EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

**FACULTE SCIENCES ET TECHNIQUES DE L'INGENIEUR**

LABORATORY OF INTELLIGENT SYSTEMS (LIS)
CH - 1015 LAUSANNE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# SEMESTER PROJECT

| | |
|---|---|
| **Title:** | Bandwidth efficient object recognition for drone swarms |
| **Student(s):** | Marco Zoverალli (IN) |
| **Professor:** | Dario Floreano |
| **Assistant 1:** | Giuseppe Cocco |
| **Assistant 2:** | Fabian Maximilian Schilling |

## Project description:

The project aims at developing a bandwidth efficient distributed object detection system which can be flown on a drone swarm. The system exploits the different points of view of the drones in the swarm to improve object recognition, while keeping the amount of data that is transmitted to the ground station as low as possible. In this way a more efficient use of the limited wireless resources can be achieved. The projects will involve the use of both off-the-shelf neural network algorithms and WiFi communication protocols.

The first part of the project will focus on the setup of the communication network between the communication modules to be mounted on the drones and the ground station.
The second part of the project will focus on the setup of the image capture/object recognition system and some basic onboard image processing. The core part of the project will consist in the implementation and optimization of the detection and communication protocol, which will build upon the modules developed so far.This part will include the evaluation of the system performance in terms of both bandwidth efficiency and detection accuracy.

## Remarks:

You should present a research plan (Gantt chart) to your first assistant before the end of the second week of the project. An intermediate presentation of your project, containing 8 minutes of presentation and 7 minutes of discussion, will be held on October 30, 2018. The goal of this presentation is to briefly summarize the work done so far and discuss a precise plan for the remaining of the project. Your final report should start by the original project description (this page) followed by a one page summary of your work. This summary (single sided A4), should contain the date, laboratory name, project title and type (semester project or master project) followed by the description of the project and 1 or 2 representative figures. In the report, importance will be given to the description of the experiments and to the obtained results. A preliminary version of your report should be given to your first assistant at the latest 10 days before the final hand-in deadline. 2 copies of your final version, signed and dated, should be brought to your first assistant before noon January 11, 2019. A 20 minute project defense, including 5 minutes for discussion, will take place between January 14 and January 25, 2019. You will be graded based on your results, report, final defense and working style. All documents, including the report (source and pdf), summary page and presentations along with the source of your programs should be handed in on a CD on the day of the final defense at the latest.

| Responsible professor: | Responsible assistant: |
|---|---|
| Signature: [illegible] Dario Floreano | Signature: [illegible] Giuseppe Cocco |

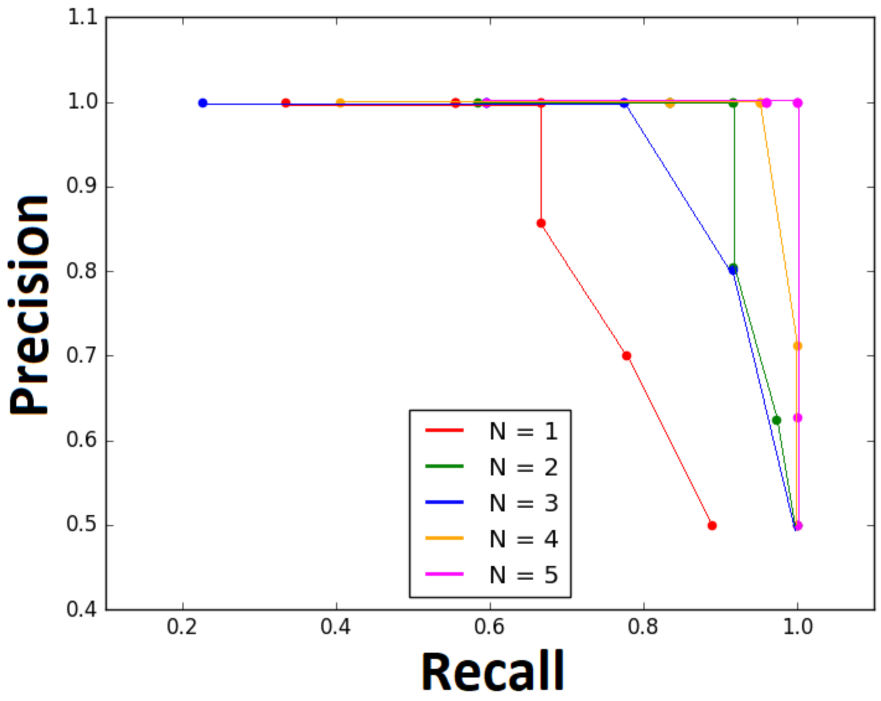
Lausanne, 18 September 2018

# Bandwidth efficient object recognition for drone swarms

MARCO ZOVERALLI
Semester Project at Laboratory of Intelligent Systems
École polytechnique fédérale de Lausanne

January 10, 2019

Getting advantage of multiple devices in order to solve computer vision-related tasks has been a popular choice during the past years. Our main goal is to improve the accuracy of object detection in the context of drone swarms. The massive deployment of drones in next generation mobile networks like 5G calls for an efficient bandwidth usage, which is particularly important in the case of multiple co-located drones such as in drone swarms. For this reason we aim at developing a system that minimizes bandwidth usage while enhancing over a single-drone object recognition system. We implemented a low-complexity and bandwidth-efficient multi-agent system that takes into consideration the multiple viewpoints and uses them to detect the presence or absence of a target object. We successfully tested the system from the image capture to the intra-swarm communication and data fusion and compared our solution to a single-angent system. Our test results show that the performance of the proposed system in terms of *precision* and *recall* quickly improves with the number of drones while keeping complexity and bandwidth usage low, thus helping scalability.



1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
Precision
0.2
0.4
0.6
0.8
1.0
Recall
N = 1
N = 2
N = 3
N = 4
N = 5

***Abstract*—Getting advantage of multiple devices in order to solve computer vision-related tasks has been a popular choice during the past years. Our main goal is to improve the accuracy of object detection in the context of drone swarms. The massive deployment of drones in next generation mobile networks like 5G calls for an efficient bandwidth usage, which is particularly important in the case of multiple co-located drones such as in drone swarms. For this reason we aim at developing a system that minimizes bandwidth usage while enhancing over a single-drone object recognition system. We implemented a low-complexity and bandwidth-efficient multi-agent system that takes into consideration the multiple viewpoints and uses them to detect the presence or absence of a target object. We successfully tested the system from the image capture to the intra-swarm communication and data fusion and compared our solution to a single-angent system. Our test results show that the performance of the proposed system in terms of *precision* and *recall* quickly improves with the number of drones while keeping complexity and bandwidth usage low, thus helping scalability.**

## I. Introduction

Object detection can sometimes suffer from non-optimal viewpoints of cameras or low cameras quality. For instance, if an obstruction hides important features of an image that represents a certain object or if an object which is similar to the target one is present in the scene, the prediction could be wrong, i.e., false positives and false negatives can occur. The goal of this project is to improve object recognition in the context of drones swarms, in terms of precision and recall measures. In particular, we aim at improving a system that, given a specific object, determines the presence/absence of this object. Instead of a single-host system, we exploit the multiple viewpoints available within a swarm. A reliable drone-based distributed object detection system can be either used to trigger an external event (e.g., activating an alarm in case potentially dangerous objects such as suspicious packages or guns are detected in a crowd or also detecting a terrorist suspect in a crowd) or to modify the behavior of the swarm, which is of fundamental importance for it to behave as an autonomous system. We focus on the first setup, which implies having a way to communicate with an external base station that is not part of the swarm, in order to propagate the decision of the group of devices to the external world.
One important constraint that we impose to our solution relates to the usage of bandwidth: it should be kept as low as possible in order to allow scalability and smooth coexistence with systems operating in unlicensed (such as current SMM spectrum) as well as licensed (such as forthcoming 5G networks) systems.

### A. Related Work

Distributed object recognition is a problem that has been studied extensively, from different perspectives.
Rahimpour *et al.* [4] have proposed a distributed object recognition system in the context of wireless camera networks. Their approach aims at extracting features from each camera, sending them to a base station, and delegate the whole object recognition phase to it. The problem with this solution is that a significant amount of bandwidth is consumed – even if their histogram compression and feature selection framework are based on Sparse Non-negative Matrix Factorization. Furthermore, their approach is not suited for autonomous drone swarms, in that data fusion is not performed by the nodes capturing the images, but rather by an external one.
In [5], the authors implemented a swarm-based approach that relies on a cloud system for performing object detection, in a similar fashion as done in [4]. This requires having a stable internet connection and the possibility of transmitting entire images to another system, which is bandwidth consuming. Also, they did not design a system that emits one final vote based on more the opinion of multiple devices.
Medeiros *et al.* [6] have proposed an algorithm to aggregate data coming from different cameras and improve object tracking. Their protocol is based on using a distributed Kalman filter to estimate the position of the object. Unlike the present project, in [6 cooperation is used to improve the tracking of the target object, rather than assessing its presence or absence.
The closest work to our contribution is the paper by Giusti *et al.* [7]. In [7 the authors have proposed a mechanism to perform distributed recognition of hand gestures through a statistical classifier using a distributed consensus protocol to make the nodes of the network converge to a single decision. The decision process takes into account the quality of the prediction of each node. However, the setup and the goal of their solution differ from ours. In the first place, the goal in the two setups is different, in that the aim in 7 is to classify an object that is assumed to be present, while in our case the goal is to reliably decide on the presence of a target object. In the second place, in 7 wheeled robots communicate at close distance, and thus do not suffer from severe bandwidth and interference constraints as in the case of wireless communication, which is exacerbated in the case of drones due to the peculiarities of the air-to-ground propagation channel [8. Finally, in [7] a Support Vector Machine (SVM) is used for classification, while we adopt a neural network, which better fits the problem we want to address.

## II. System Setup

A general target scenario is the one shown in Figure 1. Multiple drones, each one with a camera mounted on top of it, point to the same direction and try to understand whether a target object is present. This may be the case, for instance, of a surveillance system in which drones are located along the borders of a square point towards the center of it. Another case of interest is that of a

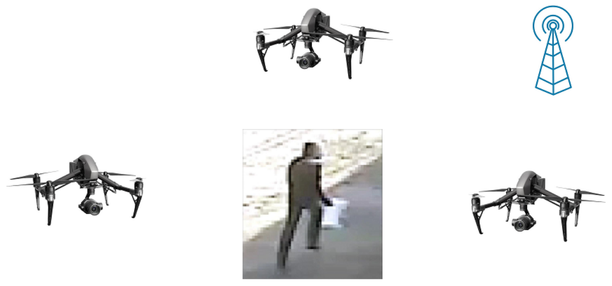


*Figure 1: System setup*

rescue mission in which a drone swarm moves inside a building affected by a fire or leakage of dangerous substances looking for survivors in order to plan a rescue mission. Our approach consists in having these devices communicating in order to share their opinions on the presence or absence of a target object with each other. Each drone runs a local object detection algorithm [1], but the target object is classified as present if and only if at least a minimum number of predictions is positive about its presence. Our design foresees to have a leader that collects all the predictions of the hosts. The leader takes care of the data fusion ands sends the final result to the base station. More details are provided in section IV.
Since the swarm of hosts is considered as an independent system, the communication occurs directly between hosts, without any third device as intermediary, which is the case in most WiFi networks. Therefore, the natural choice is to use an adhoc network for exchanging messages between the network nodes. This, together with the fact that the leader changes periodically, allows to avoid single points of failure in the system.
We now give additional information about the setup of our system and the hardware that was involved.

### A. Hardware

*1) Embedded boards:* The single-board computers that were used in this project were the Odroid XU4. Thanks to the acceptable power of this device [2], it is possible to run relatively complex algorithms, including neural networks, in hundreds of milliseconds. Ubuntu MATE is the chosen operating system, which allows to easily run modern software platforms, such as various Python libraries/toolkits and Go. Go is a programming language designed by Google engineers, whose main characteristic is to be simple and powerful 10. The board has three USB interfaces and an ethernet one. Each drone will have one of these devices attached. Any additional piece of hardware, such as connectivity modules or cameras, will be attached directly to this board.

[1]A pre-trained neural network algorithm: SSD Mobilenet, trained on MSCOCO dataset

[2]https://wiki.odroid.com/odroid-xu4/hardware/hardware

*2) Connectivity:* The WiFi Module 5 was used in order to provide wifi connectivity among the Odroid boards. The choice of this component is supported by the fact that connectivity tests have shown a positive outcome for the adhoc mode too, which is the WiFi mode that we use to make the nodes of the network communicate.

*3) Image Acquisition:* Image capture is performed by the OpenMV Cam M7 camera [3]. It easily allows the capture of images of various sizes and their transfer to the Odroid. Although some on-camera image processing can be done with such camera, due to its relatively low computational power this is done on the Odroid, i.e., the OpenMV camera is used only to capture the images. The connection with the board is done via USB.

## III. Object Detection Setup

As already mentioned, we use neural networks in order to perform object detection. Although these models perform better than several classifiers, we cannot use too complex algorithms because execution time can become an issue if we consider that each drone must do continuous predictions in order to keep its opinion up-to-date and that our single-board computers do not have the computational power of state-of-the-art computer clusters. Therefore, the idea is to compensate the lack of state-of-the-art networks through the presence of several devices trying to detect the same object.
The same holds for cameras: since we have a swarm we can mount cheaper cameras of lower quality and less weight and still get good performance thanks to the different viewpoints.

### A. Image Acquisition

Two parameters that must be tuned during the image acquisition phase are the capture rate and the frame size.

*1) Capture rate:* it defines how often images are captured and sent to the odroid board. In order to have an accurate prediction, it would be nice to have the interval between two consecutive takes as small as possible (i.e., very high rate), so that the local view is always up-to-date. However, the prediction rate of the Odroid XU4 imposes a constraint on this, since the capture rate of the camera must be lower than or equal to the prediction rate. Failure in doing this, would result in sending a higher amount of data than that can be handled, which would result in predictions that are not up to date.

*2) Frame size:* it defines how big the taken images are. Ideally, we would like to capture images whose size is similar to the ones that were used to train the neural network, in order to have better performance. However, a larger frame size implies higher computational time as well, which can have a bad impact in terms of the ”capture rate vs prediction time” issue. Also, it depends on the

[3]https://openmv.io/products/openmv-cam-m7

camera resolution as well. In our setup we tested 64x64 and 128x128 images.

### B. Model Selection and Preliminary Object Detection Tests

Among the available pre-trained models, the zoo models trained with the COCO dataset caught our attention, since they offer several alternatives in terms of performance and execution speed. We tested few of them. At fist glance, it might seem that a classification algorithm would be suitable for our task of detecting the presence of an object. However, classifiers usually handle – and are trained with – simple images with one object. In a real scenario, there would usually be several objects and the target should be detected among them. Traditional classifiers do not handle setups like this.

*1) SSD Mobilenet:* Performance and computation time vary depending on the image size and on the detected object. With 64x64 and 128x128 images, we measured an average computation time on the Odroid boards is around 400 ms. There is no significant difference in terms of computation time, but performance is improved by 128x128 images. Also, since the model was trained with the MSCOCO dataset [4], whose samples are 640x480, better performance would be achieved with 128x128 images, since the size is closer to that of the training set. This should be kept in mind for any future use of our approach: re-training the model might be necessary in order to have better performance. Since our goal was to develop a system which can handle sub-optimal components, we used the pre-trained neural network and used it out-of-the-box. Such mismatch between the image training set and the size of the images used in the system resulted in very accurate predictions for some kinds of objects 2, but not so good ones for others 3. Another consequence was that several false negatives are generated for some of these detections. In our experiments we considered classes of objects that proved to be easily detectable under ideal conditions, i.e, they are detected with high confidence score.
Finally, in order to further motivate our setup, figure 4 shows that even classes that are detected very well in some conditions (see figure 2) can be badly recognized if observation conditions change.

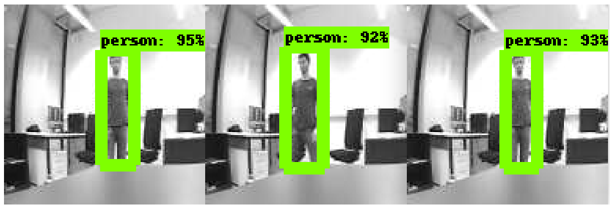


*Figure 2: Preliminary test of detection of people in a noisy environment: the confidence scores are extremely high.*

[4]http://cocodataset.org

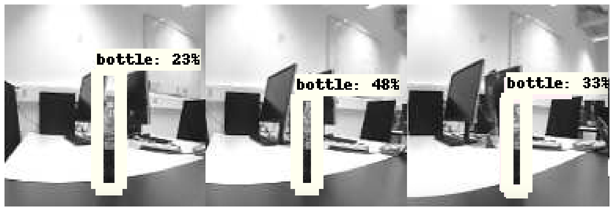


*Figure 3: Preliminary test of detection of bottle in a noisy environment: the confidence scores are lower than the ones that relate to people, although the quality of the view is comparable.*

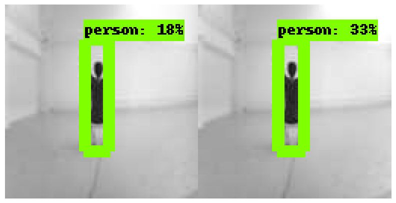


*Figure 4: A person not detected very well.*

*2) Faster R-CNN (Region Convolutional Neural Network:* These models represent the state of the art in terms of accuracy [1]. However, one big issue relates to the long prediction time, which is around 200 ms on state-of-the-art hardware (NVIDIA Tesla K40 GPU Computing Processor Graphic Cards, as mentioned in [1]), and therefore it becomes unfeasable to use them on our hardware, which has much lower computational power.

### C. Measures of Interest

Whenever a prediction is performed, the neural network provides a set of output parameters for each object class. Each prediction contains the following information:

- the class of the object that was localized.
- the confidence score of the prediction.
- the area in which the object is located (i.e. the bounding box)

Unlike from classification algorithms, here the output is not represented by a vector of probabilities whose sum amounts to 1. In fact, neural networks make several predictions on different regions, which relate to different objects and are therefore independend from each other. Specifically, for each region of the image and for each object class the NN outputs a score indicating the likelihood that the given object is present in the considered region. One first thought that we had was that, under some assumptions, the size of the bounding boxes might have given some useful information in order to give proper weights to different predictions. The idea was that, if we assume that an object has symmetric features, having a larger bounding box means being closer to an object. While this might be true, some tests we performed on the training dataset showed that the size of the bounding box has a dependency with the confidence which is not easy to be exploited. This is why we are considering the confidence score as the only value involved in the predictions calculation. As the next section IV shows, we will rely on a mechanism that considers an object as present if a subset of devices (whose size is chosen a priori) considers it as present.

## IV. Proposed System

Designing the protocol was one of the main contributions of this project. Besides being able to improve object detection with respect to a single drone, our goal is also to ensure that little bandwidth is used and that the swarm is able to take a decision as if it was a single, independent, system. The main challenges involved are:

- how a single object detection prediction is obtained and how it is communicated to other hosts
- what kinds of messages are transmitted
- how data is aggregated and combined together
- how the faults of the network nodes are handled
- how the information is transferred for external purposes.

Our protocol is divided in three different phases, that involve significantly different activities:

1) capture the image with the camera, and transfer it to the Odroid
2) perform object detection on the Odroid, and propagate the result to the other hosts of the network
3) aggregate information together in order to determine a final prediction

### A. Consensus Protocol

Whenever a consensus problem is faced, there are few intuitive approaches that define how the nodes cooperate in order to achieve a result. One way consists in having a leader that takes the local predictions from all the nodes in the network, aggregates the results, takes a final decision, and propagates the final decisions to all the nodes in the network (or to an external entity). An advantage of this approach is that the communication scheme is quite simple: all the nodes provide their prediction to a single node, who is the only one that needs to calculate the final result. This also implies that the bandwidth usage is kept low, since only the leader needs to see other nodes' messages. However, this solution requires that the leader is elected dynamically in case of failures (leader election). Furthermore, there should be some criteria to determine the eligibility of nodes to become leaders (e.g. the node that is able to communicate with more hosts). A second approach would be to have every node to receive all the values and computing the (same) solution. While this ensures robustness (especially against faults), it also means that unnecessary work is performed: the result is the same as the previous approach, with the difference there may be several more messages transmitted between hosts. Finally, a third approach consists in having each node to compute part of the final solution and then all the computations have to be aggregated. This approach is potentially very flexible and smart, because it allows to define very complex data fusion mechanisms. However, these mechanisms are usually very complex and guarantee better results only under specific conditions. For instance, many of such systems require a clustering algorithm, which adds complexity and requires communication resources for up-keeping and tracking the rapidly changing network topology and communication conditions of drones swarms. We chose the first approach because of its ease of implementation and it is optimal if there is no packet loss. Since in real systems there are packet losses, and for scalability issues, we switch leader at each round, in order to have some spatial diversity in the network.

### B. Protocol Overview

Our approach got inspired from the family of consensus protocols Paxos 11. However, our usage is different from the typical ones, which often aim to reach **state machine replication**. In fact, we want to get advantage of the hosts' predictions just to determine whether the target object is present or not. As a consequence, many of the constraints of classical consensus protocols are relaxed and do not need to be met. The idea is to define a majority (M) as a fraction of the total number of hosts of the network (N). The majority should express a meaningful lower bound in terms of number positive predictions that are needed in order to trigger a positive final prediction. A high majority implies having less false positives, while a low majority will cause less false negatives. We tested M=N/2+1, which is fair in terms of false positives and false negatives, and M=N/3+1, which favors false positives. Fairness in terms of false positives and negatives (M=N/2+1) would seem the natural choice. However, since the models that we are using sometimes fail at detecting true positives with the provided images (as explained in III-B1), it makes sense to have a system that favors positive detections.

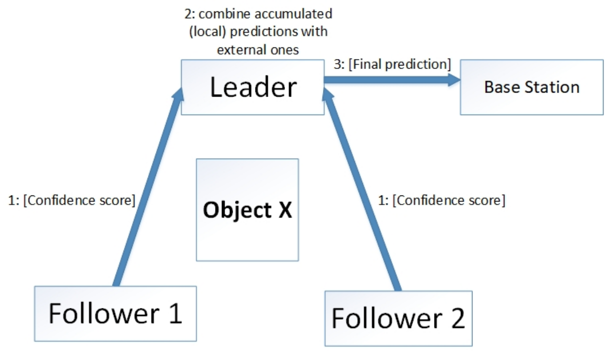


*Figure 5: Inter-host communication*

The decision process takes place in rounds: during a round the leader must take a decision that is based on its local prediction and the other hosts' predictions. The leader probes all the other hosts in order to receive their predictions: the transmission of the probe message is repeated periodically ($T_{probe}$) in order to mitigate the impact of packet losses. Each host (including the leader) has a timeout ($T_{timeout}$) after which, if not enough messages are received, it declares the object as absent (for that round) and it advances to the next round. The timeout period is taken as a multiple of the probe period, i.e., $T_{timeout} = K * T_{probe}$. The leader advances to the next round after either a decision for the current round is taken or the timer expires. This implies three cases:

- Timeout: the object is declared as absent for that round. This occurs if not enough positive predictions are received, either because some message was lost or because the received votes are negative.
- Predictions from all hosts are received, but there is no positive majority: the object is declared as absent for that round.
- A majority of positive predictions is received: the object is declared as present for that round.

A sketch of the protocol is shown in figure 5.

### C. Prediction Obtainment and Inter-process communication

Once the image is captured by the camera, it is immediately transferred to the Odroid, which allows the usage of Linux and the Tensorflow library. This has a set of benefits, including a broad flexibility in terms of choice of object detection models and language for inter-host communication.
Figure 6 summarizes the computations that occur within a single host. First, the Odroid takes care of performing object detection. This algorithm runs over a Python process, which uses Tensorflow. In order to favor modularity and scalability, we decided to separate the communication phase and to have another process handling it. Therefore, the prediction is forwarded to a Go process, which takes care of handling any kind of communication with the other hosts. The local communication occurs via a UNIX socket [5], which is a simple and reliable way to do inter-process communication. The communication with the rest of the networks occurs via UDP.
The reason behind Go's usage is that this language is much more robust, flexible and scalable than Python, and its concurrency [6] and networking features make it an excellent choice for building any kind of distributed system.

[5]http://man7.org/linux/man-pages/man2/socket.2.html

[6]Channels: https://www.sohamkamani.com/blog/2017/08/24/golang-channels-explained/

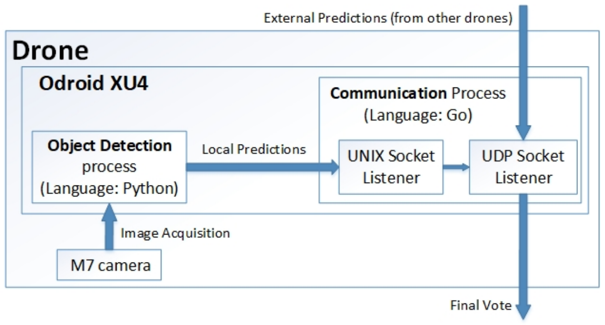


*Figure 6: Intra-host communication: this figure summarizes the leader perspective, since only the leader takes into account the external predictions, computes a final vote, and propagates it. Follower nodes behave similarly, but they do not compute/send any final vote.*

### D. Exchanged Messages

There are different kinds of messages that the hosts of our system handle and propagate. They ensure that a final prediction can be obtained and sent to a base station for external usage.

*1) Probe:* it is used to ask for the status of another host. A host propagating a Probe message expects a Status message as reply. Probes have a field indicating the round ID.

*2) Status:* it is used to propagate the status of the host that forwards the message. The status includes the round in which the sending host is, and the current local prediction that relates to the presence/absence of the target object.

*3) Final Prediction:* it contains the final prediction, after considering the other hosts' predictions as well.

*4) Acknowledgement:* a message for confirming the reception of a Final Prediction message.

*5) Start Round:* a message for forcing another host to start a new round. It contains the ID of the round that has to be started.

### E. Communication Strategy

We decided to use connectionless communication between hosts (UDP). This allows us to have a lighter communication process, because there is no need to establish and mantain communications between hosts. In order to handle faults and packet losses, each node has some timeout (described above) and resynchronization mechanism (described below).
The final prediction is propagated to a base station, and the leader does not advance to the following round until it is sure that the base station received its message. One sample protocol execution is summarized in figure 7.

### F. Propagation frequency

Since each camera sends frames quite frequently (every 400ms) to the Odroid board – which runs the object detection algorithm on each of these received frames – , having each network host propagating its decisions as soon as they are performed could cause an explosion of messages in the network. Therefore, each node propagates its decision after being probed. This implies the following advantages: drones' decisions are more reliable, bandwidth usage is reduced, and the resulting protocol is simpler to handle, test and develop.

### G. Hosts synchronization

All the nodes that vote in the consensus must be sure that they are voting in the same consensus round as the others. In an ideal environment, with no crashes, faults, and delays, the leader should trigger the beginning of a new round for all the other hosts (with a Start Round message), after it takes a decision for the current round. However, the aforementioned events occur in a real environment. This is why the advancement to a new round can be triggered by other messages as well:

- Probe message: the receiver updates its status if the received message has higher Round ID that its one.
- Status message: the receiver updates its status if the received message has higher Round ID that its one. If the received message has lower Round ID, then a probe message with the current Round ID is sent back (here the leader is trying to force the follower to update its round).
- Start Round message: as described above, it forces a host to start a new round.

*Leader Election:* If we assume that all the nodes have good reachability with each other, it is not important who the leader is. Instead, it is important that the whole swarm does not rely always on the same node to be the leader, in order to avoid a single point of failure. For this reason, the leader is simply determined by the Round ID: each leader has a Node ID, and at round i, the leader is the node with Node ID = i % N, where N is the number of hosts in our network.

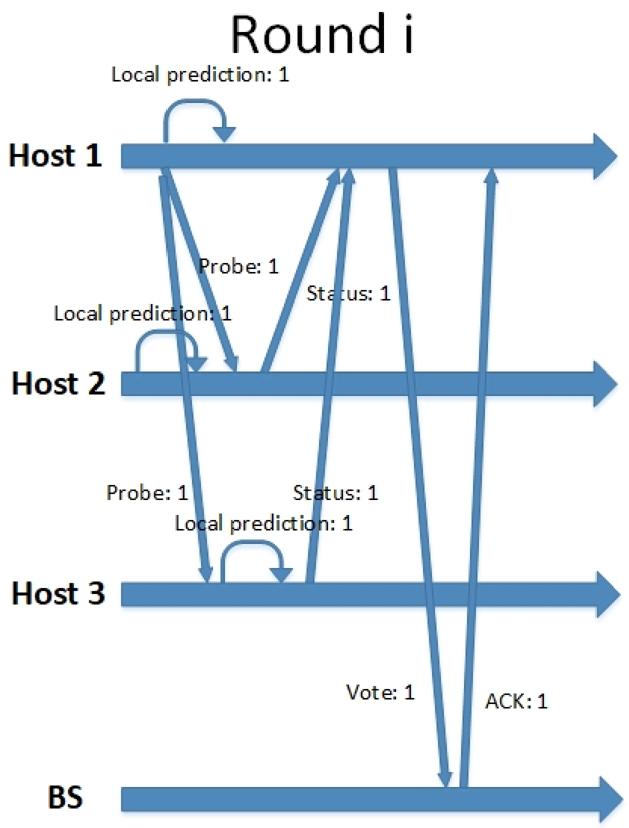


*Figure 7: Example of execution of one round of the protocol, without packet losses.*

## V. Experiments

In this section we describe the results of our approach, by showing experiments performed on a class of objects. We consider a setup that involves a clean environment, without surrounding objects. Our experiments aim to validate different aspects of our solution:

- data fusion mechanism
- communication for the consensus protocol convergence

We separated the two procedures, in order to have a cleaner assessment of the system performance.

### A. Data Fusion Validation

In order to validate the data fusion mechanism we compared our multi-agent system with the single-host system. The measure that we are using relates to the Precision and the Recall. Since Precision = $\frac{TP}{TP+FP}$ and Recall = $\frac{TP}{TP+FN}$ [7], we need two different scenarios (with the same setup, in order to have consistent conditions) to evaluate them. We assumed to have ideal communications, i.e., no packet losses. This allowed us to consider an arbitrary

[7]TP are the true positives, FP are the false positives and FN are the false negatives

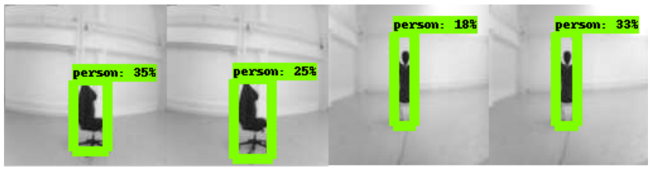


*Figure 8: Setup: we are investigating the presence/absence of a person. In the two leftmost images, a chair is misclassified as a person. In the two rightmost images, a person is not recognized too well.*

number of devices in order to compare the two approaches (single-host system and multi-host system). Let N be the number of hosts that we consider in our system. If N=1, we are in the special case of the single-host system. Let K be the number of different views that we consider for our experiments. For each point of view, we captured some pictures with the camera while it was pointing to the object. The predictions of these pictures are aggregated in the same way as the consensus protocol does for the local prediction process. Specifically, for each picture a confidence score is generated. If the score is above a given threshold, the drone votes in favor of the object being present. Otherwise a negative vote is sent to the leader. More considerations on the thresholds are presented in the following. In our clean environment, the object could be either what we were looking for, or something else (that could be confused with what we were looking for). In this way, we could compare the two solutions in terms of false positives and false negatives. This is how the evaluation was performed:

*Single-host system:* There are K samples (one per viewpoint), and the precision and recall are computed on the basis of the outcomes of the predictions on these samples, in the two distinct scenarios.

*Multi-host system:* Here, the number of samples (from which precision and recall are derived) depends on N. Here, each sample is represented by a set of local predictions from N hosts. This means that for each scenario we have the number of samples that is equal to the combinations of K over N: $\binom{K}{N}$.

Figure 8 shows two views of the setup that we used in order to validate our method: the camera is positioned 3 meters away from the object, and multiple viewpoints are considered. In this particular setup, 9 different viewpoints are considered, on a range of 180 degrees. This means that the angular distance between two viewpoints is 22.5 degrees.

Figure 9 and 10 show the comparison between the single-host system and our multi-host system in the ideal environment, with M=N/2+1 and M=N/3+1, respectively.

It is interesting to notice that in both cases, under some circumstances, we can find advantages with our approach. The general trend for any system is to have a high

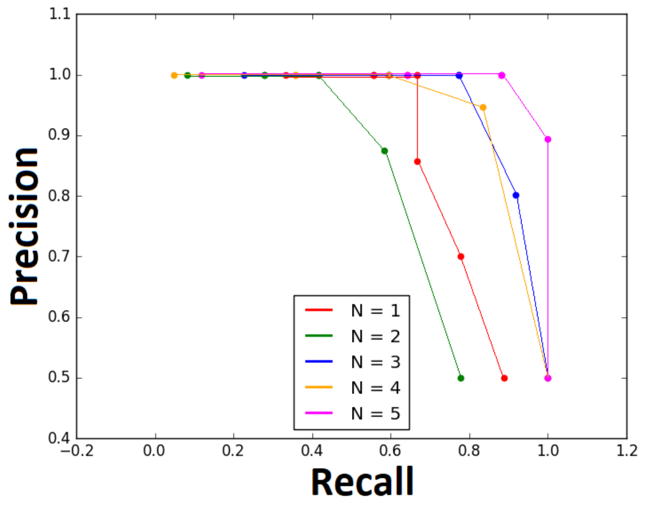


*Figure 9: Majority M = N/2 + 1. This is the number of hosts that need to surpass the threshold with their prediction in order to declare the object as present.*

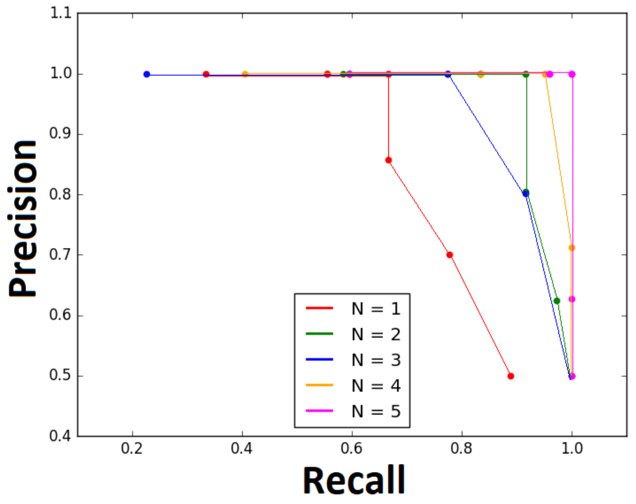


*Figure 10: Majority M = N/3 + 1. This is the number of hosts that need to surpass the threshold with their prediction in order to declare the object as present.*

precision for high thresholds and a high recall for low thresholds. If a high threshold is picked with the single-host system, we can see that the recalls starts degrading as soon as the threshold goes over 0.2 and 0.3 (see figure 11), which are pretty low values. The same reasoning can be done if we decide to pick a low threshold: the recall will benefit from this choice, but the precision is very low for threshold values below 0.3. If we compare this with the multi-host approach, we can see that the choice of the threshold can be more flexible. In fact, if we pick any threshold, we can see that if the number of hosts in the system is appropriate, we can improve the precision (recall) without degrading the recall (precision) too much. For instance, if our threshold is 0.4, the single-host system

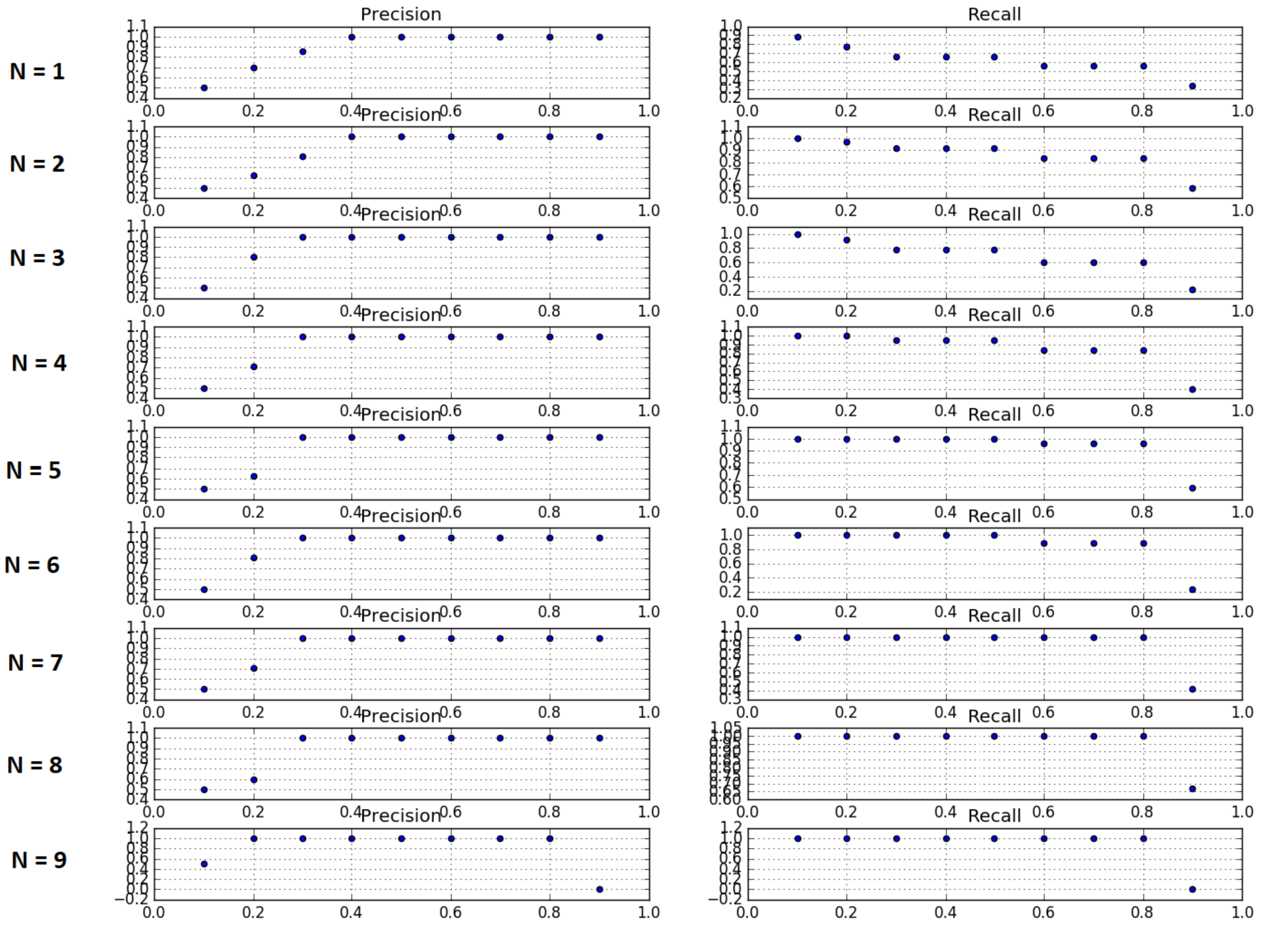


*Figure 11: Same as figure 10, but showing the threshold on the X-axes and the precision/recall on Y-axes (left and right column, respectively).*

will have a high precision. However, if N=1 we can see that the recall is degraded. If we pick a higher N (e.g. N=6), we can see that the recall has a higher value.

### B. Communication and Consensus Test

In order to test the communication and the convergence of the consensus protocol, we used a setup like the one shown in figure 1: there are three drones that point at the center, and they are trying to detect a person. We aim at showing that having a minority of devices (one out three, in this case) that has a non-optimal viewpoint does not impact on the correctness of the system's output.
We firstly put a chair instead of a person, and we saw that two out of three devices detected the absence of a person quite accurately: their confidence score was around 10%. The third drone's confidence score fluctuated between 30% and 40%. During the experiments, we saw that the system was accurately detecting the person as absent although a threshold of 25% was being used: no false positives.
The same procedure was followed again, but with a person with outstretched arms, and the outcome was similar: two drones were correctly detecting the presence of the person (confidence score between 80% and 90%) and one drone with a lower confidence (between 50% and 60%). Again, setting the confidence threshold to 80% did not lead to false negatives.
The test showed that the fully integrated system works properly (image capture and local voting, intra-swarm communication protocol, data fusion and final result transmission).

## VI. Results

Here we further discuss the results that we described in the previous section. We can notice that our system, under some conditions, provides better results than a single-host system. In particular, for both the majority values that we tested we had improvements in terms of precision and recall. The best overall performance is obtained with M=N/3+1. The main reason, as already mentioned, is that the models that we are using have more recall issues than precision's, i.e. in general there are more false negatives than false positives.
It can be noticed that there are some cases in which preci-

sion and recall fall to 0 11. This is caused by the limited amount of data that we have: we just used 9 different positions for our experimental setup. This implies that if there are not enough optimal views, once the number of hosts that participate in the system is sufficiently high, the majority cannot be reached in any way. This is what happened in our case: among the 9 views that we used for the data fusion validation phase, only 3 of them see a person with more than 90% as confidence score. This is why for N >= 6 and M >= 4 (M=N/2+1), precision and recall will always be 0. Similarly, if M=N/3+1, this will occur for N >= 8. However, our results also show that, already with a relatively small number of drones, there is at least one threshold value (often a set of values) for which very good performance in terms of precision and recall are obtained.

### A. *Packet Losses*

The evaluation of data fusion in V-A assumed no packet losses. In figures 12, 13, 14, 15 and 16 we show the same test, but we added a loss probability for each prediction packet. We can see that for very low loss probabilities (0.1 and 0.2), the performance of the system is still acceptable. For higher values, such as 0.5, performance degrades significantly.

## VII. Conclusion

We implemented a system that is able to improve object detection accuracy, by getting advantage of multiple views of objects. Our solution is easy to implement, mantain, and extend.
Part of the future work relates to the integration of different models for object detection, in our system. As mentioned in section III, re-training the model for specific applications may improve performance significantly.
Additional improvements could be done on the communication protocol: it would be interesting to extend the system to have independent clusters of devices within the same network. This would imply implementing some kind of distributed routing protocol.

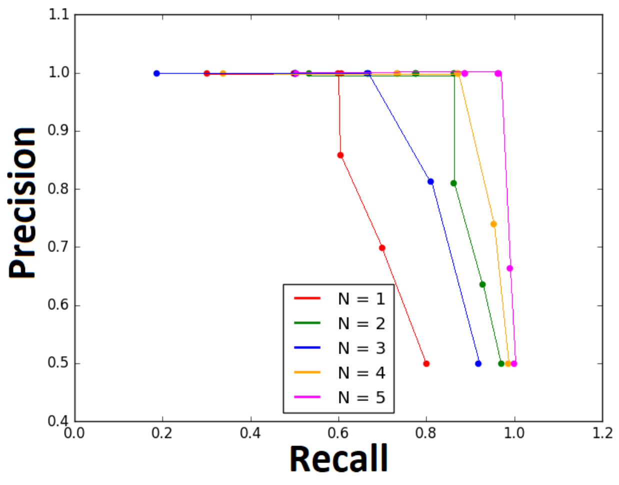


*Figure 12: M=N/3+1, P(loss)=0.1*

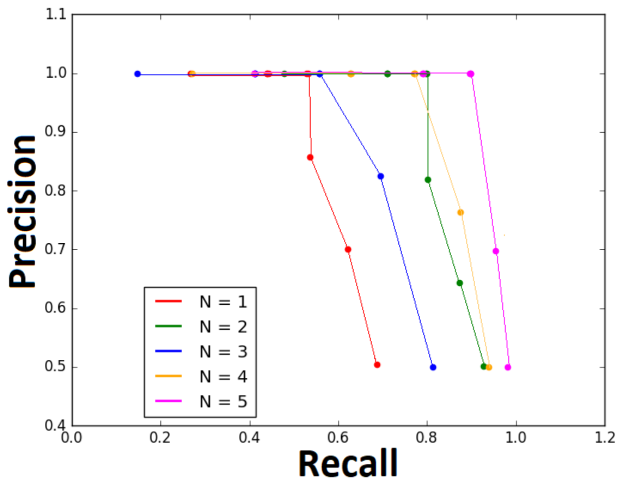


*Figure 13: M=N/3+1, P(loss)=0.2*

## References

[1] S. Ren, K. He, R. Girshick, and J. Sun *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, June 2015.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg *SSD: Single Shot MultiBox Detector*, December 2016.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, April 2017.

[4] A. Rahimpour, A. Taalimi, J. Luo, and H. Qi , *Distributed Object Recognition in Smart Camera Networks*, 2016.

[5] J. Lee, J. Wang, D. Crandall, S. Sabanovic and G. Fox, *Real-Time Object Detection for Unmanned Aerial Vehicles based on Cloud-based Convolutional Neural Networks*, 2017.

[6] H. Medeiros, J. Park, and A. C. Kak, *Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks*, 2008.

[7] A. Giusti, J. Nagi, L. Gambardella, and G. A. Di Caro, *Cooperative Sensing and Recognition by a Swarm of Mobile Robots*, 2012.

[8] B. V. Der Bergh, A. Chiumento and S. Pollin, *LTE in the sky: trading off propagation benefits with interference costs for aerial nodes*, in IEEE Communications Magazine, vol. 54, no. 5, pp. 44-50, May 2016.

[9] T. Mirzayans, N. Parimi, P. Pilarski, C. Backhouse, L. Wyard-Scott, and P. Musilek, *A Swarm-based System for Object Recognition*, 2005.

[10] Kincaid, Jason (November 10, 2009). *Google's Go: A New Programming Language That's Python Meets C++*. TechCrunch. Retrieved January 18, 2010.

[11] L. Lamport (November 10, 2009). *Paxos Made Simple*. November, 2001.

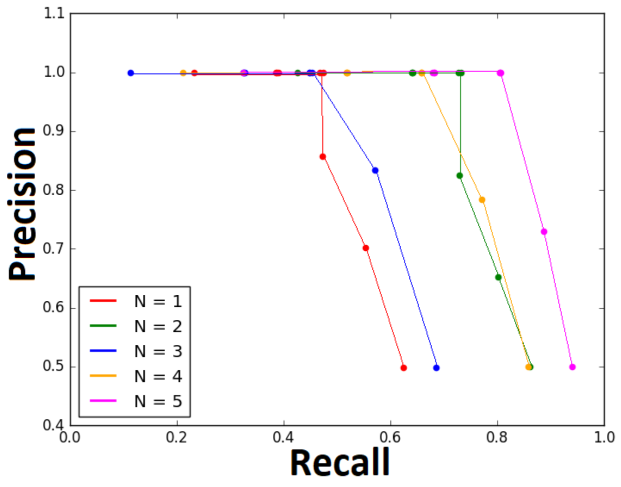


*Figure 14: M=N/3+1, P(loss)=0.3*

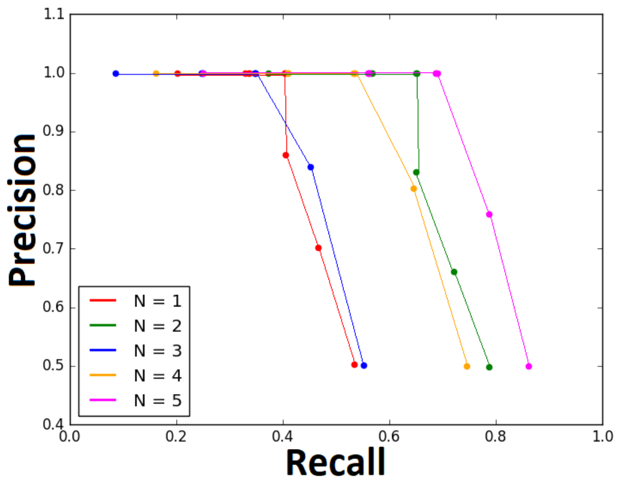


*Figure 15: M=N/3+1, P(loss)=0.4*

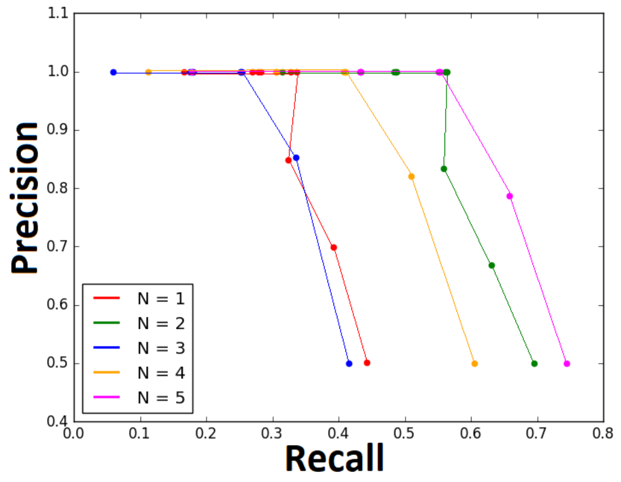


*Figure 16: M=N/3+1, P(loss)=0.5*