, b_y, c_y)} \frac{p(\neg l_x)}{p(l_x)}} && \text{(dividing by } p(b_x|l_x, l_y, b_y, c_y)p(l_x)\text{)} \\
& \quad = (1 + e^{\psi})^{-1}
\end{aligned}
$$

where

$$\psi = log\big(p(b_x|\neg l_x, l_y, b_y, c_y)\big) + log\big(p(\neg l_x)\big) - log\big(p(b_x|l_x, l_y, b_y, c_y)\big) - log\big(p(l_x)\big).$$

This is similar to the scaling function used for the original external support function (a logistic function of $log\big(p(b_x|l_x, l_y, b_y)\big)$), but with different constants, found through a more principled approach, and includes the $p(b_x|\neg l_x, l_y, b_y, c_y)$ term. The $p(b_x|l_y, b_y, c_y)$ term exposes the difficulty in using density values for discrimination in Situate. Marginalizing $p(b_x|l_y, b_y, c_y)$ with respect to l_x produces $p(b_x|\neg l_x, l_y, b_y, c_y)$, which is the distribution of bounding box parameters for boxes that do not contain the object of interest. When actually used in Situate, $p(b_x|\neg l_x, l_y, b_y, c_y)$ is essentially identical to $p(b_x|l_y, b_y, c_y)$ (the sampling distribution), which is itself a model of $p(b_x|l_x, l_y, b_y, c_y)$, as it is trained on bounding boxes that *do* contain the object of interest. With $p(b_x|\neg l_x, l_y, b_y, c_y)$ and $p(b_x|l_x, l_y, b_y, c_y)$ being essentially the same, we end up with no discriminative power at all. This is likely why we saw little discriminative power when we used the same distribution for sampling and for density calculations in Figure 3.1.

However, the external support value does remain important, if for no other reason, because it provides higher scores for proposals that were generated while informed by context over those that were generated without context. To construct a useful scaling method, I used the above probabilistic approach, but assumed a uniform distribution for the negative instances of bounding boxes, which made $p(b_x|\neg l_x, l_y, b_y, c_y)$ a constant, and produced a function much like the original scaling function for external support. Figure 3.14 shows the relationship between predicted $p(l_x|b_x, l_y, b_y)$ and IOU values. Although these estimates are not particularly accurate, it shows that the function has predictive power. Additionally, the scaling is easily interpretable and is similarly scaled to our internal support values, making their combination a bit more straightforward.

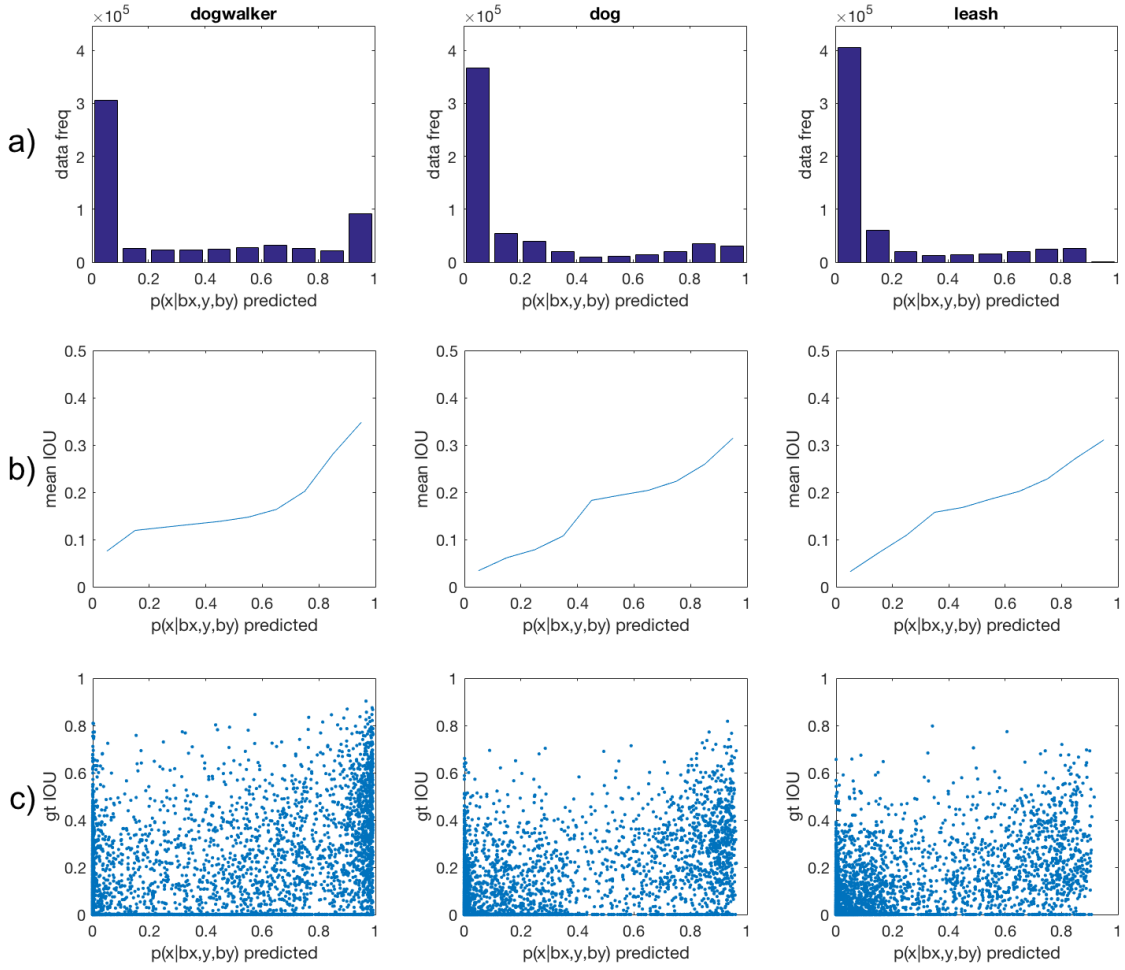


Figure 3.14: (a) The first row of figures shows the distribution of predicted probabilities of IOU greater than .5. Between half and three quarters of samples resulted in 0 external support (depending on the object type). (b) The second row of plots show the relationship between the predicted probabilities and the mean IOU of sampled boxes, grouped by predicted probabilities. The probability predictions themselves are not very accurate (not shown), but the resulting IOU scores are monotonic. The scatter plots in the final row (c) show that these relationships are a trend, and not necessarily trustworthy for each individual comparison.

3.2.3 Combined internal and external support

The combination of the output of the classifier with contextual information produces the total support value, which expresses the degree to which an observation appears to be reliable. In the discussion at the start of this section, I looked at total support from the perspective of the probability of an object being correctly localized given the classifier output and all information available in the Workspace, that is,

$$p(l_x|b_x, c_x, l_y, b_y, c_y) = \frac{p(l_x|c_x)p(l_x|b_x, l_y, b_y, c_y)}{p(l_x)}.$$

In practice, this version of total support is quite incompatible with Situate's Workspace. Once an object has been added to the Workspace, the contextual information becomes profoundly important to the addition of any additional objects. If that initial object is an incorrect detection, it is nearly impossible to correct or for the Workspace to progress. Instead, an additive combination of the internal and external support is preferable, but leaves the issue of which terms to use and how to combine them.

There are a number possible inputs to the total support function, including:

- c_x, the output of the IOU estimator,
- $AUROC_{obj}$, the training AUROC for the IOU estimator for the object of interest,
- $p(l_x|c_x)$ using distribution estimates gathered during training, and
- $p(l_x|b_x, l_y, b_y)$.

In addition to expressing how reliable an observation is, the combination of internal and external support should also accomplish several additional goals, including:

- Total support should be a point of interpretability for post-hoc analysis,

- A classifier with an AUROC score of .5 should be given no weight, as it provides no discriminative power,
- A perfect classifier should still be influenced by external support, as external support determines which objects should be included in a Workspace when there are several instances of an object type,
- There should always be at least one object type for which internal support alone is sufficient to add it to the Workspace without additional contextual information.

To explicitly satisfy the above requirements, I constructed a linear combination of internal support ($S_{int,obj}$) and external support ($S_{ext,obj}$) that uses a mixing term based on the AUROC values of the classifiers for the objects of the situation:

$$S_{total,obj}(S_{int,obj}, S_{ext,obj}, \phi) = w_{ext,obj}(\phi)S_{ext,obj} + w_{int,obj}(\phi)S_{int,obj}$$

where

- $w_{ext,obj}(\phi) = w_{ext,min} + |objects| \times (w_{ext,ave} - w_{ext,min}) \times (1 - \phi_{obj})/(\Sigma_{obj}(1 - \phi_{obj}))$
- $w_{ext,min}$ is minimum external support weight for a situation object,
- $w_{ext,ave}$ is the average external support weight over all situation objects,
- $\phi_{obj} = 2(AUROC_{obj} - .5)$,
- ϕ is a vector of ϕ_{obj} for each object in the situation,
- $w_{int,obj}(\phi) = 1 - w_{ext,obj}(\phi)$,
- $S_{int,obj} = c_{obj}$, the output of the IOU estimator for object obj, and
- $S_{ext,obj} = p(l_x|b_x, l_y, b_y)$ for object x and Workspace objects y.

The ϕ term scales trust in the classifier linearly based on its reliability as measured by the AUROC score. Each object type has a minimum external support weighting and internal support can't become ignored entirely based on a fixed average external support score.

3.3 Evaluation of the updated Situate agent

I evaluated the Situate agent v2 using the same set of evaluation images from the Portland Simple Dog-walking data set used previously when evaluating Situate agent v1. In this experiment, I alloted each agent (Situate agents v1 and v2) a higher number of calls to the image classifier, increasing the limit from 300 to 1,000. This additional resource allocation was meant to let the agents run until they had developed a stable Workspace that did not change substantially. Figure 3.15 shows the relationship between the number of calls to the classifier and the quality of Workspaces generated by Situate agents v1 and v2. The Situate agent v2 developed Workspaces relatively quickly. The Situate agent v1 benefited substantially from the additional computational resources, adding more objects to its Workspace and improving the quality of those localizations, but did so over many more calls to the classifier.

Figure 3.16 shows the grounding results for the Situate agent v1, the Situate agent v2, and the top scoring bounding boxes for each situation object type using Faster-RCNN. The Situate agent v2 detects the correct person with an IOU greater than .5 in approximately 80% of images and localizes the dog at about the same rate, but still fails to sufficiently localize the leash object in more than about 20% of images. This does not represent an improvement over the Situate agent v1 for the person or leash objects, but is a substantial improvement for the dog object.

What mistakes are the agents making that lead them to poor groundings? Figure 3.17 shows examples of groundings that had a high final situation score but had

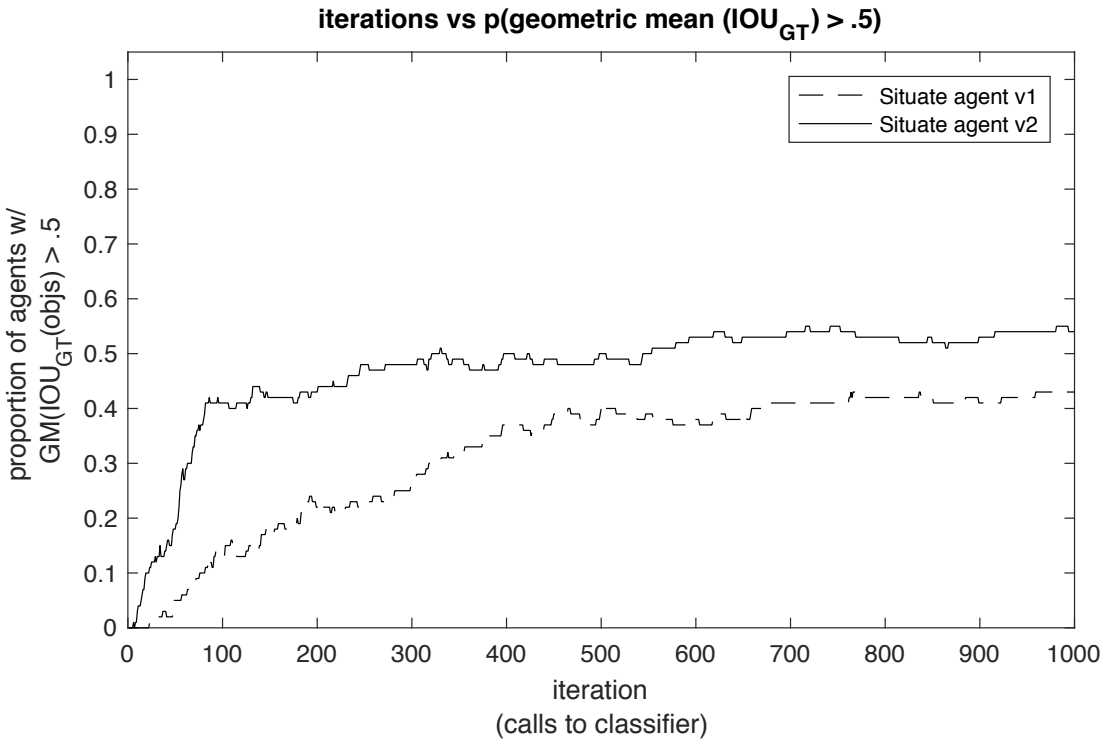


Figure 3.15: While the agents were running on the evaluation images, I recorded the geometric mean of the ground truth IOU values for each of the objects represented in the Workspace (and included the value .01 for objects that were not represented in the Workspace). This plot shows the proportion of the agents (each running on one image from the evaluation set) that have a geometric mean value greater than .5. Although this value is not equivalent to the number of agents that have produced correct solutions, it is a good proxy for the quality of Workspaces.

The Situate agent v1 makes changes to its Workspace even after many iterations, continuing to improve at a steady rate through the first 500 calls to the classifier. The Situate agent v2 develops its good Workspaces fairly quickly, with about 40% of agents having developed a good Workspace within about 100 calls to the classifier (of the 55% of agents that will eventually do so).

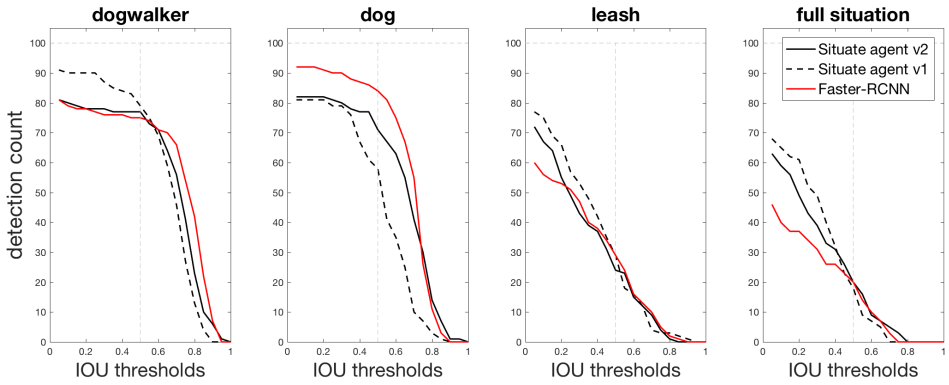


Figure 3.16: The object localization quality for the two versions of the Situate agent and Faster-RCNN are similar to one another for the dog-walker and leash objects at the .5 IOU threshold (although differ a bit for less tightly localized bounding boxes). The localization quality of the Situate agent v1 improves substantially with additional calls to the classifier. The Situate agent v2 improves on v1 in detecting the dog object.

very low ground truth scores for one or more objects. That is to say, these images are examples of what leads the Situate agent v2 to have high confidence in a poor response. There is a frequent tendency to localize an object that is of the correct type but is an incorrect instance (i.e., an incorrect person in the image). When this happens, the final Workspace generated by the agent tends to remain incomplete or to include bounding boxes that do not contain an object of interest, but for which the low internal support score was bolstered by high external support. There is no mechanism by which the agent backs away from initial detections and redirects its attention for the other regions in the image.

Figure 3.18 shows examples of the dog-walking situation that produced the lowest situation scores from the Situate agent v2. Here, we again see what appears to be the agent committing an erroneously detected object to the Workspace and then floundering. The initial objects committed to the Workspace were not the highest scoring examples of their object types in the image, but once committed, the Situate agent v2 failed to develop its Workspace.

The above errors demonstrate the tension between the goal of developing a Workspace

Figure 3.17: The above Workspaces were generated by the Situate agent v2 and were given high situation scores, but had very low ground truth scores for one or more situation objects. These examples demonstrate the tendency of the Situate agent v2 to localize incorrect instances of objects of the correct type and to accept low quality instances of contextually recognizable objects, rather than to identify the correct instances of objects. (Best viewed in color)

quickly by committing objects to the Workspace as soon as possible and goal of developing good Workspaces that start with good examples of constituent objects. Once poor examples are committed to the Workspace, the agent is essentially in a local optimum with respect to situation score, and is not able to improve. Although it is a failure mode, the results in figures 3.16 and 3.15 show that the Situate agent v2 is at least able to arrive at this failed state more quickly than the Situate agent v1. For this reason, the Situate agent v2 will be used moving forward. In the next chapter I will describe how Situate allocates resources such that failed agents do not continue to receive resources and remaining resources are used to search for alternate solutions.

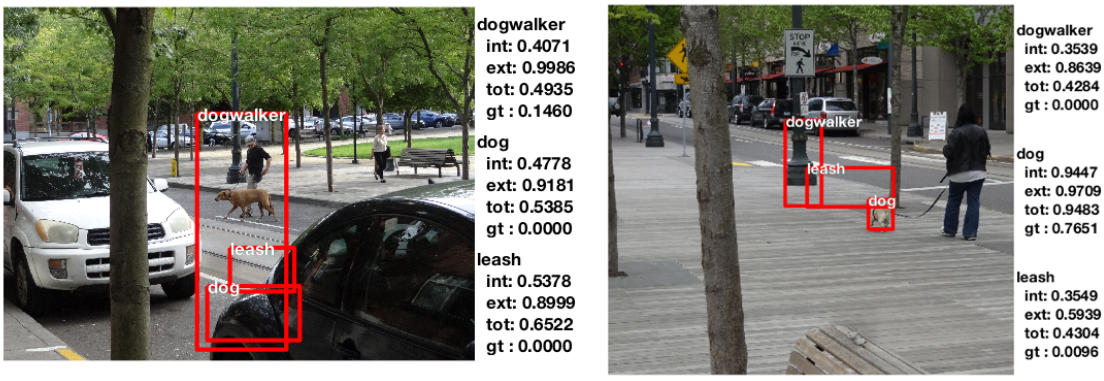


Figure 3.18: The above Workspaces were generated by the Situate agent v2 and were given the lowest situation scores from among positive instances of the dog-walking situation. These examples demonstrate, like in Figure 3.17, the tendency of the Situate agent v2 to commit to incorrect instances of objects and to accept low quality instances (or outright hallucinations) of other objects. (Best viewed in color)

Chapter 4

Managing Multiple Situate Agents

In this chapter I will discuss how Situate uses multiple agents to perform the situation recognition task. The use of multiple agents allows multiple possible solutions to an input to be considered while allowing the individual agents to pursue solutions greedily. I will describe the mechanisms by which the multi-agent approach addresses some of the issues identified in the previous chapters, the implementation challenges associated with the approach, the solutions to those challenges, and will compare the full multi-agent implementation of Situate to an individual Situate agent and to Faster-RCNN.

4.1 Multi-Agent Approach

In previous chapters, I discussed the individual Situate agent and how it uses an updating Workspace to detect objects of interest related to a situation. An individual Situate agent often fails to identify the correct objects of a situation for two related reasons:

Failing to detect objects important to the situation

Results from previous chapters included multiple examples of agents that failed to identify objects that were important to the situation. Often these objects appeared in low-likelihood locations in the input or in regions that were not of interest based on the current contents of the Workspace. In both cases, these failures seems to be due to insufficiently exploring the entire input.

Failing to detect relevant examples of objects of the situation

Other examples of failed situation groundings included well localized instances of objects that are of a type included in the situation, but were not the correct instance (such as localized people in dog-walking images that were not walking the dog). The agent then expended its resources searching around the already localized object and failed to recognize other instances of the object. This occurs because the only way to replace an object in the Workspace is to commit a higher confidence instance, but the confidence may not be higher for the correct instance absent supporting contextual evidence.

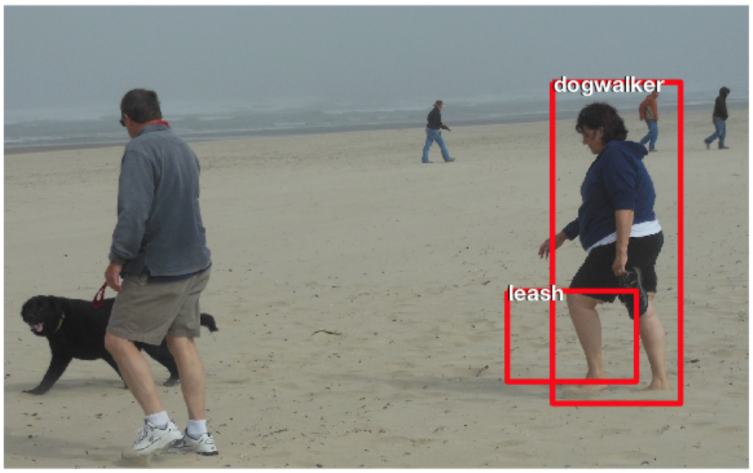
Figure 4.1 shows two examples of these failures and the incorrect final Workspaces that result.

- In figure 4.1a, the dog was correctly localized, and there were two people that could have been reasonably identified as the dog-walker. The incorrect person was detected, and the Workspace was finished with an incorrect leash detection.

  At this point in the evaluation, the other person (on the right) would not have higher total support than the detected person because the person on the right side of the image would be less compatible with the erroneous leash detection. Likewise, the correct leash detection would not have higher total support than the existing erroneous leash due to the location of the detected person. Situate would just expend any remaining processing resources without changing the Workspace. That is to say, Situate has found a local optima that will not be escaped with any single change to the Workspace.

  A better use of remaining computational resources might been to consider other possible sets of objects. Those alternative Workspaces may or may not have resulted in a higher situation score than the one that did result, but the alternative Workspaces should be considered.

(a)



(b)

Figure 4.1: The upper image contains two reasonable options for the dog-walker, one of which was selected and committed to. The resulting workspace looks reasonable, but the alternative was never evaluated. Even if it were, the fact that there were two reasonable responses was not communicated by Situate in its output. The lower image shows multiple reasonable options for the dog-walker. The person selected did not lead the agent to find an instance of the situation, but the agent also never moved on from its initial detection of a person. (Best viewed in color)

- In figure 4.1b, the person that was detected focused the agent's attention to a constrained image region. A Workspace that started with the other person was never considered by the agent, and, as the other person may have had lower internal support than the detected person, would not have replaced it in the Workspace. The agent expended its remaining resources searching for a dog that was not present in the location suggested by its Workspace.

Using multiple agents, Situate addresses these issues by a) searching more of the image for possible starting objects for a Workspace, and b) allowing detected objects to lead to multiple Workspaces through a forking procedure. However, this approach creates issues related to resource allocation and redundancy that Situate needs to address, as multiple agents exploring from similar Workspaces adds nothing, and expending resources on agents with poor Workspaces before developing promising Workspaces delays possible findings. Therefore, resource allocation to Situate agents should accomplish several goals:

- Unproductive agents should be recognized so avoid wasted resources.
- Resources should not be allocated to multiple agents that have substantially similar Workspaces.
- Promising agents should have resources allocated to them early so that solutions can be developed quickly. Remaining resources should be utilized to develop alternative responses.

My method for accomplishing these goals came by way of borrowing from the game tree traversal method *Monte-Carlo tree search*. Game trees are structures that represent turn-based games, usually with multiple players and perfect information (like chess, checkers, or Go), where nodes represent a possible state of the game,

edges connect game states that result from available moves in the game, and the goal is to find a sequence of moves that lead to a "winning" game state.

Games with a limited number of moves and a short duration can be exhaustively evaluated, but others have a sufficiently large number of available moves and possible board states that exhaustive evaluation is unfeasible. In this situation, strategies for evaluation game trees are generally based on heuristic methods that score potential moves by estimating how likely they are to lead to a winning game state. The features that relate game tree traversal to Situate are the sequential nature of building solutions, the large number of possible changes to the solution state at each step, and the inability to evaluate the quality of a state without some sort of subsequent investigation.

Monte-Carlo tree search (MCTS) has become especially well known due to its success related to the game Go, a game notable for being particularly challenging due to the number of available moves from each game state and the difficulty in estimating the quality of a game state. MCTS is a heuristic method that addresses both of these issues through sampling and simulation [7].

MCTS uses stochastic selection of available moves and repeated "play-outs" of a game from the resulting game state to estimate the quality of the move. A play-out consists of a series of randomly selected legal moves that result in either a win or a loss. The ratio of play-outs that result in a win serves as an estimate of the quality of the game state. That estimated quality of the game state is used to bias the selection of game states toward states that are promising, which improve the quality of the estimation. Eventually, the estimating stops and MCTS commits to the move that produced the highest quality game state, now having been repeatedly tested. This combination of stochastic move selection and repeated play-outs means that MCTS delays committing to a move until it has exhausted its allotted iterations, and biases

the allocation of those resources toward its most promising options.

MCTS accomplishes its task in a manner that generally satisfies the aforementioned goals for our agent management system.

- Unproductive game states do not receive many play-outs due to their low sampling probability.
- Game states are uniquely represented in a well constructed game tree, meaning redundant play-outs are never applied to duplicates.
- Promising board states are the most likely to be sampled, meaning the quality of those states are evaluated as early as possible.

Situate manages its agents in a manner that is largely similar to MCTS. Rather than spending computational resources on playing-out board states, computational resources (here, iterations) are allocated to agents to investigate the input from the context of their Workspaces. That allocation is done in proportion to the current situation score of the agent's Workspace. Hence, promising Workspaces are likely to be evaluated early, producing reasonable solutions quickly. After a fixed allocation of iterations have been expended, the agent is retired. These agents are no longer used for evaluation, but serve as markers of Workspaces that have been sufficiently evaluated and should not be inadvertently recreated.

Table 4.1 provides psuedocode for Situate's agent management system and figure 4.2 provides an example of the resulting Workspaces after Situate has been applied to an image.

The root of the graph in figure 4.2 shows the input image and several objects detected during an initial object search. The initial object search consists of an application of Faster-RCNN to identify objects that have sufficient internal support to be included in an otherwise empty Workspace. Each of those detected objects are

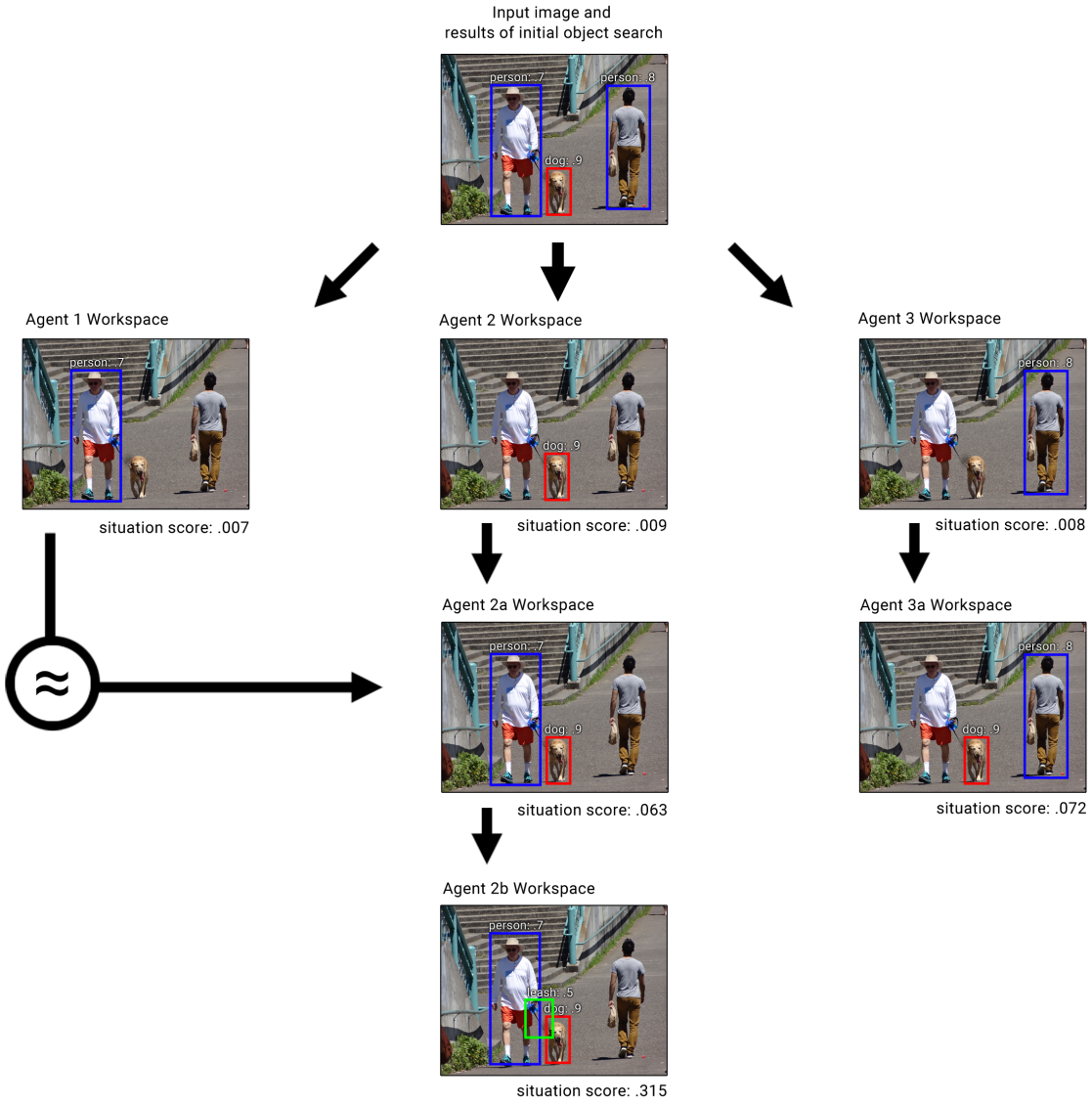


Figure 4.2: Situate performs an initial search for relevant objects. Each detected object is used to generate an initial Workspace that is assigned to a Situate agent.

Situate samples from among the agents, which in turn search the input for additional objects to update their Workspaces. Updated Workspaces are compared to those of other agents to prevent redundancy.

Once agents have reached their termination conditions, Situate returns the Workspace with the highest situation score. (Best viewed in color)

used to initialize a separate Workspace and are assigned to their own agents (agents 1 through 3 in this example).

Situate then selects an agent through a weighted sampling procedure and the input is searched from the perspective of the selected agent's Workspace. If something is found and added to the Workspace, then a new agent is generated with the new Workspace. For example, suppose that agent 3 is sampled and that it locates the dog at the center of the image. The Workspace from agent 3 and the dog will produce a new Workspace. It will be compared to the other Workspaces known to Situate. If the new Workspace appears to be unique, agent 3a will be generated with the new Workspace. The existing agent, agent 3, remains as an available agent to be selected by Situate but has a low probability of being sampled due to agent 3a's higher situation score. The Workspace in agent 3a narrows its search for a leash object to a location that does not contain a leash. After searching for some number of iterations, agent 3a will be retired by setting its probability of being sampled by Situate to zero. Once an agent is retired, it remains known to Situate to avoid generating new agents that replicate its work.

The set of remaining agents includes: agent 1, agent 2, and agent 3. Suppose that agent 2 is selected and it locates the person in the left of the image, generating agent 2a. Then suppose agent 1 detects the dog at the center of the image. The resulting Workspace would be substantially similar to the Workspace of agent 2a, so no new agent would be generated.

Once all agents have expended their alloted computational resources (or have met another termination condition), Situate returns the Workspace with the highest situation score. In this case, the Workspace contained in agent 3a. Returning the other Workspaces can confirm to a user that Situate did consider alternative objects and combinations of objects.

4.1.1 Differences from MCTS

There are important differences between exploring unstructured data and evaluating game states that lead to a few notable differences between how Situate manages agents and how MCTS selects moves.

- There are no play-outs in Situate. Situate uses agents to investigate its input through sampling and scores the samples in the context of a Workspace. Table 4.1 refers to the evaluation performed by a Situate agent, as described in previous chapters, as *evaluate*.

- Board states have an estimated probability of leading to a winning board state, and that estimate is used to bias the distribution of "play-outs". In Situate, there are no such winning game states, but Workspaces do have a situation score, which is used to bias the distribution of computation resources in the form of iterations allocated to an agent.

- Without discrete moves, it is not as obvious when two states are so similar that they are essentially doing redundant work. This means that Situate needs a method for identifying very similar Workspaces. The measure I constructed uses the same .5 IOU threshold used in other contexts to check the similarity between individual objects in the Workspaces. If each object in a Workspace has a correlate in the other Workspace, and each pair of objects has a .5 IOU score between them, then the Workspaces are considered to be "almost equal". This approximate measure allows Workspaces to be identified as substantially similar before iteratively refining bounding boxes. In table 4.1, the $\approx$ symbol stands in for this rough equivalence.

- The quality of the estimate of a game state continues to improve from additional play-outs even after the game state has received significant attention. There are

limited benefits associated with expending additional resources on a Workspaces after bounding box adjustment has completed and a good number of samples have been drawn from its context. To address this difference, I set a fixed number of iterations after which, if an agent has not produced a new Workspace, the agent is retired. The specific number of iterations remains a parameter (called *retire_threshold* in table 4.1).

- Most games have a fixed starting board state, so rather than evaluating the same set of early moves each game, a set of good early moves can be stored. This set helps to constrain the complexity of the early game. The appropriate set of "early moves" for Situate is to do a cursory scan of the input. This lets Situate identify clearly detectable objects and can provide an assurance that all image regions are examined at least once.

  A straightforward method for doing this is to simply pre-process the image with Faster-RCNN and to include high confidence bounding boxes as initial entries in the Workspaces of Situate agents. Other methods could also be used, including simply starting with a pre-defined grid of bounding boxes that cover the image. Regardless of method, this process is referred to as *initial_object_detection* in table 4.1.

input: $im : image, s : (situation\_structure, visual\_classifiers, situation\_model)$

output: $w* : workspace$

initialize

$Detections \leftarrow initial\_object\_detection(im, s)$

$Workspaces \leftarrow \emptyset$

$Workspaces_{inactive} \leftarrow \emptyset$

for each $d \in Detections$

if $d.support > detection\_threshold$

$w_{new}.objects[0] \leftarrow d$

$Workspaces \leftarrow Workspaces + w_{new}$

for each $w \in Workspaces$

$w.evaluation\_queue \leftarrow \{d \in Detections | d.type \neq w.objects[0].type\}$

Run

for each $i \in 1$ **to** $total\_iterations$

$w \leftarrow Workspaces.select()$

$w_{new} \leftarrow w.evaluate(im, s, eval\_iterations)$

$w.iterations\_since\_update \leftarrow w.iterations\_since\_update + eval\_iterations$

$collisions \leftarrow \{w \in W \cup W_{inactive} | w \not\approx w_{new}\}$

if $collisions = \emptyset$

$W \leftarrow Workspaces + w_{new}$

$w.iterations\_since\_update \leftarrow 0$

else if $w.iterations\_since\_update > retire\_threshold$

$Workspaces_{inactive} \leftarrow Workspaces_{inactive} + w$

$Workspaces \leftarrow W - w$

if $Workspaces = \emptyset$, **break**

$W \leftarrow Workspaces + Workspaces_{inactive}$

$w* \leftarrow Workspaces[argmax(w.support \textbf{ for } w \in Workspaces)]$

return $w*$

Table 4.1: Psuedocode for Situate's agent management method.

4.2 Evaluation of grounding performance

To evaluate the situation recognition quality of the multi-agent implementation of Situate, I applied it, as well as an individual Situate agent and the method based on Faster-RCNN, to the set of 100 testing images withheld from the Portland Simple Dog-walking dataset. Both the multi-agent implementation of Situate and the single agent were allowed a maximum of 1,000 calls to the image classifier (although in reality, the average number of calls to the visual classifier varies substantially for multi-agent Situate, as some images do not contain any confidently localized object during the *initial object detection* step, and therefore no additional calls to the classifier). Figure 4.3 shows the object grounding performance results on the dog-walking situation recognition task for each method.

Multi-agent Situate improves on Faster-RCNN and on the performance of an individual Situate agent for each of the object types. For the leash object, multi-agent Situate's improvements over Faster-RCNN in detections at the .5 IOU threshold are minor. However, multi-agent Situate is able to identify the approximate location with significantly higher frequency than Faster-RCNN. The bounding boxes generated by Faster-RCNN localize the leash at the .2 IOU threshold in approximately 40% of images, where multi-agent Situate meets that standard in approximately 70% of images. Although this is not the metric for correct grounding of a situation in an image, it does indicate that the Workspaces generated by Situate represent the content of the image to a reasonable degree for most images.

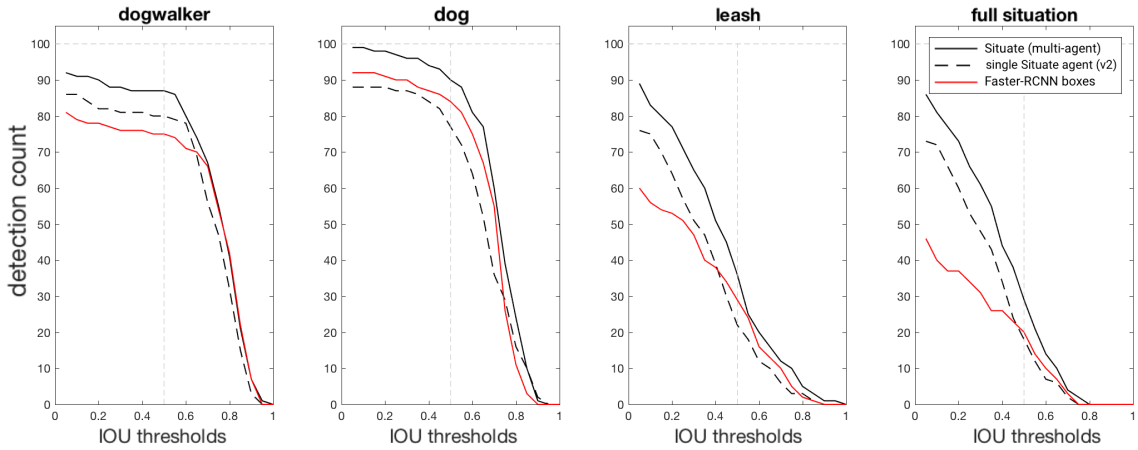


Figure 4.3: Object grounding quality comparison between Faster-RCNN, a single Situate agent (using the improvements from the previous chapters), and the multi-agent implementation of Situate.

Chapter 5

Applications and Additional Situations

5.1 Situate for retrieval

Situate's structured notion of a situation and its agent-based approach is useful for the situation recognition task, but the situation recognition task is not itself a use case. As discussed in Chapter 1.4, the situation recognition task can express several other tasks, including action recognition, referring expressions, and graph-based image retrieval. In this section, I will evaluate Situate from the perspective of image retrieval and compare it to the IRSG system described in chapter 1.4.

Recall that image retrieval tasks involve returning images from a corpus that satisfy a query. Queries are often expressed in natural language terms, but in this case we are using queries in the form of scene graphs, where a scene graph is a graph connecting objects and relationships [19]. Retrieval is performed by generating a score that relates how much each image in a corpus satisfies the query and then returning the highest scoring images. The quality of a retrieval system can be expressed or summarized in a number of ways, the most common including *precision/recall* (PR) curves, *receiver operating characteristic* (ROC) curves, and the area under the ROC curve (AUROC). Each communicate how well the retrieval system's scores predict the correctness of an image retrieved from the corpus. The notable difference between PR curves and ROC curves is that ROC curves are agnostic to the distribution of correct responses and incorrect responses in the corpus. That is to say, the ROC curve should not change if there are 100 positive instances and 100 negative instances

in a data set, or if there are 100 positive instances and 10,000 negative instances in the data set. PR curves are sensitive to the distribution within the data set. The AUROC simply summarizes the ROC curve in a single value.

To evaluate Situate in the context of image retrieval, I used Situate, IRSG, and the Faster-RCNN method to score images that contained a situation of interest and images that did not contain the situation of interest. A set of 5,000 images from the Visual Genome dataset [20] were used as a general set of negative examples for several situations. This particular subset was used in [19]. I removed several images that contained the situations of interest, leaving 4,978 negative examples The 100 testing images from the Portland Simple Dog-walking dataset served as the first set of positive instances of a situation. Additional situations are discussed later in this chapter.

Situate and IRSG are both constructed such that they generate a single score for an image with respect to a given query. To get a comparable value from the Faster-RCNN method, I used the padded geometric mean of the confidence values assigned to the highest confidence bounding boxes for each object type in the situation. That is, the situation score for the Faster-RCNN method is

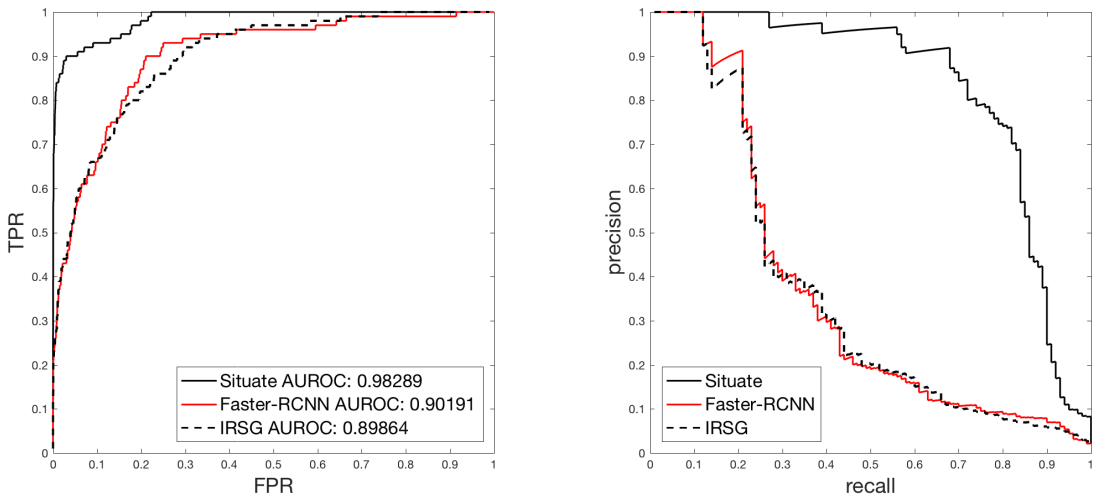
$$score(x, T) = \Big(\prod_{t \in T} \max_{b \in B_t(x)} \big(conf(b) + \epsilon \big) \Big)^{\frac{1}{|T|}},$$

where T is the set of object types in the situation, $B_t(x)$ is the set of bounding boxes generated by Faster-RCNN for objects of type t and the image x, $conf(b)$ is the confidence associated with bounding box b, and ϵ is a padding value (which I set to .01). If an object type from the situation was not detected, the padding value prevents the image from having a score of zero.

Figure 5.1 (a) shows the receiver operating characteristic (ROC) curves for Situate, IRSG, and the Faster-RCNN method. Each point along the curve is created by

selecting a threshold value and assigning each image in the corpus a predicted class based on whether the image's retrieval score is above or below that threshold. Then, based on those assignments, the false positive rate (FPR) and true positive rate (TPR) are calculated and plotted. This process is repeated for threshold values that span the full range of retrieval scores. If the values assigned to inputs were random, this process would produce a line from the bottom left to the upper right, representing the guess rate. A perfect classifier would produce a higher output value for each positive input than for every negative input, and would result in a curve that connects the bottom left corner to the upper left corner, and the upper left corner to the upper right corner. Figure 5.1 shows that Situate outperforms the Faster-RCNN method and IRSG by a substantial margin.

Image retrieval systems are almost never going to be applied to balanced data sets, so looking at performance given an imbalance can be informative. Figure 5.1 (b) shows the precision/recall curves for Situate, IRSG, and for the Faster-RCNN method given 100 positive images and approximately 5,000 negative images. These curves show that IRSG and the Faster-RCNN method were able to retrieve more than 10% of dog-walking images before making their first error. For the Faster-RCNN method, I presume that this is largely because images that contain high confidence instances of each of the constituent objects of the situation are, in fact, images containing the situation of interest. Notably, the quality of the results from IRSG (and even the very specific shapes of its ROC and PR curves) match those of the Faster-RCNN method very closely. IRSG applies Faster-RCNN to its input and uses the resulting bounding boxes in an energy minimization procedure. However, this minimization very rarely causes a deviation from the selection of the best individual bounding boxes that are greedily selected in the simpler Faster-RCNN method. This result is consistent with findings in the original report on IRSG and is discussed in more depth in Conser et



(a) ROC curves **(b)** Precision/Recall curves

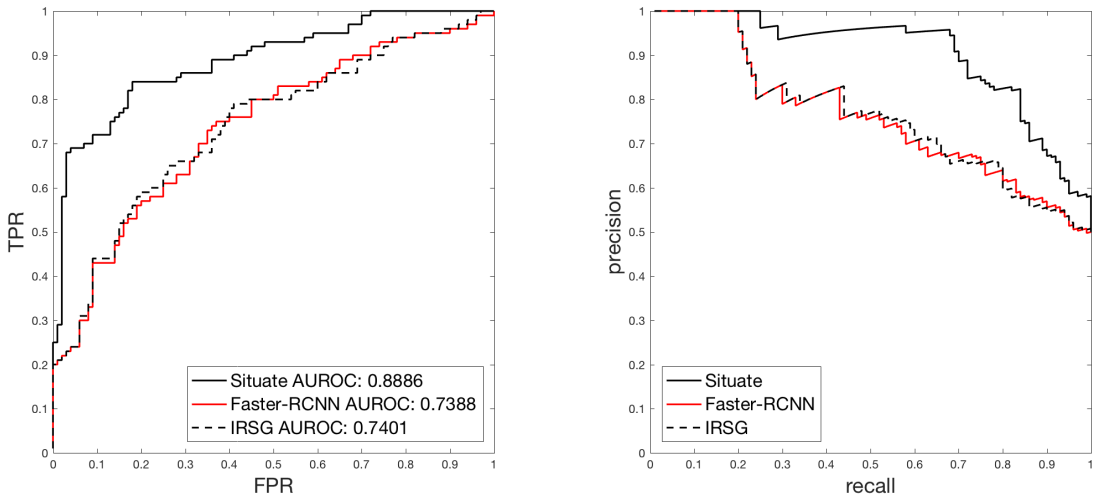
Figure 5.1: The curves above show the performance of the Faster-RCNN method, IRSG, and Situate on a retrieval task when a *general* set of negative images are used as the background class. These images, from the Visual Genome dataset, contain natural images from a variety of scene types, many including people. This task simulates general image retrieval when there is little known about the images that make up the corpus. The ROC curve gives a sense of the trade-off between sensitivity and selectivity in a manner that is agnostic to the proportion of images with positive instances of the situation and images without the situation in the data set. The precision/recall curve is not agnostic to the number of images containing the situation and the number of images without the situation in the data set, but does give a clearer indication of differences in performance when the positives are relatively rare. Here, the positive instances of the situation make up about 2% of the corpus.

al. [19, 6].

A practical application of the situation recognition to an image retrieval task on a large corpus of images would likely start with a preliminary labeling of objects in the images. However, even a robust labeling of objects is unlikely to include all object types that are especially indicative of situations, such as the leash object in the dog-walking situation. Then, querying the corpus for the situation might start by filtering for images that contain objects that were included in the preliminary search. In this context, the configuration of the objects and the presence of rare objects of the

situation are what distinguish the positive instances of the situation from negative instances of the situation. This would lead to a difficult, secondary phase of the image retrieval task where both positive and negative instances of a situation would both contain several objects of the situation.

To evaluate how well Situate would perform the retrieval task in the context of negative images that contain relevant objects, I compared the retrieval performance of Situate, IRSG, and the Faster-RCNN method on a set containing 100 images that contain positive instances of the dog-walking situation and 100 images that contain a dog and at least one person, but are not instances of the dog-walking situation. These images were from an additional set of images included with the Portland Simple Dog-walking dataset. Figure 5.2 shows the ROC curves that result from this evaluation. Not surprisingly, this much more difficult task led to a reduced classification quality for all methods, but the Situate method substantially out performs both the Faster-RCNN method and IRSG (which, again, have nearly identical performance to one another).



(a) ROC curves **(b)** Precision/Recall curves

Figure 5.2: The curves above show the performance of the Faster-RCNN method, IRSG, and Situate on a retrieval task when a *challenging* set of negatives is used as the background class. Challenging negatives include at least one person and at least one dog, but do not contain the dog-walking situation. This task simulates a hypothetical image retrieval application of Situate which assumes a corpus of images with a preliminary labeling that includes dogs and people as labeled objects, and assumes the corpus is pre-filtered based on the inclusion of both a person and a dog. Each method shows degraded performance when compared to the general set of negative background images (see Figure 5.1), but the performance of Situate degrades substantially less than that of the other methods. Note that these precision/recall curves should not be directly compared to those in Figure 5.1, as the proportion of positive images in the corpus has changed from approximately 2% to 50%.

5.2 Additional Situations

While developing Situate, I made many implementation decisions in the context of a single situation —the dog-walking situation. I tried to make those decisions in a manner agnostic to what I was learning about that situation, but very limited conclusions can be drawn from results on a single situation. To begin evaluating how well Situate generalizes to more situations, I applied multi-agent Situate to a few additional situations and compared situation detection and retrieval results to IRSG and the Faster-RCNN method described previously.

The additional situations evaluated in this section are "people playing ping-pong" and "people shaking hands". The handshaking situation consists of two people engaging in a handshake, and is defined by a graph consisting of the two separate people and their grasped hands. The ping-pong situation is defined as two players, the ping-pong table, and the net. For both of these situations, there are frequently additional people in the image that are not engaged in the situation of interest.

For both of these new situations, there are two instances of a single object type. In the case of ping-pong, there are two players, and in the case of the handshaking situation, there are two participants. Situate handles this by simply defining two distinct objects in the situation graph (e.g., *player1* and *player2*), although those objects share a single classifier. To apply the Faster-RCNN method to situations with multiple instances of the same object type, there is a single classifier that localizes instances of the object type. The highest confidence bounding box for the object type is used for the first instance of the situation object and the next highest confidence bounding box (which does not overlap with the first, based on the IOU greater than .5 standard) is used for the second instance.

For each of these situations, there were 500 positive examples that were gathered and manually labeled as part of my research group's project of constructing a situation

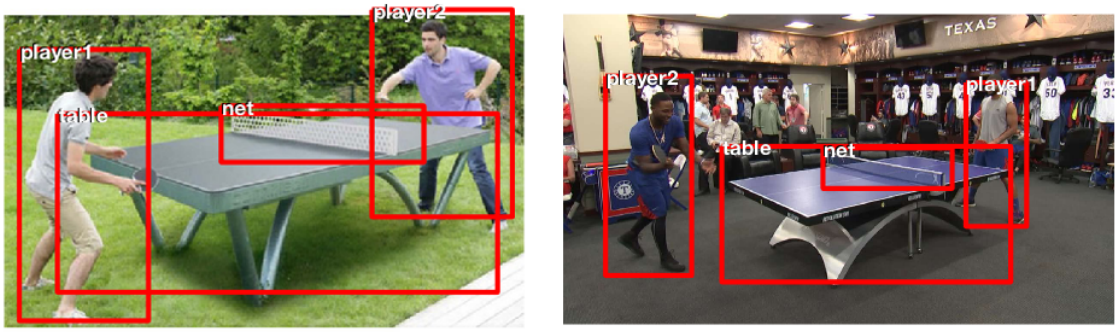
recognition dataset. Of the 500 images for each situation, I used 400 images for training and I reserved 100 images for testing. The 400 training images were used to tune the visual classifiers for each constituent object type, to train the situation model, and to estimate the reliability of the constituent object classifiers. For retrieval analysis, the 100 test images were considered in the context of a corpus made up of the same (approximately) 5,000 general negative images that was used when evaluating the dog-walking situation.

Examples of final workspaces generated for positive instances of the ping-pong situation by the Faster-RCNN method and Situate can be seen in figures 5.3(a) and 5.4(a). Figures 5.3(b) and 5.4(b) show several false positive detections from each method.

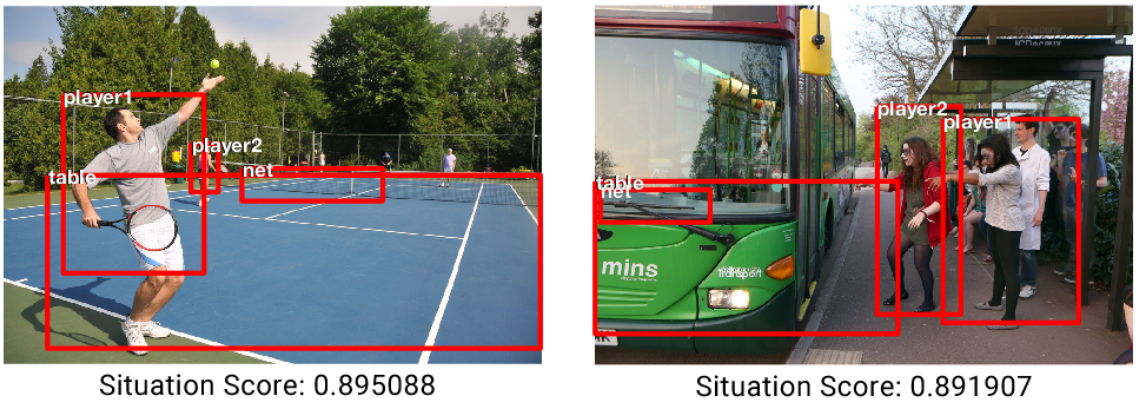
Figures 5.5 and 5.6 show examples of correct and false detections for the handshaking situation produced by the Faster-RCNN system and Situate.

Figures 5.7 and 5.8 show that the Faster-RCNN method and Situate perform similarly with respect to constructing groundings for positive instances for both situations. The notable differences are with respect to the localization of people participating in the situation and with respect to partial localization of other objects in the situation. However, in the context of image retrieval, there is a more clear difference. Figure 5.9 shows the ROC curves for the handshaking and ping-pong situations for Situate, the Faster-RCNN method, and IRSG. The handshaking situation appears to be rather difficult for all methods, likely due to the large number of images in the negative set that contain multiple people and the unreliably detectability of the handshake object. The occasional localization of people that are not facing one another suggests that adding more detail to the definition of the situation might improve the quality of the classification being made by Situate.

Applying Situate to these additional situations produced a similar pattern to that

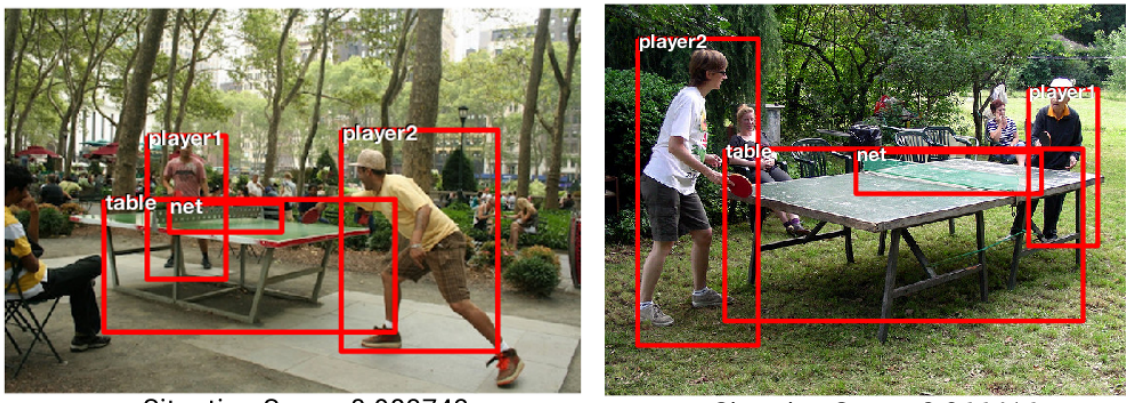


(a) High scoring ping-pong situation groundings generated using Faster-RCNN for images that do contain the situation.



(b) High scoring ping-pong situation groundings generated using Faster-RCNN for images that do not contain the situation.

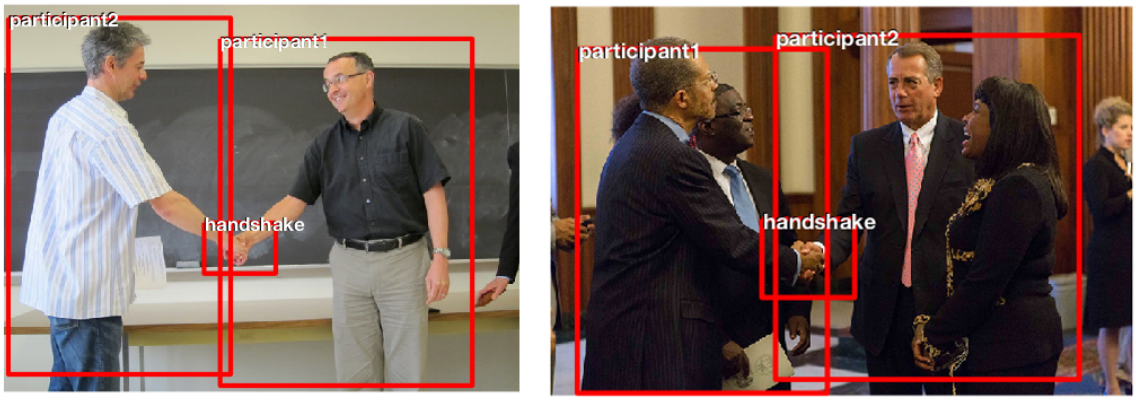
Figure 5.3: The set of bounding boxes in (a) were generated from the ping-pong images that scored the highest using the Faster-RCNN method. The bounding boxes in (b) were generated from the negative situation images that led to the highest scores. Although the image of people playing tennis is a charming, and possibly minor, error, the people waiting for the bus demonstrates that the detectors themselves are not selective enough to identify the most characteristic objects of the situation. (Best viewed in color)



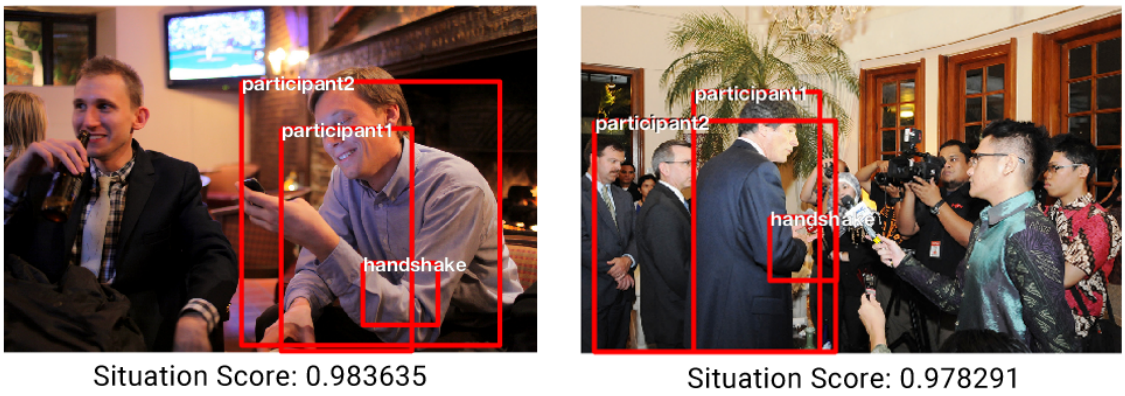
(a) High scoring ping-pong situation groundings generated using Situate for images that do contain the situation.

(b) High scoring ping-pong situation groundings generated using Situate for images that do not contain the situation.

Figure 5.4: The sets of bounding boxes in (a) and (b) represent the highest scoring examples of ping-pong found by Situate for actual instances of the situation and for images that do not contain the situation. The errors in (b) are substantial, but also suggestive of the sources of error. The "net" classifier is not specific enough, nor is the table classifier. These results suggest to me that it may be worthwhile to further define the situation by including characteristics like "players facing the table" or "paddle" objects. (Best viewed in color)

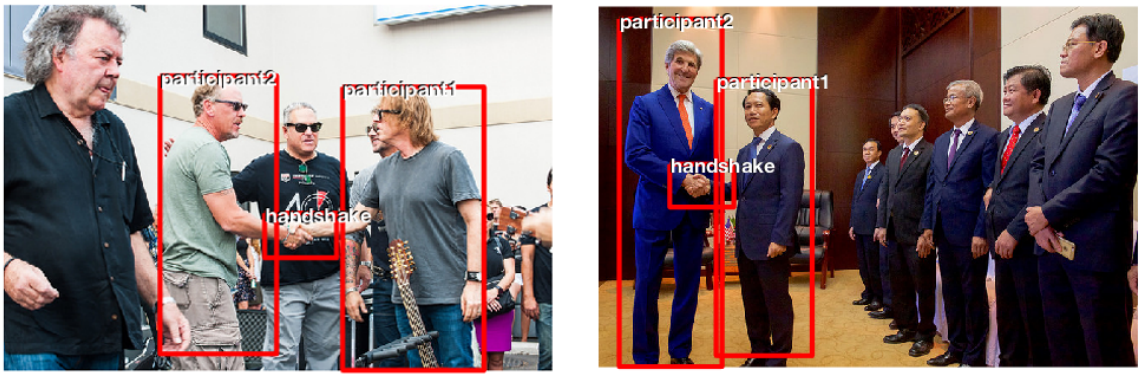


(a) High scoring handshaking situation groundings generated using Faster-RCNN for images that do contain the situation.



(b) High scoring handshaking situation groundings generated using Faster-RCNN for images that do not contain the situation.

Figure 5.5: The sets of bounding boxes in (a) represent the instances of the handshaking situation that the Faster-RCNN system scored the highest. The sets of bounding boxes in (b) represent the negative images that the Faster-RCNN system scored the highest. Although the "handshake object" detector was surprisingly accurate during this experiment, I believe it falls into the category of objects that are not likely to be explicitly searched for during the preliminary search of an image when entering it into a partially labeled database. The false positive detections show that the handshake object classifier is prone to false detections that can easily lead to false situation detections. (Best viewed in color)



(a) High scoring groundings for the handshaking situation using Situate for images that do contain the situation.

(b) High scoring handshaking situation groundings generated using Situate for images that do not contain the situation.

Figure 5.6: The sets of bounding boxes in (a) are from the instances of handshaking that Situate scored the highest. The sets of bounding boxes in (b) were generated from negative images that produced the highest scores for the handshaking situation. Both images in (a) demonstrate that the situation model is helpful in identifying the correct pair of people that are engaged in the handshake, but (b) demonstrates the limitations of the simple model used to represent the spatial distributions. The proximity and *connectedness* of objects is not sufficiently enforced, nor is the logic limiting the overlap of bounding boxes on a shared region of interest. (Best viewed in color)

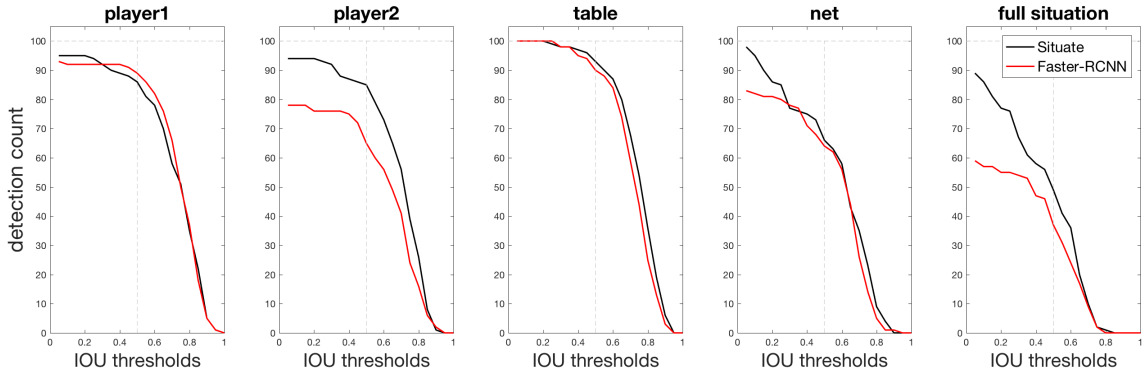


Figure 5.7: Localization quality for the ping-pong situation indicates that Situate performs marginally better than the Faster-RCNN method largely due to more consistently localizing both players. The localization quality for the player 2 object is substantially better for Situate. The starkness of the difference is due, in part, to the way that players 1 and 2 are defined for the Faster-RCNN system. For both systems, there is only one "player" classifier. For the RCNN system, the highest confidence instance of a player is recorded as player 1 and the second highest confidence instance (that does not collide with the first with an IOU score greater than .5) is recorded as player 2. Hence, the results for player 1 are generally better than for player 2. Situate's assignments are based on the sequence of the objects' addition to the Workspace, so are not distinguished by final confidence level.

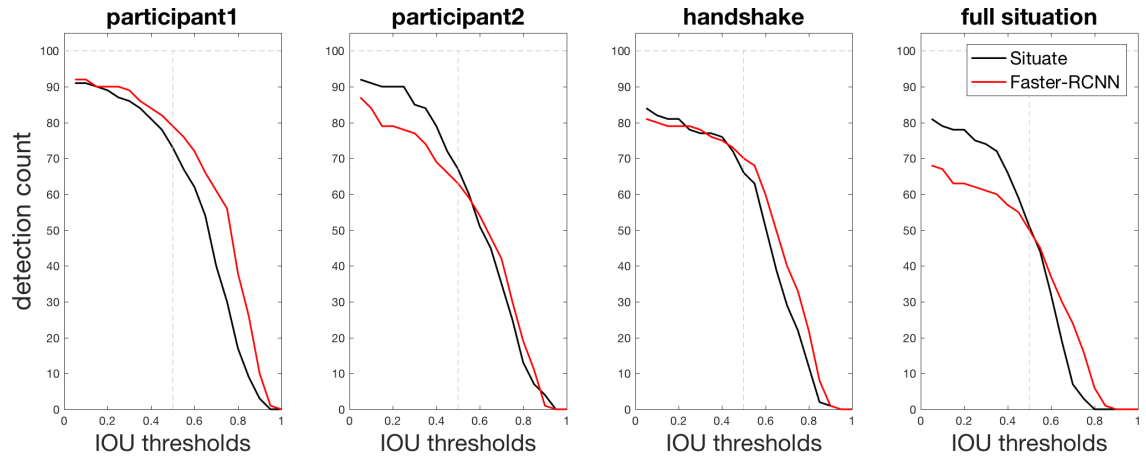
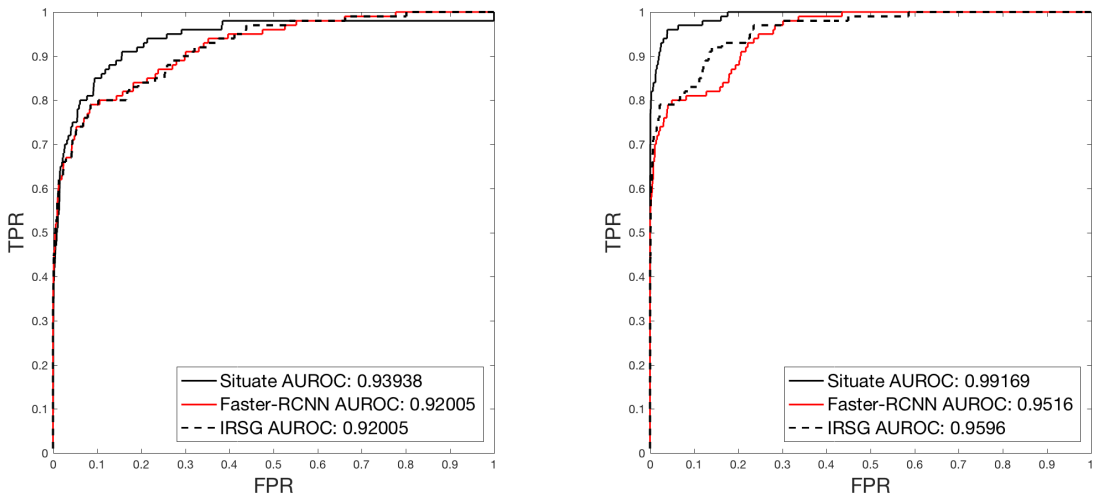


Figure 5.8: The total number of detected participants at the .5 threshold is very similar for both systems. However, just below that threshold of localization quality, we see that the correct people in the image are detected substantially more frequently by Situate, albeit with an imprecise bounding box.



(a) handshaking ROC **(b)** ping-pong ROC

Figure 5.9: Retrieval quality of Situate and the Faster-RCNN system for the ping-pong and handshaking situations. The positive instances consist of 100 images gathered from a variety of sources and the negative instances consist of approximately 5,000 images from the Visual Genome project.

seen for the dog-walking situation, where the localization of constituent objects is fairly similar in quality to the use of Faster-RCNN, but the discrimination results used for retrieval applications show Situate performing noticeably better.

Situation Score: 0.999436 Situation Score: 0.979733

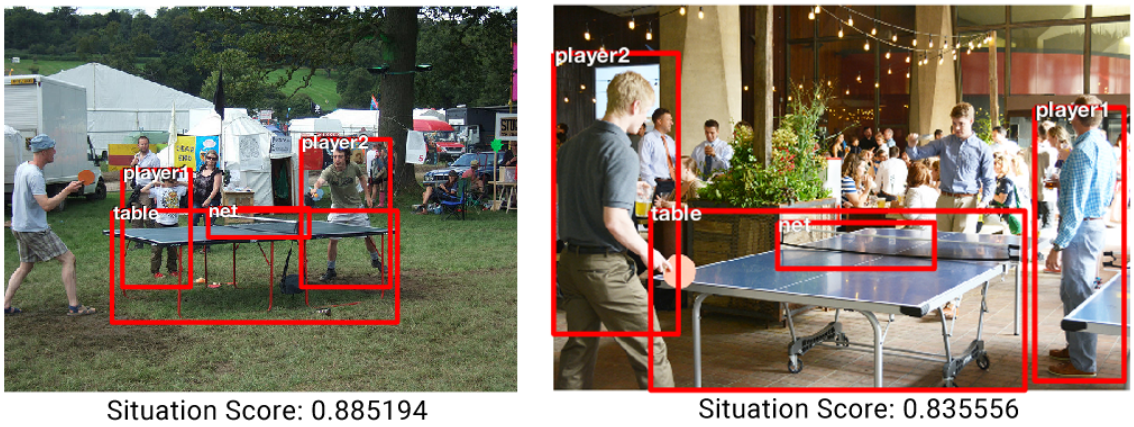
(a) Handshaking situations that received high situation scores from the Faster-RCNN method, but for which the ground truth objects of the situation were not correctly localized.

Situation Score: 0.860938 Situation Score: 0.872644

(b) Handshaking situations that received high situation scores from Situate, but for which the ground truth objects of the situation were not correctly localized.

Figure 5.10: Failures to correctly ground the handshaking situation by the Faster-RCNN method are usually associated with the misidentification of participants in the handshaking situation. The *handshake object* is, surprisingly, detected fairly accurately. Errors by Situate are more often associated with the misidentification of the handshake object. The Situate system might benefit from a more fleshed out model of the situation. For example, labeling the direction that each participant is facing and tuning the classifiers accordingly. (Best viewed in color)

(a) Ping-pong situations that received high situation scores from the Faster-RCNN method, but for which the ground truth objects of the situation were not correctly localized.



(b) Ping-pong situations that received high situation scores from Situate, but for which the ground truth objects of the situation were not correctly localized.

Figure 5.11: Failures to correctly ground an instance of ping-pong are similar for both Situate and the Faster-RCNN method, generally being the misidentification of people in the scene as players. The errors made by both methods are similar and demonstrates that a high quality localization of each object of a situation is challenging even given that the situation has been correctly recognized. (Best viewed in color)

Chapter 6

Discussion and Future Work

In chapter 1 I described several qualities of machine learning systems that I claimed were worth perusing, including:

- understandable decisions,
- a flexible trade-off between run-time and response quality, and
- flexible adherence to classification criteria.

Toward these ends, the work in this dissertation has made the following contributions:

- it presents Situate, an agent based approach to image understanding that integrates convolutional neural networks for classification with a simple situation model and tree building structure that constructs a structured interpretation of an input image;
- it presents the situation recognition task that includes a standard of evidence for image classification problems;
- it demonstrates an attentional system that uses initial detection quality estimate to improve bounding box regression;
- and it makes novel use of Monte-Carlo tree search for managing agents for image understanding.

Situate makes progress in the area of understandable decisions for situation recognition problems by providing individual confidences for pieces of evidence used in the

final decision, as well as returning alternative solutions that were considered when making the decision. By providing a record of what was or was not considered by the system, Situate grants insight into the decision, whether that decision was correct or incorrect, supporting the goals of producing trustworthy decisions and correctable errors.

Situate's resource allocation method prioritizes the most promising partial solutions, allowing it to provide reasonable solutions quickly, even if not all solutions have been fully evaluated, and using additional processing resources to either develop confirmatory evidence for the initial solution (by way of identifying and rejecting alternative solutions) or by replacing the initial solution (by making improvements along the margins or replacing the solution with something entirely different). Together, the balance of fast initial solutions and slower, more robust solutions provides some trade-off between run-time and response quality.

Situate compared favorably to a system based on standard components (the Faster-RCNN method) and to its closest relative found found in the literature (IRSG) on localization and retrieval tasks. There remain clear limitations in my work on Situate up to this point, as well as clear avenues for future work. Some of the clear limitations include:

- The situation model currently used in Situate can be unnecessary for some visual situations, where relatively easily recognized objects are a strong indicator of the situation; and insufficient for others, where specific properties of objects should be recognized and more complex relationships should be encapsulated. However, as the distributions of parameters relating objects becomes more complex, the use of limited training data becomes more tenuous.

  To develop Situate for visual situation recognition further, I would explore the use of more flexible distribution modeling structures that could express more

complex relationships while remaining easy to condition quickly, such as Gaussian mixture models (GMMs). I briefly explored GMMs for the dog-walking situation, but found that the model generally produced similar distributions to the single high-dimensional normal that was already being used, meaning the added distribution complexity was not necessary for the situation as constructed. For more complex situation graphs, I imagine that would not be the case. To address issues of extrapolating from small data sets, certain relationships could start with a prior that is tuned during training. For example, the relationship "x is to the left of y" could instantiate an appropriate prior, which could then be adjusted based on a small amount of training data for the specific objects. A similar mechanism is included in IRSG, however, as we saw in chapter 5.1, this complexity did not seem to be helpful for the simple situations considered in this paper.

- For the analyses in this paper, Situate assumes a particular situation when considering an input. Situate found the strongest evidence for the given situation, rather than deciding from among multiple possible labels. Extending Situate to consider multiple situations at the same time should be a relatively straightforward process. Rather than having a fixed situation, a Situate agent would have a distribution over possible situations. To search for objects, the Situate agent would sample from among its known situations, and then probe the input as usual. If an object is detected and added to the Workspace, the distribution over Situations would be updated, which would in turn influence what the agent searches for and where.

  Of course, developing a robust graph that contains many situations and encodes relationships between objects that reflect the common sense understanding of those situations is a significant project. The development of this sort of knowl-

edge graph might be aided by automated methods, but using it to engender trust in systems that use such a graph requires that the graph be trusted as well, so would likely require significant human effort.

Additionally, this would be a significant step toward one of the motivating goals for Situate; the ability to consider non-standard examples, like those shown in Figure 1.2. This functionality would primarily come from an expansion of the graph that relates objects and situations, but could be utilized by Situate based on the modifications made to search for multiple situations.

Finally, I hope Situate can be applied to more diverse domains than visual situation recognition. Situate was designed to be flexible so that classifiers and situation models could be updated for additional situations and domains. Some of Situate's structures are more complicated than is necessary for very simple visual situations, and the situation model used in this paper is likely not complex enough to model complex visual situations or situations outside of the visual domain. However, the structure of a graph that stores information about how objects relate to one another, unreliable classifiers that relate those graphs to data, and a system for navigating those interactions has general applicability.

An obvious extension past still images would be into video, where a Workspace can be used to generate predictions about future observations, and Workspaces that make more accurate predictions gather more support. Depending on the application, this could require significantly more complex situation models, but for some specific use cases, I expect that relatively simple models could be useful. For example, errors related to a failure to disambiguate overhead signs and trucks obstructing roadways seem to be responsible multiple major accidents involving self-driving cars. The expected movements of objects in the roadway and overhead can be modeled relatively

easily, even if a visual classifier cannot distinguish them using raw classifiers. [1].

In retrospect, the consideration of calls being made to the classifier during input sampling was probably not necessary for the visual situation recognition task, as the classifier used to recognize component objects can be applied very efficiently. However, for other applications, situations might be made up of whole documents in a corpus that require substantial resources to interpret. In that case, the attentional mechanisms in Situate might look very different, but the notion of prioritizing attention and limiting classifier calls would become more important.

[1] In 2016, the CEO of Tesla motors indicated that the difficulty in distinguishing between large trucks obstructing a lane and overhead highway signs had been responsible for serious accidents. After years of tuning the system, the same problem continues to recur. A basic model that encodes expected observations when an object is at road level versus well overhead might disambiguate this particular confusion. https://www.tesla.com/blog/tragic-loss, https://twitter.com/elonmusk/status/748625979271045121, https://www.thedrive.com/news/33789/autopilot-blamed-for-teslas-crash-into-overturned-truck

Bibliography

[1] Bogdan Alexe, Nicolas Heess, Yee Whye Teh, and Vittorio Ferrari. Searching for objects driven by context. In *Advances in Neural Information Processing Systems*, volume 2, pages 881–889, 2012.

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

[3] H S Chang, M C Fu, J Q Hu, and S I Marcus. An adaptive sampling algorithm for solving Markov decision processes. *Operations research*, 53(1):126–139, 2005.

[4] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *British Machine Vision Conference*, 2014.

[5] Volkan Cirik, Louis Philippe Morency, and Taylor Berg-Kirkpatrick. Visual referring expression recognition: What do systems actually learn? In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 2, pages 781–787, 2018.

[6] Erik Conser, Kennedy Hahn, Chandler M. Watson, and Melanie Mitchell. Revisiting visual grounding, 2019.

[7] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4630 LNCS, pages 72–83, 2007.

[8] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. Language models for image captioning: The quirks and what works. In *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, volume 2, pages 100–105, 2015.

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

[10] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[11] Carolina Galleguillos, Andrew Rabinovich, and Serge Belongie. Object categorization using co-occurrence, location and appearance. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016.

[13] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 3022–3031, 2015.

[14] Guodong Guo and Alice Lai. A survey on still image based human action recognition. *Pattern Recognition*, 47(10):3343–3361, 2014.

[15] Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Applications to Nonorthogonal Problems. *Technometrics*, 12(1):69–82, 1970.

[16] Douglas R. Hofstadter and Melanie Mitchell. The Copycat project : A model of mental fluidity and analogy-making. In *Fluid Concepts and Creative Analogies*, pages 31–112. Harvester Wheatsheaf, 1995.

[17] Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 6693–6702, 2019.

[18] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228, 2018.

[19] Justin Johnson, Ranjay Krishna, Michael Stark, Li Jia Li, David A. Shamma, Michael S. Bernstein, and Fei Fei Li. Image retrieval using scene graphs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 3668–3678, 2015.

[20] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, and David A. Shamma · Michael S. Bernstein · Li Fei-Fei. Visual Genome. *ArXiV*, pages 1–44, 2016.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[22] Will Landecker, Michael D. Thomure, Luis M.A. Bettencourt, Melanie Mitchell, Garrett T. Kenyon, and Steven P. Brumby. Interpreting individual classifications of hierarchical networks. In *Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, pages 32–38, 2013.

[23] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Lecun Y., Bengio Y., and Hinton G. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 11–20, 2016.

[25] Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. Modeling context between objects for referring expression understanding. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9908 LNCS, pages 792–807, 2016.

[26] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. BLEU : a Method for Automatic Evaluation of Machine Translation. *Computational Linguistics*, (July):311–318, 2002.

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

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[30] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, volume 2017-December, pages 3857–3867, 2017.

[31] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[32] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[33] Michael D. Thomure, Melanie Mitchell, and Garrett T. Kenyon. On the role of shape prototypes in hierarchical models of vision. In *Proceedings of the International Joint Conference on Neural Networks*, 2013.

[34] J. R R Uijlings, K. E A Van De Sande, T. Gevers, and A. W M Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[35] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[36] Limin Wang, Zhe Wang, Wenbin Du, and Yu Qiao. Object-Scene Convolutional Neural Networks for event recognition in images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2015-October, pages 30–35, 2015.

[37] M. D. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning, 2019.

Appendix A

Additional Experiments

A.1 Full Situation Localization and Retrieval

I performed a brief experiment to evaluate how well the combination of IOU estimating classifier and bounding box regression can localize a full situation when the constituent objects of the situation are considered as a single object.

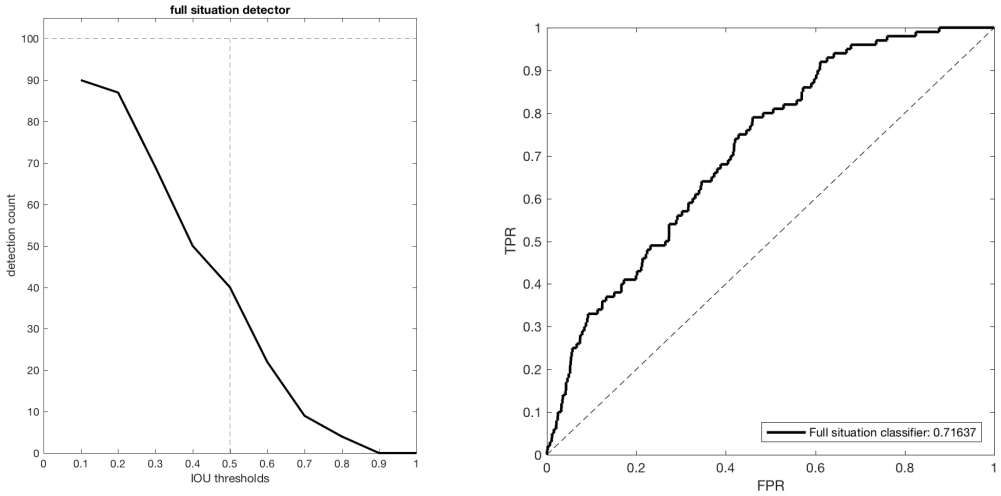
The approach to identifying the full situation "object" is very similar to the original RCNN method. For each image in a training set, I generated the minimal bounding box that contained the bounding boxes of each constituent object of the situation. I used these bounding boxes to train an IOU estimator as had been done for the constituent objects in chapter 3.1.1. I also constructed the models necessary for bounding box regression as was done in chapter 3.1.2. Then, for each image in the testing sets, I applied the classifier and bounding box regression system to a set of image crops that constituted a covering of the input image at various sizes and aspect ratios. The resulting bounding box with the highest estimated IOU with the underlying "object" was returned.

Figure A.1a shows localization results for positive instances of the dog-walking situation situation, i.e. the intersection over union between the highest confidence bounding box generated and the ground truth bounding box for the full situation for images that contain the situation. The method appears to do fairly well, with a .5 IOU localization of 40% of the positive images. However, one must keep in mind that there is no direct comparison between this result and the localization of parts performed in the situation recognition task, as the the components are not localized

here, simply contained within the larger bounding box that has been generated.

When considered from the perspective of retrieval, where negative instances are considered, the approach largely fails. Figure A.1b shows that the area under the ROC curve is just over .7, far below that of Situate and the RCNN-based system used in chapter 5, and not a particularly useful level of discrimination ability.

Figure A.2 shows some of the positive instances of the dog-walking situation that elicited the highest confidence from the localization system. For many of these images, the constituent objects of the situation are not actually included in the final bounding box, and although localizing the full situation with an IOU greater than .5 does not mean the standard for situation detection defined in chapter 1.2, the localization quality was still at least reasonable. Localization with an IOU of greater than or equal to .5 was successful in about 40% of images.



(a) **(b)**

Figure A.1: The localization quality of the full situation bounding box for the dog-walking situation is shown in (a). Of the 100 positive instances of the situation used for evaluation, 40% were localized at the .5 IOU threshold. The confidence in these localizations does not give the classifier strong discriminatory power, as can be seen in the ROC analysis in (b). The area under the ROC curve is .71637, indicates that the classifier is not particularly reliable.

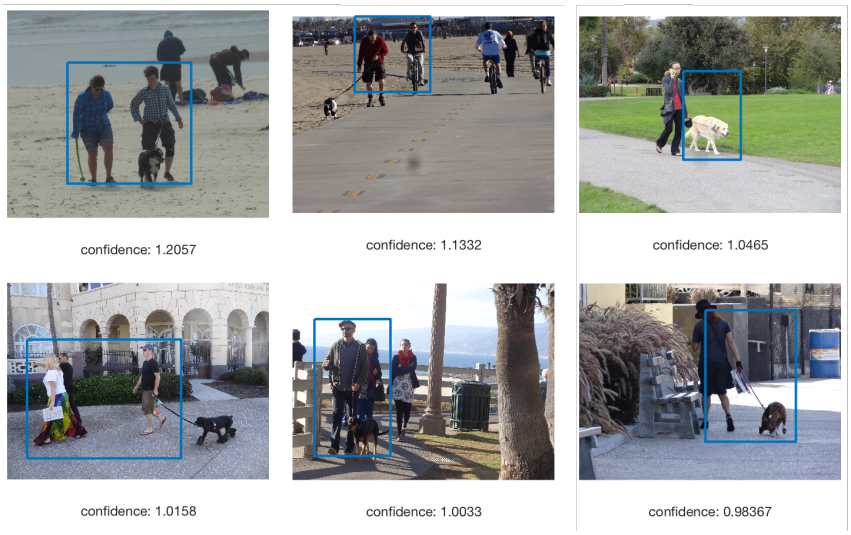


Figure A.2: Above are the highest scoring instances of the full situation detected by the full situation localization system. The IOU estimator has no enforced upper bound on its estimate, but these are among the highest activations that the estimator produces. A few of these high confidence bounding boxes are good, but they tend to be larger than necessary to localize the objects, and often contain only one of the constituent objects. (Best viewed in color)

Why is it the case that a system that is able to successfully localize the situation in question is unable to distinguish images that contain the situation from those that do not? Figure A.3 shows images from the set of images that do not contain the situation that elicited the highest confidences from the classifier. While one of the localizations is certainly an understandable error, most are quite inscrutable and are scored with confidences similar to the highest confidence positive instances of the situation. There are many possible explanations, including issues regarding the pre-training data and classes used to train the underlying convolutional neural network, the use of the regression based detector rather than a discriminative classifier, and the high degree of variability in the situation. To further confuse the diagnosis, the images that contain the situation but that elicited the lowest confidence from the classifier include several images for which the situation was actually localized quite

well, as can be seen in figure A.4.

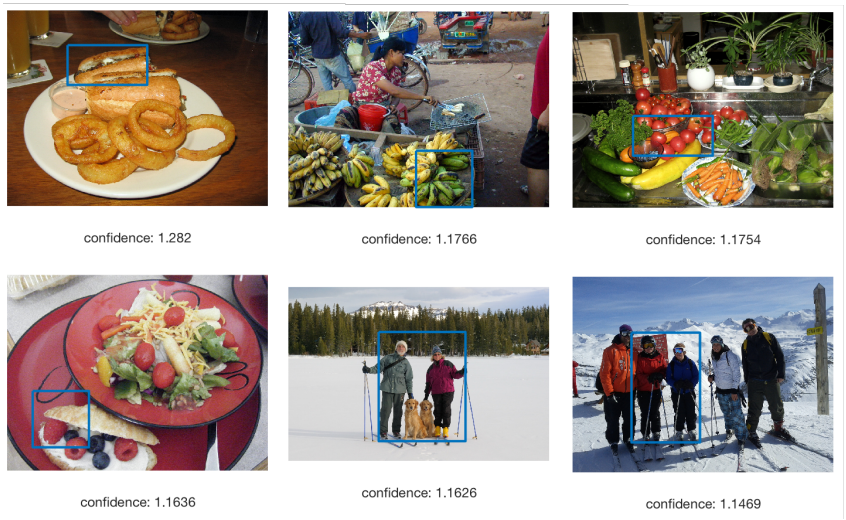


Figure A.3: The above bounding boxes generated the highest confidences from the set of images that did not contain the dog-walking situation. While it could be argued that a few of the images are reasonable mistakes (such as both images with skiers), most are difficult to understand. It is also worth noting that the confidence values associated with these images is as high or higher than the highest confidence images from the positive set. (Best viewed in color)

Although there are some obvious ways one might go about improving this classifier, such as providing more training data and doing more training that just the top level classifier, it's hard to speculate about what is currently leading to these errors. The situation consists of objects that the underlying CNN has been exposed to during pre-training and we know that classifiers for the individual objects perform fairly well from their performance in chapter 3. If significantly more training data could help the classifier to model more of the variability in the situation, or if more retraining of the network would suffice with the available data, the decisions made by the classifier would be no more clear and the returned structure would be no more useful outside of the narrowly defined localization problem.

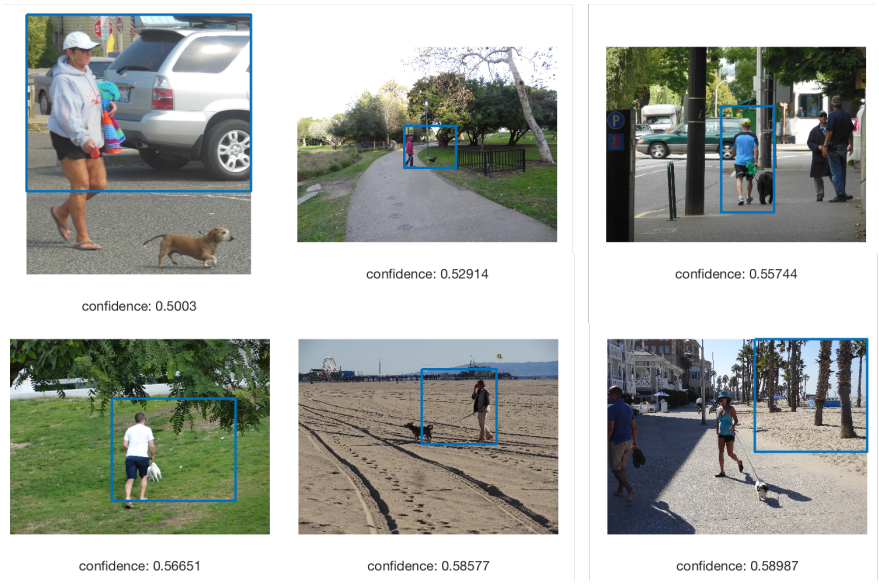


Figure A.4: The above images were the lowest confidence examples from the set of positive images. These images were recognized as the dog-walking situation with more confidence than about 10% of the images from the negative set. Several of these examples are actually fairly well localized instances of the situation, making it hard to speculate about what exactly is making them challenging examples. (Best viewed in color)