1.5 Multi-Instance Learning Models for Automated Support of Analysts in Simulated Surveillance Environments

Multi-Instance Learning Models for Automated Support of Analysts in Simulated Surveillance Environments

Mihnea Birisan and Peter Beling
The University of Virginia
mb5yv@virginia.edu, pb3a@virginia.edu

Abstract. New generations of surveillance drones are being outfitted with numerous high definition cameras. The rapid proliferation of fielded sensors and supporting capacity for processing and displaying data will translate into ever more capable platforms, but with increased capability comes increased complexity and scale that may diminish the usefulness of such platforms to human operators. We investigate methods for alleviating strain on analysts by automatically retrieving content specific to their current task using a machine learning technique known as Multi-Instance Learning (MIL). We use MIL to create a real-time model of the analysts' task and subsequently use the model to dynamically retrieve relevant content. This paper presents results from a pilot experiment in which a computer agent is assigned analyst tasks such as identifying caravanning vehicles in a simulated vehicle traffic environment. We compare agent performance between MIL-aided trials and unaided trials.

1.0 INTRODUCTION

As the number of surveillance projects has increased over the years, so has the amount of data collected that requires analysis. Projects such as Gorgon Stare have produced UAVs that can record video with 12 cameras simultaneously, thus amassing large quantities of video over short periods of time. While the collected information is a significant resource for defense analysts, the sheer volume of video that requires processing can be overwhelming. As a result, instead of improving mission effectiveness, the extra information strains the analysts, perhaps decreasing their effectiveness.

In this paper, we propose and test a method to decrease strain on analysts by dynamically presenting them with data most pertinent to the cognitive task they are currently carrying out. We assume that the adversaries targeted by analysts are highly adaptable in their approaches given past US defense responses. Therefore, the data filtering system we propose seeks to maximize the performance of analysts in the face of changing enemy doctrine.

While we discuss here the filtering of video data, we are not supplying a solution to the computer vision problem. Rather, we assume that data extraction from video is already possible and we therefore work with higher-level features stemming from video feature extraction. In order to provide some structure to the problem, we limited our environment to that of vehicle traffic and consequently built it to support possible analyst tasks regarding vehicle surveillance data. We assume that any possible analyst task will have a valid representation in our vehicle feature set. Our data filtering system first learns the analyst's task based on simple input from the analyst and then proceeds to support the analyst by providing information relevant to the learned task. We discretize an otherwise continuous data flow by time epics. The input given by the analyst is simply a “useful/not useful” label on the time epic of data just seen. If the data just presented to the analyst was useful in accomplishing their task, they will provide a “useful” label for that particular time epic. Otherwise, they will provide a “not useful” label for that time epic. This method of obtaining input from the analyst is advantageous because it does not require the analyst to describe their task at any length and technical detail. Instead, we learn the analyst's task based on the features present in the data for the time epics labeled useful. The analyst's task is both complex and dynamic and we believe that our approach is flexible enough and that building a template for each task would be impossible under the given time constraints and complexity of tasks.

Driving our data filtering system is a machine-learning algorithm know as Multi-Instance Learning (MIL). MIL is useful given our problem because it is responsive to changes in the analyst's task and because it does not require a label for every piece of data. A detailed description of the architecture and functioning of MIL follows in a later section.

Finally, in order to evaluate the value added to mission effectiveness by the MIL-driven data filtering system, we devised a way to test analyst mission effectiveness both with the MIL filtering system in place and without it. To this end, we devised a series of tasks relating to vehicle information and assigned them to a computer agent, which performed them both with and without MIL aid in a simulated environment. The agent was scored under both aid conditions and the scores were compared across all tasks. We will show that the agent performed better with MIL aid.

2.0 BACKGROUND

Machine learning can broadly be divided into two different approaches: *supervised learning* and *unsupervised learning*. In the supervised learning approach, the learning algorithm is provided with a label for every training example. Oftentimes, it is not feasible or possible to provide labels for training examples. Thus, in unsupervised learning, the learning algorithm is provided with completely unlabeled training examples, with learning algorithms in unsupervised learning stemming from clustering principles. MIL blurs the difference between supervised and unsupervised learning because it use partially labeled training examples.

MIL was first introduced in the context of the drug activity prediction problem described in reference [2]. Reference [2] also proposed the first algorithm to solve the MIL problem. In drug activity prediction, one must predict if a given molecule will bind to a target binding site. The binding site is located on a much larger molecule, such as a protein, and has a very specific shape, making it impossible for any given molecule to bind there unless is has the perfect matching shape. Incomplete information stems from the fact that while it is possible to tell whether a molecule did or did not bind to the target site, it is impossible to tell what shape it had when it did bind to the target site. This happens because each molecule can take on several different shapes based on its bond angles. Thus, the positive label given to a molecule that did bind to the target site is ambiguous in that it does not define the shape that the molecule took on when the binding occurred.

Following reference [2], reference [3] tested a new MIL algorithm on the drug activity data set and also tried two new applications: forming a concept of what a person looks like from a series of labeled pictures and dealing with noise in stock selection. Reference [4] then used the new algorithm from reference [3] for natural scene image classification. Reference [6] used the drug activity data set to test yet another MIL algorithm. Two more applications of MIL have been in automated, content-based image retrieval described in reference [7] and text categorization describe in reference [1]. Reference [7] used the same algorithm as reference [3], but applied it to a different problem (that of content-based image retrieval), thus showing that MIL algorithms are flexible enough to have a variety of applications as long as the structure of the problem is maintained. The content-based image retrieval problem, specifically, was also worked on by reference [5], who proposed a new algorithm for this problem. In this paper we use the algorithm first described in reference [3].

3.0 THE ARCHITECTURE OF MIL

Before we proceed, we must establish some terminology. We will refer to examples, such as the training examples discussed in the Background section, as *objects*. Each object has a representation in feature space known as an *instance*. In the supervised learning case, each object is described by a single instance and each instance has a *label*. Figure 1 shows the one-to-one relationship of instances, objects, and labels.

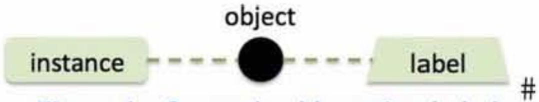


Figure 1 – Supervised Learning Labels

In the MIL case, however, there is no one-to-one mapping of instances, objects, and labels. Instead, each object can be represented by multiple instances in feature space. In the MIL case, labels are assigned to each object, not to each instance. Figure 2 depicts the architecture of labels in MIL.

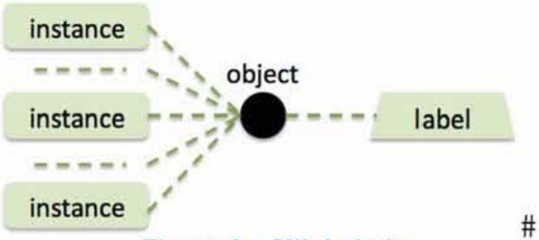


Figure 2 – MIL Labels

In MIL, the term *bag* is used to refer to an object. The term bag is used in order to illustrate that an object can "contain" or be described by several instances – a bag of instances. Since the label is not placed on each instance, but rather on the bag as a whole, rules must be set for labeling a bag as a function of the instances. A bag is labeled *positive* if at least one instance in the bag is positive. A bag is labeled *negative* if all the instances in the bag are negative. When looking at a bag labeled positive, it is ambiguous which instance triggered the positive label. MIL algorithms examine the instances in positive bags in order to find a feature space representation of the instances that triggered positive labels.

4.0 SIMULATION

We constructed a vehicle simulation environment to test if MIL-based data filtering would add value to operator mission effectiveness. In the simulation each vehicle exhibits either a normal behavior or a rogue behavior. Normal behaviors consist in entering traffic through an entry point, following traffic rules, and ultimately leaving the simulation through an exit point. Rogue behaviors consist in abnormal patterns that contradict traffic rules. In addition, each vehicle will also have two characteristics: color and type – car or truck. Table 1 shows each type of behavior and the rule used to determine if that behavior applies to a given vehicle.

Table 1 – Vehicle Behavior Definitions

Behavior	Rule
Speeding	Instantaneous speed >> speed limit
Slow Moving	Instantaneous speed << speed limit
Caravanning	2 or more vehicles with max Euclidian distance < epsilon distance, matching speed at every step within some epsilon speed, matching at least 4 turns
Abandoning	In a set of caravanning vehicles, one vehicle stops
Circling	Vehicle makes a series of more than 8 same-direction turns
Multiple U-Turns	Vehicle reverses coordinates

In MIL terminology, vehicle behaviors are instances, time epics are bags, and the description of behaviors in terms of low-level features defines the feature space. To support the instances (vehicle behaviors) in our simulation, we defined a feature space that is consistent with vehicle behavior metrics. Table 2 shows the features present in the simulation.

Table 2 – Feature Set

Feature Name	Type
Speeding Cars Present	Boolean

Number Speeding Cars	Integer
Slow Moving Cars Present	Boolean
Number Slow Moving Cars	Integer
Caravans Present	Boolean
Number Caravans	Integer
Abandoned Cars Present	Boolean
Number Abandoned Cars	Integer
Circling Cars Present	Boolean
Number Circling Cars	Integer
Multiple U-Turn Cars Present	Boolean
Number Multiple U-Turn Cars	Integer
Number Cars	Integer
Number Trucks	Integer
Number Red Vehicles	Integer
Number Blue Vehicles	Integer
Global Maximum Speed	Integer
Global Minimum Speed	Integer

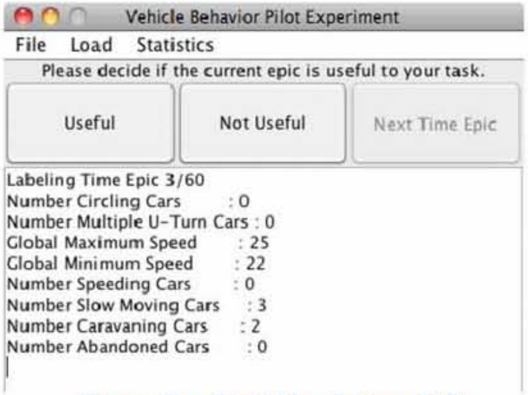
The simulation will be launched in two stages: the pilot stage and the full simulation stage. This paper describes the structure and test results on the pilot stage of the simulation and outlines the structure of the full simulation stage.

4.1 Pilot Stage

The pilot stage simulation is written in Java and is designed to be a proof of concept for MIL-based data filtering in a defense analysis environment. Since the pilot simulation is smaller in scale, it contains a subset of all the features present above. The features present are illustrated in Figure 3 below. The pilot simulation consists of three main components: the GUI, the simulation engine, and the MIL classifier.

4.1.1 Pilot Stage GUI

The pilot stage GUI is designed solely to obtain labels from an analyst or agent on different time epics. Figure 3 shows the pilot simulation GUI. The top part of the GUI allows the experiment organizer to load the simulation for each participant or agent. Each agent can then use the three buttons to provide a label on the summary data for a given time epic and to move on to the next time epic. Below the input buttons is a window that shows the summary data for a given time epic. The agent is only allowed to view the next time epic after they have provided a label for the current epic.



Please decide if the current epic is useful to your task.

```
Labeling Time Epic 3/60
Number Circling Cars        : 0
Number Multiple U-Turn Cars : 0
Global Maximum Speed        : 25
Global Minimum Speed        : 22
Number Speeding Cars        : 0
Number Slow Moving Cars     : 3
Number Caravaning Cars      : 2
Number Abandoned Cars       : 0
```

Figure 3 – Pilot Simulation GUI

4.1.2 Pilot Stage Simulation Engine

The simulation engine is in charge of loading the summary data for each time epic, presenting that to the agent, recording the label for each time epic, and finally instantiating the MIL classifier with the labeled data for each time epic. The simulation engine takes as input a text file containing the summary data for 60 training time epics. The engine reads the text file line by line, displays it to the agent, and then records the same data along with the labels provided by the agent in a new text file formatted to be readable to the MIL classifier. Once the agent has finished labeling each of the 60 training time epics according to their task, the simulation engine instantiates and trains the MIL classifier using the labeled data file as input.

Once the classifier object is trained on the labeled data, a new raw data file is presented to the classifier. This new file contains summary data for 60 evaluation time epics. Based on the concept learned in training, the classifier now predicts a label for each of the 60 evaluation time epics. The agent then proceeds to the first of two evaluation phases. The first evaluation phase is an aided phase where the agent is shown only time epics that the MIL classifier labels as matching the learned concept of the agent's task. In the evaluation phase, the "Useful/Not Useful" input buttons no longer provided labels for the classifier, but rather score the classifier, with the classifier's score increasing for every "Useful" time epic label provided by the agent in evaluation. After the aided evaluation phase described above, the agent is shown an equal number of randomly selected time epics. These time epics are not selected by the MIL classifier and thus may or may not be relevant to the agent's task. Once again, a score is incremented each time the agent labels a time epics as "Useful". In the Results section, we compare the evaluation scores in the MIL-aided and the unaided evaluation phases.

4.1.3 Pilot Stage MIL Classifier

The pilot stage MIL classifier is built based on the Diverse Density algorithm introduced by reference [3]. The classifier is instantiated in the pilot simulation using a Java jar file from the WEKA data-mining package. The simulation engine utilizes the classifier's training, prediction, and cross evaluation functions. Following general data mining rules, the simulation engine presents completely different data sets to the classifier for training and prediction. For each task we also run a cross evaluation on the prediction data set. The cross evaluation function uses parts of the prediction data set for training and parts for evaluation, recursively changing which parts are used for training and which for evaluation. We will also present cross evaluation scores across all tasks in the Results section.

4.2 Full Simulation Stage

The full simulation stage will be built on the existing pilot stage. The goal is to design a visual way to present each time epic to a human participant as opposed to a computer agent. Instead of showing the participant summary data for each time epic, the simulation will instead show a map with vehicles moving from entry to exit points. Each vehicle will have an associated track – GPS coordinates for each time step. Most vehicles will have normal, random behaviors, but some vehicles will exhibit "rogue" behaviors. Rogue behaviors will consist in speeding, moving too slowly, two or more cars caravanning, cars being abandoned, and so forth. While each time epic unfolds, the simulation engine will compute the value of each feature in feature space. Instead of labeling the time epic based on parsed summary data, the participants will have to observe how each time epic unfolds by following the cars and seeking behaviors relevant to their task. This method of presenting the participants with information is more realistic and similar to what an analyst would experience in a defense environment.

5.0 EVALUATING MIL

We used an agent-based simulation to measure if the MIL-based data filtering we propose in this paper can add value to analyst mission effectiveness. To that end, we created a list of fourteen tasks relating to vehicle behavior that an analyst might be interested in. Table 3 shows a list of the tasks. Each task is based in identifying at least one rogue vehicle behavior.

Table 3 – Tasks

Task ID	Task Type	Task
1	Simple	Identify ABANDONED vehicles
2	Simple	Identify CARAVANS
3	Simple	Identify CIRCLING vehicles
4	Simple	Identify SLOW vehicles
5	Simple	Identify SPEEDING vehicles
6	Simple	Identify U-TURN vehicles
7	Composite 1/2	Identify ABANDONED & CARAVANS
8	Composite 1/2	Identify CIRCLING & U-TURNS

9	Composite 1/2	Identify SLOW & SPEEDING
10	Composite 1/2	Identify SPEEDING & CARAVANS
11	Composite 2/3	Identify CARAVANS & ABANDONED & SLOW
12	Composite 2/3	Identify SLOW & CIRCLING & U-TURNS
13	Composite 2/3	Identify SLOW & SPEEDING & U-TURNS
14	Composite 2/3	Identify SPEEDING & CARAVANS & U-TURNS

Tasks 1 through 6 ask the agent to provide positive labels for time epics that exhibit a single vehicle behavior given in the task. Tasks 7 through 10 ask the agent to provide positive labels for time epics that exhibit at least one of two vehicle behaviors given in the task. Finally, tasks 11 through 14 ask the agent to provide positive labels for time epics that exhibit at least two of three vehicle behaviors given in the task.

A computer agent was assigned each of the fourteen tasks in turn. The agent labeled all 60 training time epics according to the task at hand. After the training phase, the agent proceeded to the evaluation phase. In the aided evaluation phase, the agent was only shown time epics that the MIL classifier labeled as matching the agent's task. The agent was given a positive point for every positive label it assigned to time epics in the evaluation phase. In the unaided evaluation phase, the agent was shown a random set of time epics and again scored on the number of positive labels it assigned. The evaluation data set did not contain exactly the same number of time epics matching each of the fourteen tasks. To be fair, the agent was shown exactly the same number of random time epics in the unaided phase as the number of time epics matching the agent's task in the aided phase. To illustrate, suppose that in the training phase, the MIL classifier learned that the agent had been assigned task 2. In the aided evaluation, the MIL classifier showed the agent all time epics matching task 2 in the evaluation data set. Suppose there were 13 matches. Then, the agent was also shown 13 random time epics in the unaided evaluation some of which happened to match task 2 and some that did not. This was done to ensure that the agent had a chance to score the same number of points in both the aided and the unaided evaluations.

As we discussed, the next stage in the simulation will provide real human subjects with a visual display of each time epic. This will mimic a defense analysis environment better than the pilot simulation, but will introduce the possibility of human error into all measurements. In order to simulate the effect of human error on the accuracy of the MIL classifier, we ran six more trials with imperfect labels. In these trials, the agent injected 1, 2, or 3 wrong labels when labeling the 60 training time epics. The wrong labels were both false positives (i.e. a bag was incorrectly labeled positive even though it did not match the task at hand) and false negatives (i.e. a bag was labeled negative even though it did match the task at hand), thus adding six more trials in addition to the perfect labels trial. This approach is perhaps unfair toward the MIL classifier because we simulate human error or indecision in the training phase, yet in the evaluation phase, the agent makes no mistakes in labeling, thus adding negative bias to the accuracy of the classifier. Nonetheless, the point was to stress the classifier by simulating real-world conditions. The Results section shows the performance of the MIL classifier across the perfect labels trial as well as the six imperfect labels trials.

6.0 RESULTS

We present the results from all seven trials as lift charts. We graph the scores form the unaided evaluations on the x-axis and the scores from the aided evaluations on the y-axis. If the aided and unaided evaluations exhibited similar performance, all points would lie on the 45-degree line. On the other hand, if the aided evaluation scores are higher than the unaided scores, then we expect the points to lie above the 45-degree line. Figure 4 shows the results from the perfect labels trial.

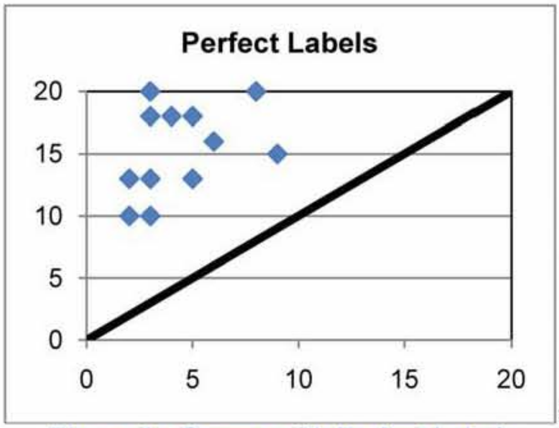


Figure 4 – Scores with Perfect Labels

The scores from all fourteen tasks are in the top left of the chart, showing that the MIL algorithm has added value to mission effectiveness by showing the agent time epics that matched its task in the aided evaluation phase. Figure 5 shows the results from the two trials with one wrong label. In this and all following figures the green (triangle) points represent false positive labels and the blue (diamond) points represent false negative labels.

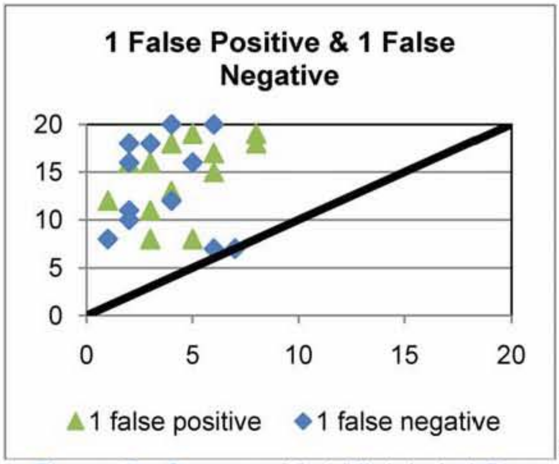


Figure 5 – Scores with 1 Mislabeled Bag

Here we see that some of the tasks have scores that are closer to the 45-degree line, suggesting less lift from the MIL classifier. Nonetheless, the majority of task scores remain in the top left region of the chart. Figure 6 shows the results with two wrong labels.

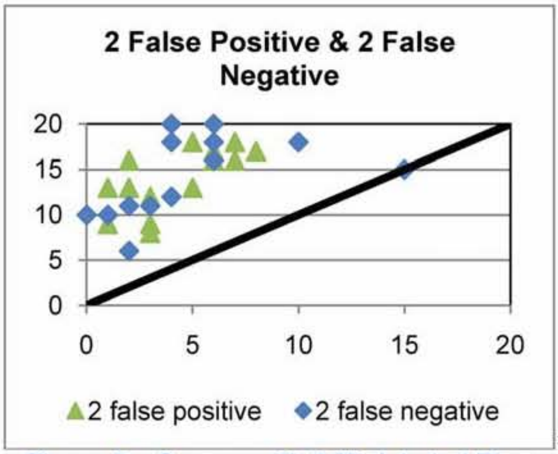


Figure 6 – Scores with 2 Mislabeled Bags

Finally, Figure 7 shows the results with three wrong labels.

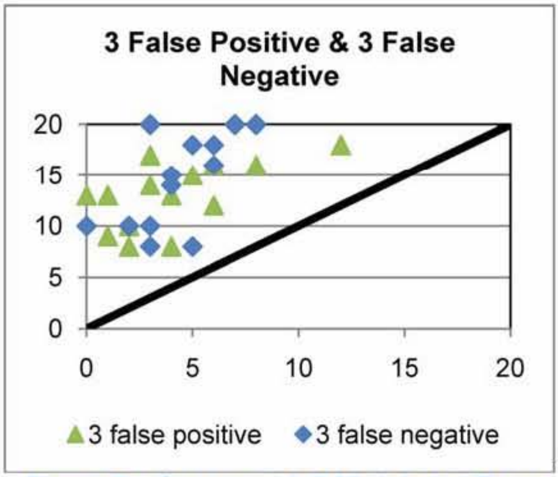


Figure 7 – Scores with 3 Mislabeled Bags

The difference between the one wrong label case and the two wrong labels case is not immediately noticeable. In fact, it appears that, overall, the one wrong label case resulted in lower scores than the two wrong labels case. In the three wrong labels case it is noticeable that aided scores are overall lower, detracting from the lift of the MIL classifier. Table 4 shows average score values across all tasks by the number of wrong labels. Averaging across all label types, we observe that the agent obtained aided evaluation scores that were 3.3 times higher than unaided evaluation scores.

Table 4 – Average Scores Across Tasks

	Score	
Label Type	**Aided**	**Unaided**
Perfect	15.50	4.57
1 False Positive	14.14	4.35
1 False Negative	13.79	3.71
2 False Positive	13.93	4.21
2 False Negative	14.64	4.79
3 False Positive	13.00	4.07
3 False Negative	14.07	4.14

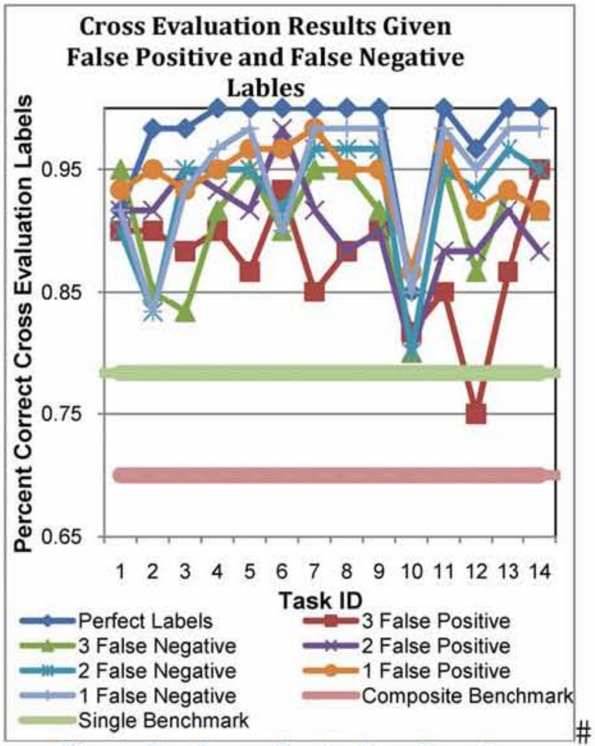


Figure 8 – Cross Evaluation Results

Given the current data, it is difficult to conclude if false positive or false negative training labels are more detrimental to the accuracy of the MIL classifier. Thus far, the scores show that false negative labels were more detrimental to the classifier if only one bag was mislabeled. If two or three bags were mislabeled, false positive labels were more detrimental to the accuracy of the classifier.

Figure 8 shows the cross evaluation results obtained on the evaluation data set. Each line shows the percentage of correct labels from the MIL classifier given different numbers of wrong training labels. Naturally, the perfect labels line is highest in the chart, indicating best classifier performance. The green horizontal line marks the single behavior task benchmark. In other words, if the classifier labeled all bags negative, it would still get 78% correctly labeled bags because only 22% of bags have true positive labels in the evaluation data set. The red horizontal line marks the composite behavior task benchmark. The classifier would get 70% correct labels if it labeled all bags negative because there were only 30% true positive bags in the evaluation data set. It is interesting to note that none of the single behavior tasks (1-6) dipped below the single benchmark and none of the composite behavior tasks (7-14) dipped below the composite benchmark, regardless of the number of wrong training labels. It is also noteworthy that all cross evolution results exhibited lower performance on tasks 10 and 12, indicating that there exists possible task dependence in the performance of the MIL classifier. To determine if such dependence exists with statistical significance, it is necessary to add more tasks and trials to the agent-based pilot experiment.

7.0 CONCLUSIONS

Based on the results from the pilot simulation, the Multi-Instance Learning classifier added value to agent mission effectiveness. The robustness of the results in the face of mislabeled training bags suggests that the classifier will continue to add value to mission effectiveness as we transition from agent-based simulations to human subject experiments on the full simulation.

With the current data it is not possible to determine with any statistical significance if false positive or false negative training labels are more detrimental to concept learning on the part of the MIL classifier. However, given more trials on the full simulation it may be possible to determine which type of label is more detrimental. This information is vital in creating a fully featured data filtering system for defense analysis applications.

This proof-of-concept simulation served to show that (1) a data filtering system is useful in relieving strain from today's information-flooded defense analysts and (2) that employing machine learning techniques is a feasible approach in building such a system.

8.0 REFERENCES

[1] S. Andrews, I. Tsochantaridis, T. Hofmann. Support Vector Machines for Multiple-Instance Learning. *NIPS 2002.*

[2] T. G. Dietterich, R. H. Lathrop, T. Lozano-Perez. Solving the Multiple-Instance Problem with Axis-Parallel Rectangles. *Artificial Intelligence Journal,* 89, 1997.

[3] O. Maron , T. Lozano-Pérez, A framework for multiple-instance learning, *Proc. of the 1997 Conf. on Advances in Neural Information Processing Systems* 10, p.570-576, 1998.

[4] O. Maron & A. L. Ratan, (1998). Multiple-instance learning for natural scene classification. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 341–349).

[5] S. Tong & E. Chang, (2001). Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia* (pp. 107–118).

[6] J. Wang and J. D. Zucker. Solving the multiple-instance problem: a lazy learning approach. *Proc. 17th Int'l Conf. on Machine Learning*, pp. 1119-1125, 2000.

[7] C. Yang and T. Lozano-Perez. Image database retrieval with multiple-instance learning techniques. *Proc. of the 16th Int. Conf. on Data Engineering*, pp.233-243, 2000.

ACKNOWLEDGMENT

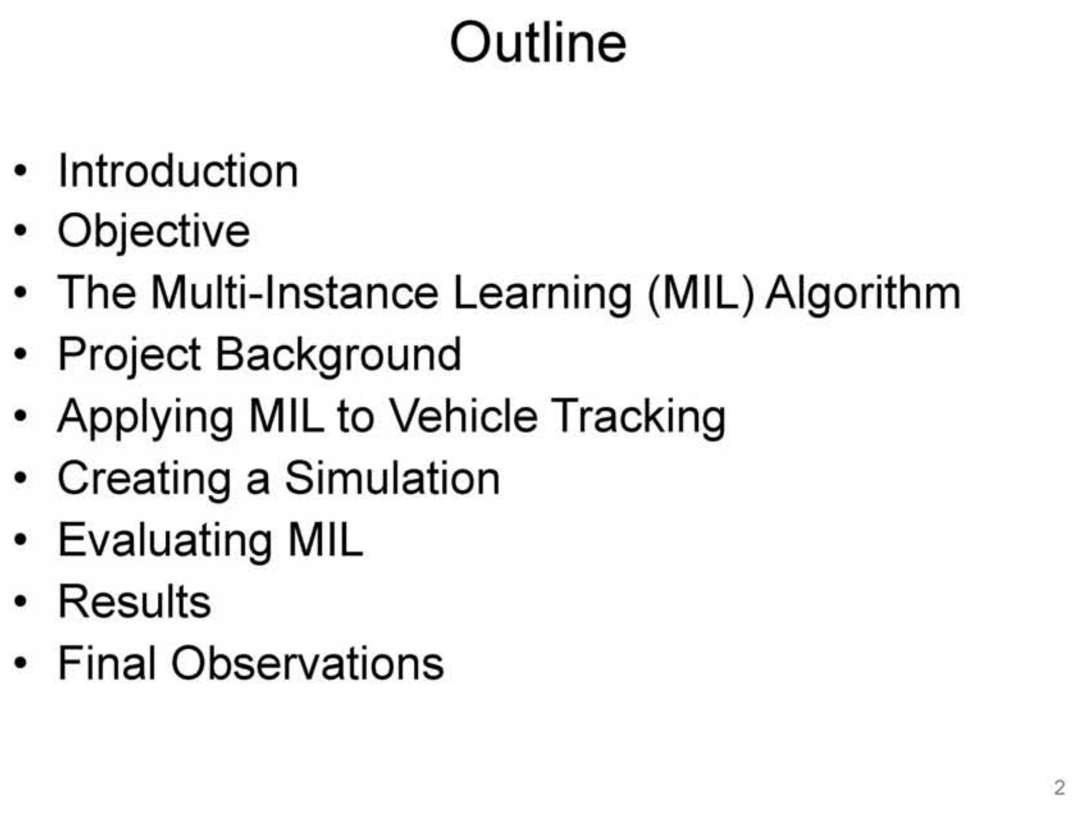
We would like to thank Brian Rigling of SAIC for supporting Mihnea Birisan in this research.

Multi-Instance Learning Models for Automated Support of Analysts in Simulated Surveillance Environments

Mihnea Birisan, Peter Beling

Department of Systems and Information Engineering

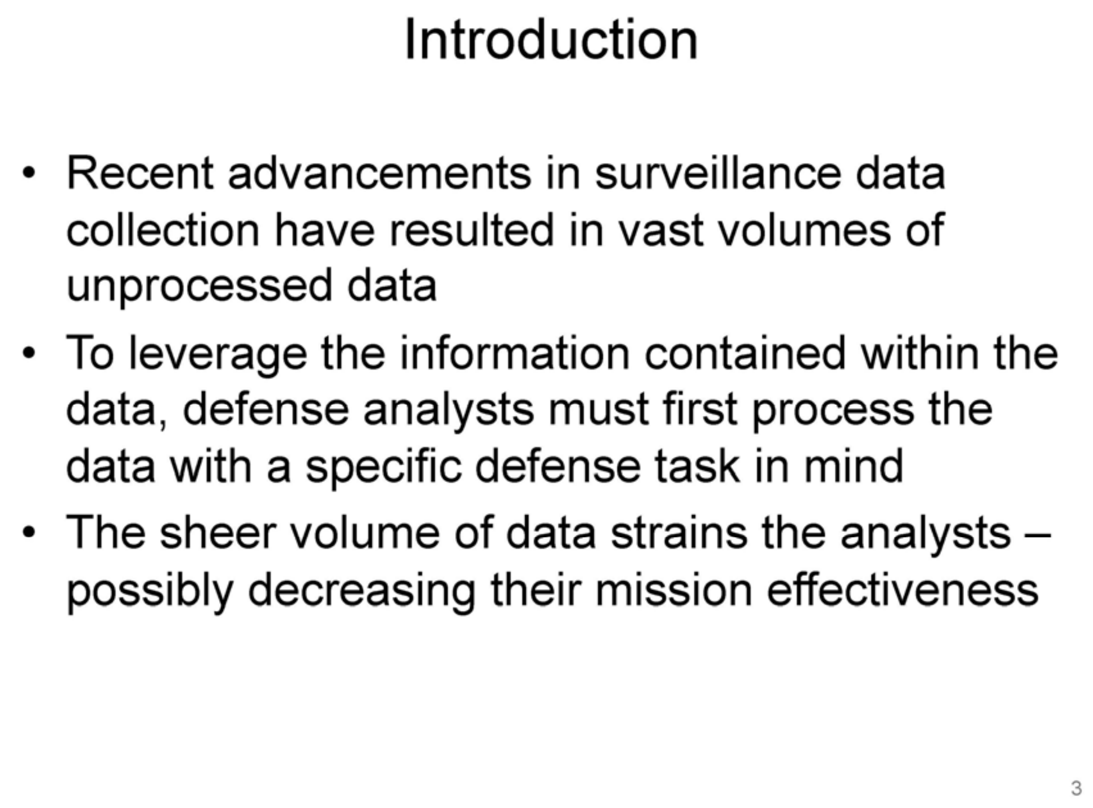
University of Virginia



Outline

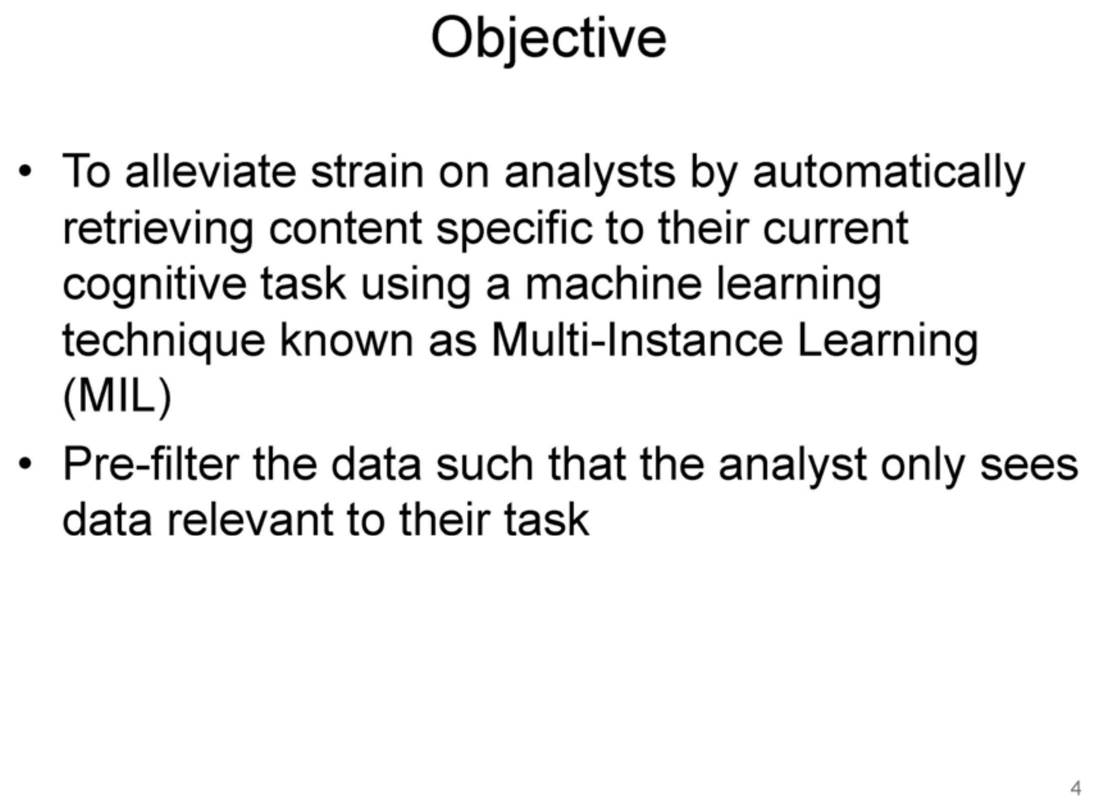
- Introduction
- Objective
- The Multi-Instance Learning (MIL) Algorithm
- Project Background
- Applying MIL to Vehicle Tracking
- Creating a Simulation
- Evaluating MIL
- Results
- Final Observations

2



Introduction

- Recent advancements in surveillance data collection have resulted in vast volumes of unprocessed data
- To leverage the information contained within the data, defense analysts must first process the data with a specific defense task in mind
- The sheer volume of data strains the analysts – possibly decreasing their mission effectiveness

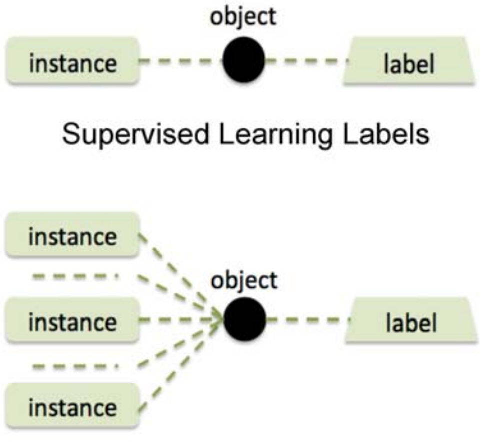


Objective

- To alleviate strain on analysts by automatically retrieving content specific to their current cognitive task using a machine learning technique known as Multi-Instance Learning (MIL)
- Pre-filter the data such that the analyst only sees data relevant to their task

About MIL

- A variation of supervised learning with ambiguity in **labels**
- A **bag** (object) can be represented in multiple ways – know as **instances**
- Each **bag** receives a **positive** or **negative label**, but it is unknown which **instance(s)** triggered the **label**.
- A bag will be labeled
  - positive if at least one instance in the bag is positive
  - negative if all instances in the bag are negative

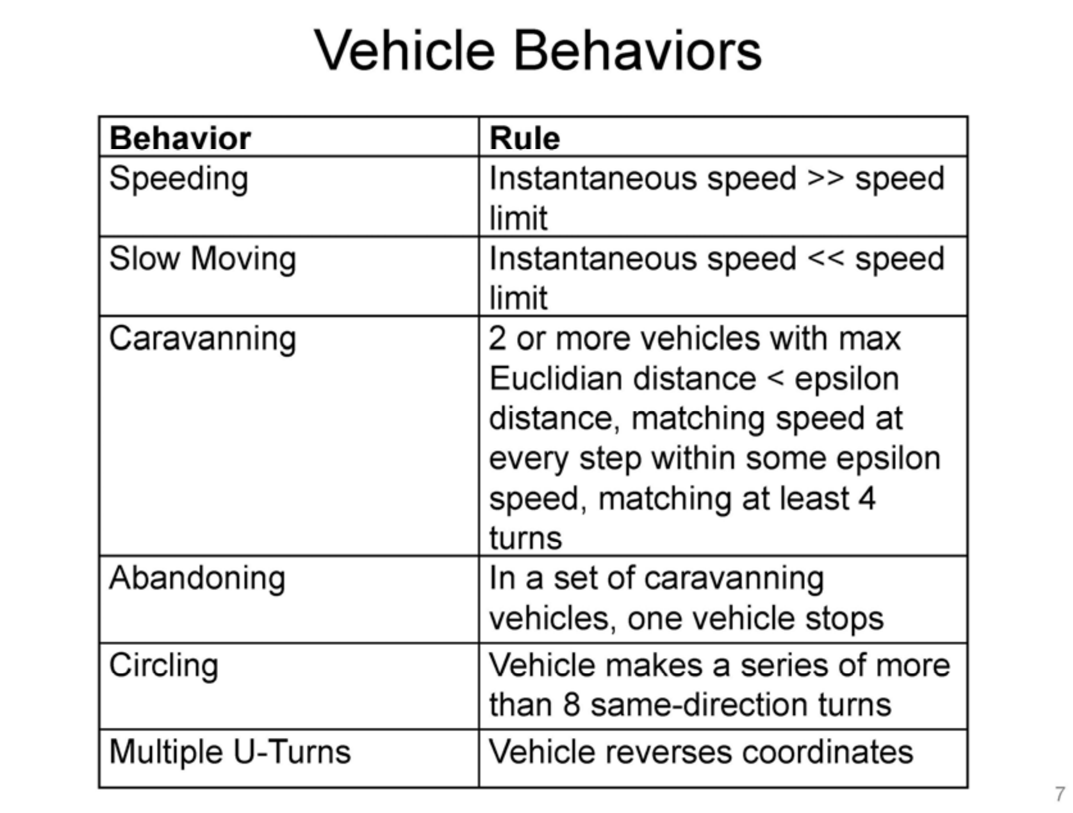


Supervised Learning Labels

Multi-Instance Learning Labels

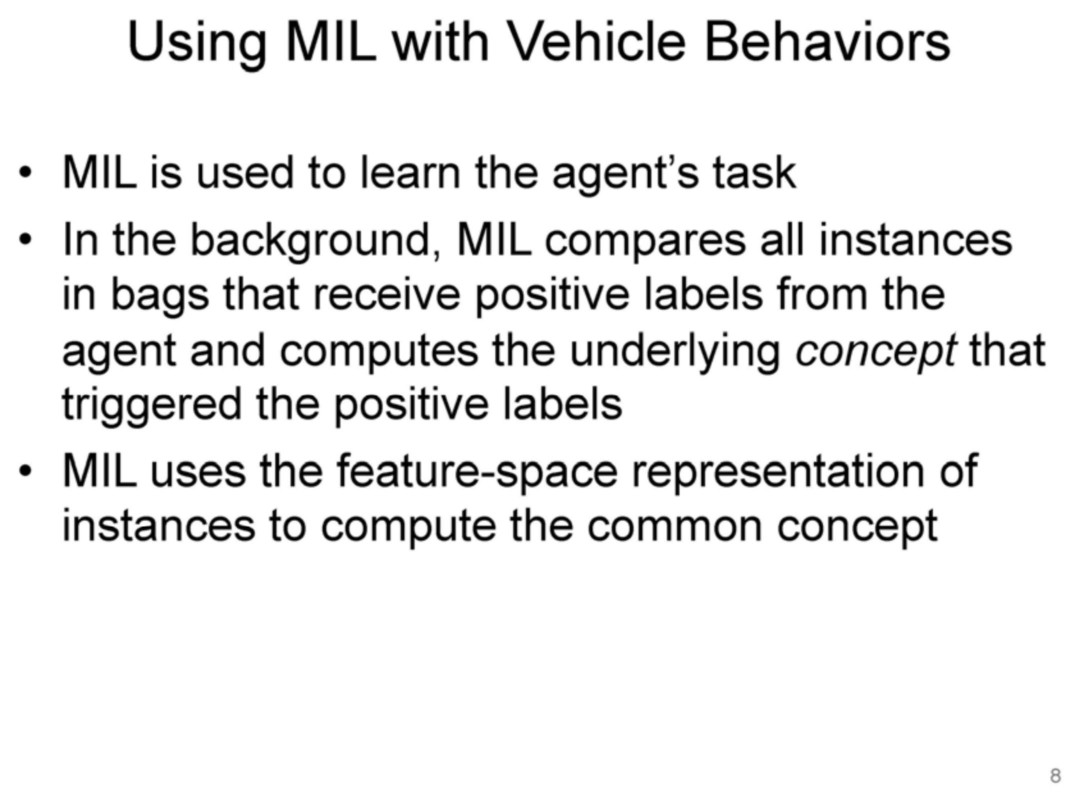
Some Project Background

- Picked vehicle tracking environment as proof-of-concept scenario
- Simulated environment models vehicle tracking activities
- Analyst is replaced by an agent
- Assigned the agent tasks that might be performed by an analyst tracking vehicles based on their behavior



Vehicle Behaviors

Behavior	Rule
Speeding	Instantaneous speed >> speed limit
Slow Moving	Instantaneous speed << speed limit
Caravanning	2 or more vehicles with max Euclidian distance < epsilon distance, matching speed at every step within some epsilon speed, matching at least 4 turns
Abandoning	In a set of caravanning vehicles, one vehicle stops
Circling	Vehicle makes a series of more than 8 same-direction turns
Multiple U-Turns	Vehicle reverses coordinates



Using MIL with Vehicle Behaviors

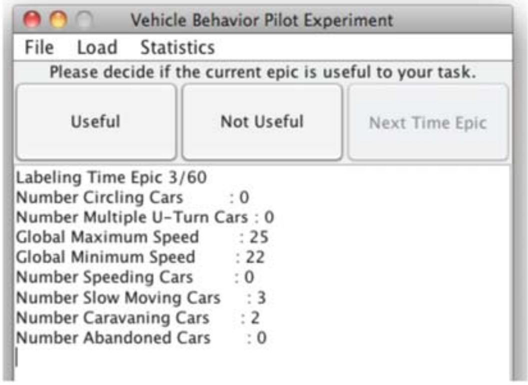
- MIL is used to learn the agent’s task
- In the background, MIL compares all instances in bags that receive positive labels from the agent and computes the underlying *concept* that triggered the positive labels
- MIL uses the feature-space representation of instances to compute the common concept

Feature Space

Feature Name	Type
Speeding Cars Present	Boolean
Number Speeding Cars	Integer
Slow Moving Cars Present	Boolean
Number Slow Moving Cars	Integer
Caravans Present	Boolean
Number Caravans	Integer
Abandoned Cars Present	Boolean
Number Abandoned Cars	Integer
Circling Cars Present	Boolean
Number Circling Cars	Integer
Multiple U-Turn Cars Present	Boolean
Number Multiple U-Turn Cars	Integer
Number Cars	Integer
Number Trucks	Integer
Number Red Vehicles	Integer
Number Blue Vehicles	Integer
Global Maximum Speed	Integer
Global Minimum Speed	Integer

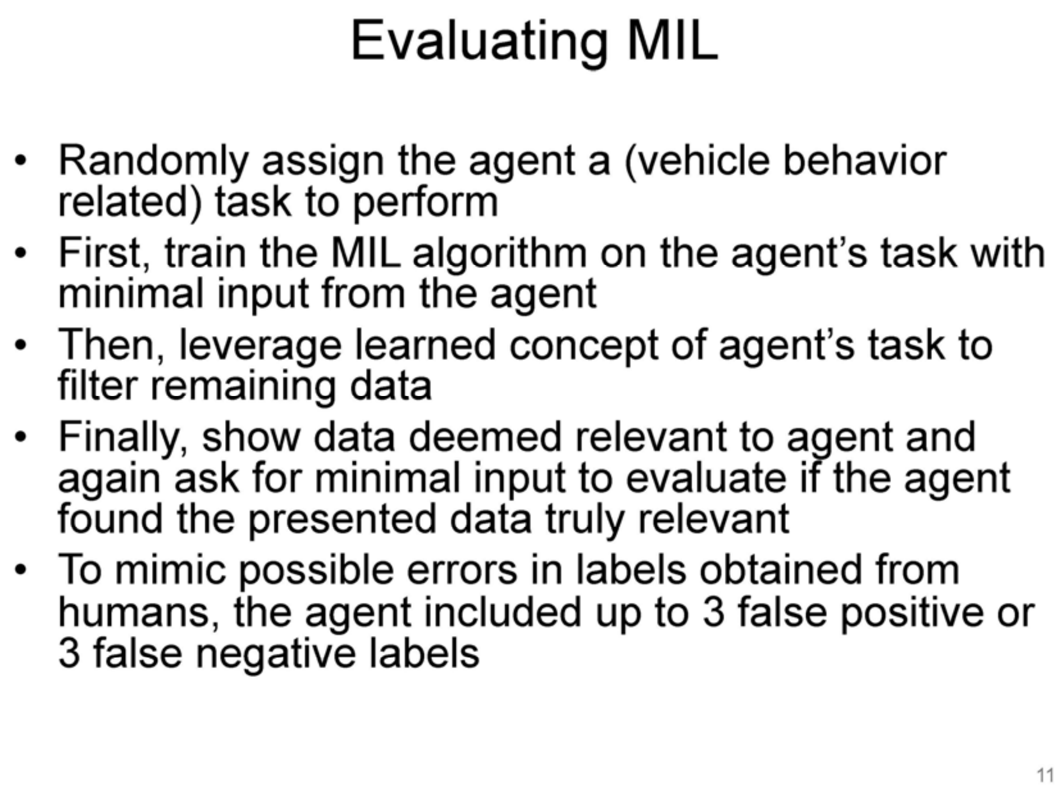
About the Simulation

- Used as a tool to evaluate the MIL algorithm
- Presents agent with randomly picked time epics of vehicle behavior
- Two main stages:
  - Training – MIL works in the background to learn the agent's task based on agent input
    - Agent input – answer to question: Was the information in the past time epic useful in fulfilling the task?
  - Evaluation – MIL filters data based on learned concept of agent's task
    - Simulation subsequently only presents relevant data unless performing an unaided trial



Please decide if the current epic is useful to your task.

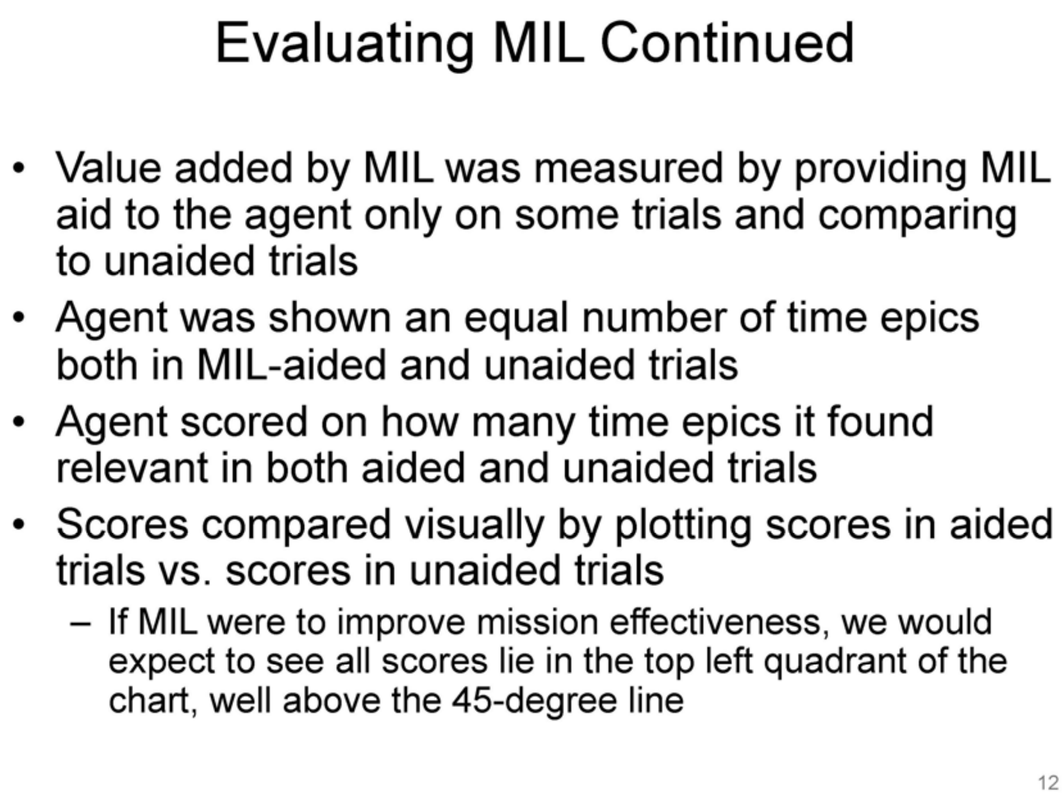
```
Labeling Time Epic 3/60
Number Circling Cars       : 0
Number Multiple U-Turn Cars : 0
Global Maximum Speed        : 25
Global Minimum Speed        : 22
Number Speeding Cars       : 0
Number Slow Moving Cars     : 3
Number Caravaning Cars      : 2
Number Abandoned Cars       : 0
```



Evaluating MIL

- Randomly assign the agent a (vehicle behavior related) task to perform
- First, train the MIL algorithm on the agent's task with minimal input from the agent
- Then, leverage learned concept of agent's task to filter remaining data
- Finally, show data deemed relevant to agent and again ask for minimal input to evaluate if the agent found the presented data truly relevant
- To mimic possible errors in labels obtained from humans, the agent included up to 3 false positive or 3 false negative labels

11



Evaluating MIL Continued

- Value added by MIL was measured by providing MIL aid to the agent only on some trials and comparing to unaided trials
- Agent was shown an equal number of time epics both in MIL-aided and unaided trials
- Agent scored on how many time epics it found relevant in both aided and unaided trials
- Scores compared visually by plotting scores in aided trials vs. scores in unaided trials
  - If MIL were to improve mission effectiveness, we would expect to see all scores lie in the top left quadrant of the chart, well above the 45-degree line

12

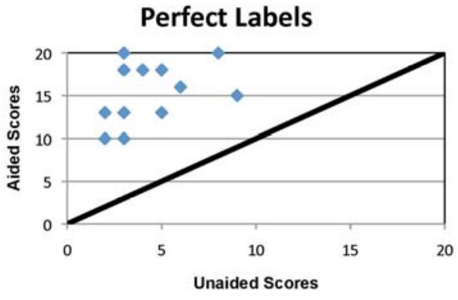
Agent Tasks

Task ID	Task Type	Task
1	Simple	Identify ABANDONED vehicles
2	Simple	Identify CARAVANS
3	Simple	Identify CIRCLING vehicles
4	Simple	Identify SLOW vehicles
5	Simple	Identify SPEEDING vehicles
6	Simple	Identify U-TURN vehicles
7	Composite 1/2	Identify ABANDONED & CARAVANS
8	Composite 1/2	Identify CIRCLING & U-TURNS
9	Composite 1/2	Identify SLOW & SPEEDING
10	Composite 1/2	Identify SPEEDING & CARAVANS
11	Composite 2/3	Identify CARAVANS & ABANDONED & SLOW
12	Composite 2/3	Identify SLOW & CIRCLING & U-TURNS
13	Composite 2/3	Identify SLOW & SPEEDING & U-TURNS
14	Composite 2/3	Identify SPEEDING & CARAVANS & U-TURNS

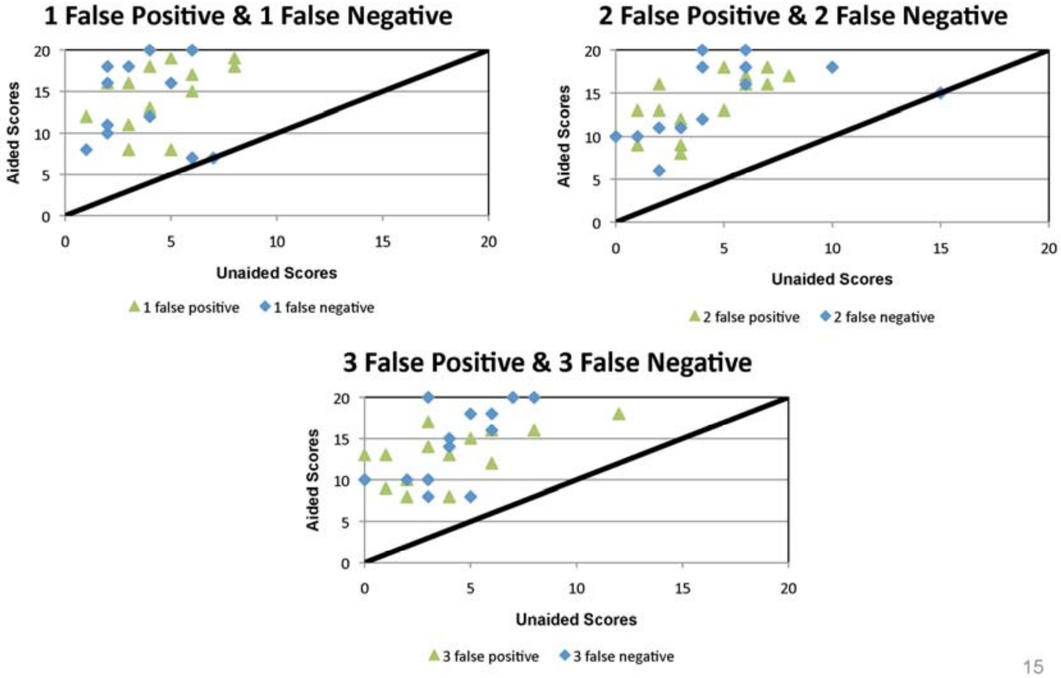
Results

- The agent received higher scores with the aid of MIL!
- Lift shown here implied improved mission effectiveness

	Score	
Label Type	Aided	Unaided
Perfect	15.50	4.57
1 False Positive	14.14	4.35
1 False Negative	13.79	3.71
2 False Positive	13.93	4.21
2 False Negative	14.64	4.79
3 False Positive	13.00	4.07
3 False Negative	14.07	4.14



Results – Type I & Type II Errors

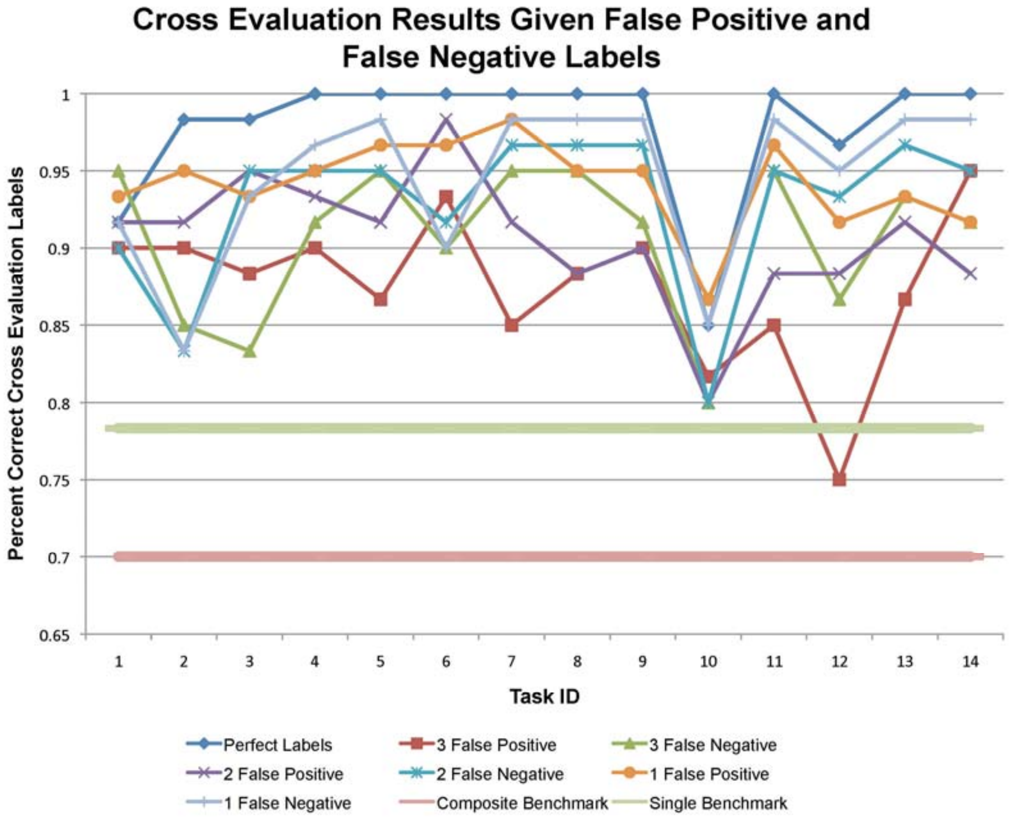


1 False Positive & 1 False Negative

2 False Positive & 2 False Negative

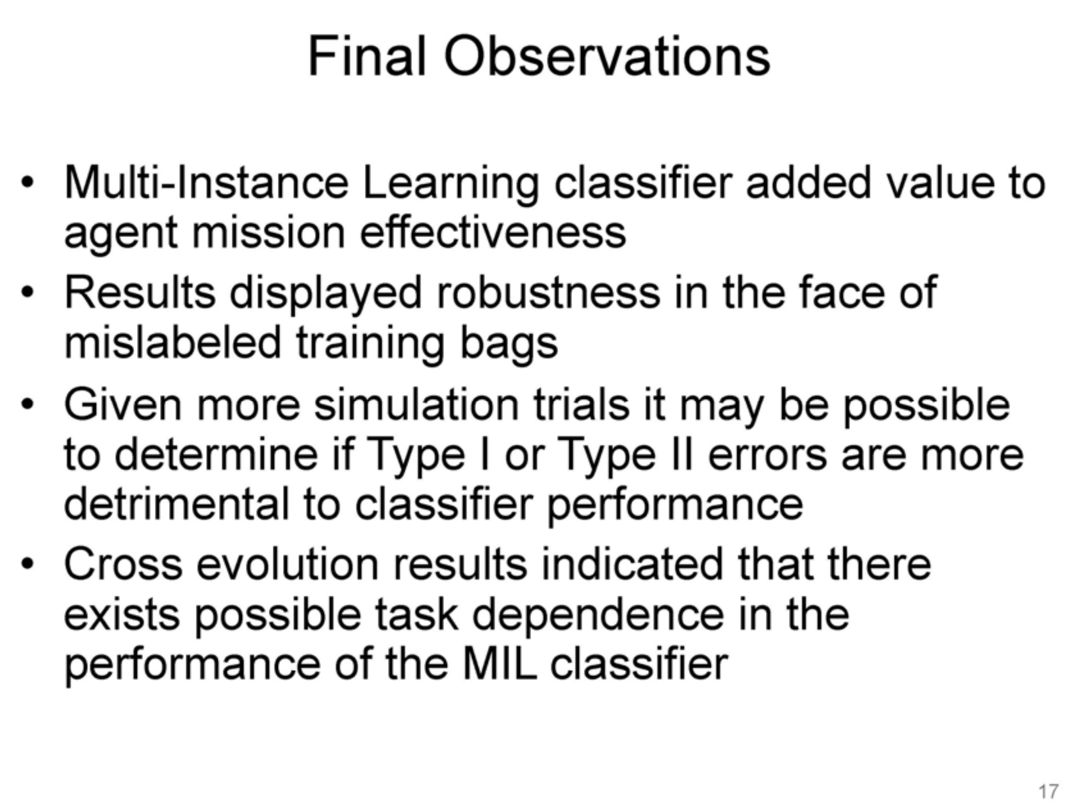
3 False Positive & 3 False Negative

15



Cross Evaluation Results Given False Positive and False Negative Labels

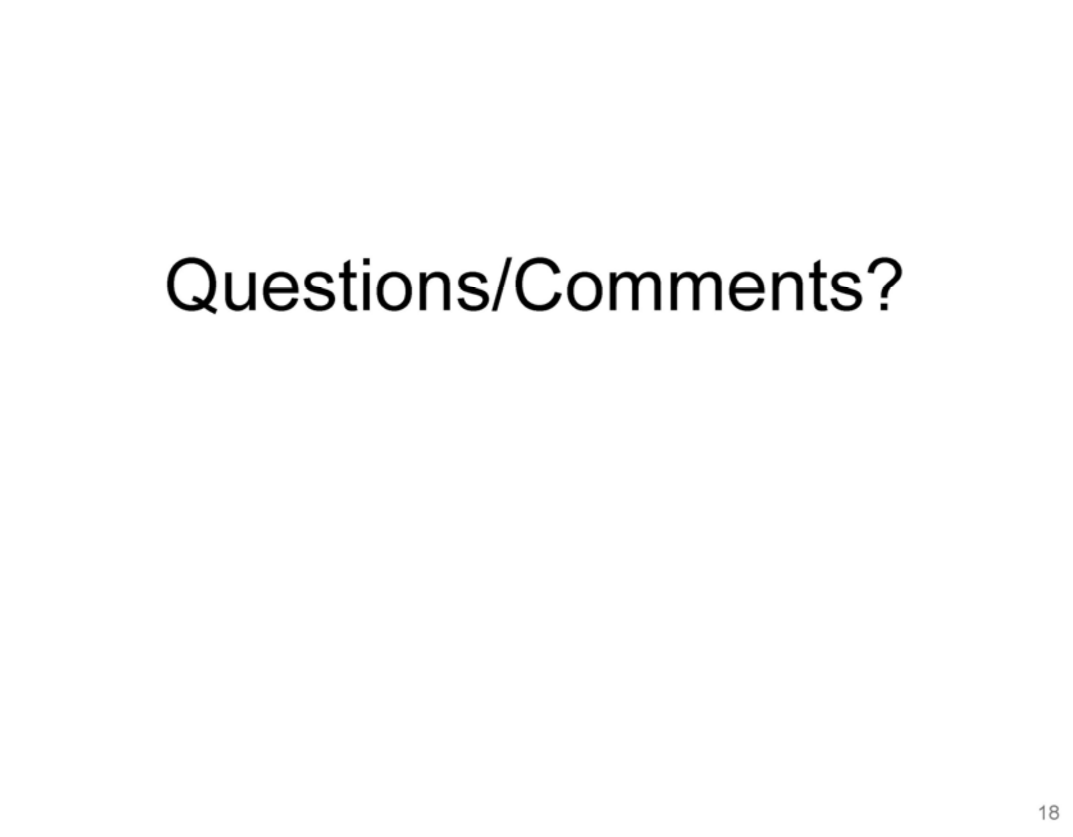
16



Final Observations

- Multi-Instance Learning classifier added value to agent mission effectiveness
- Results displayed robustness in the face of mislabeled training bags
- Given more simulation trials it may be possible to determine if Type I or Type II errors are more detrimental to classifier performance
- Cross evolution results indicated that there exists possible task dependence in the performance of the MIL classifier

17



Questions/Comments?

18