Improving Learning in a Mobile Robot using Adversarial Training

Todd W. Flyr[1][a] and Simon Parsons[1,2][b]
[1] *Department of Computer Science, Graduate Center, City University of New York, New York, U.S.A.*
[2] *School of Computer Science, University of Lincoln, Lincoln, U.K.*

Keywords: Mobile Robotics, GANs, Adversarial Training, Machine Learning.

Abstract: This paper reports research training a mobile robot to carry out a simple task. Specifically, we report on experiments in learning to strike a ball to hit a target on the ground. We trained a neural network to control a robot to carry out this task with data from a small number of trials with a physical robot. We compare the results of using this neural network with that of using a neural-network trained with this dataset plus the output of a generative adversarial network (GAN) trained on the same data. We find that the neural network that uses the GAN-generated data provides better performance. This relationship holds as we present the robot with generalized versions of this task. We also find that we can produce acceptable results with an exceptionally small initial dataset. We propose that this is a possible way to solve the "big data" problem, where training a neural network to learn physical tasks requires a large corpus of labeled trial data that can be difficult to obtain.

1 INTRODUCTION

Machine learning is essential if robotics is going to reach its full potential. The alternative, that every detail of what every robot does is individually programmed into it, will just not scale. However, there are many challenges in applying machine learning to robotics. One, in particular, is our focus here. That is the problem of ensuring that there is sufficient data for a robot to learn a task. Generating a single datapoint with a physical robot can be time-consuming, and that makes the data needs of modern deep-learning approaches extremely hard to meet even for simple robot tasks.

Robotics is not alone as an application area of machine learning where it is difficult to find enough data to feed deep methods (though it is an area in which it is particularly hard to collect data). As a result, researchers have come up with a number of ingenious methods for augmenting the datasets that they can collect. For example, in computer vision, it is common to generate additional data by manipulating existing data — flipping and/or cropping images is one common approach (Chatfield et al., 2014).

A more general approach is to use a generative adversarial network (GAN) to create synthetic data that can improve learning, as in (Bowles et al., 2018). We will discuss GANs in more detail below, but a well-trained GAN will generate elements that are indistinguishable from those in the original dataset.

Recently, we (Flyr and Parsons, 2019) demonstrated that it was possible to use a GAN to generate data that can successfully augment the dataset on which a robot is trained to complete a simple task. We compared the performance of a neural network controller learnt from a dataset of robot trials, and a controller learnt from that dataset augmented with the output of a GAN. This was restricted to situations in which the initial conditions for the robot task were close to those in the training set. In other words, little generalization was demonstrated. In this paper we go further, showing that augmenting a dataset of robot trials with data from a GAN improves the performance of the robot on trials that are well outside the scope of the initial dataset. This is possible since the GAN, in effect, can hypothesize a wide range of plausible trial data.

The specific task that we study here is that of a robot learning to strike a ball towards a target on the ground. We use a very simple instance of this task — the robot is a simple mobile base, and the only customization is a metal plate attached perpendicular to the side of the robot. The robot hits the ball by rotating so that the plate strikes it. One might view this as a putting in golf, or as the kind of "kick" used by robot in early iterations of RoboCup. The challenge

[a] https://orcid.org/0000-0003-2011-1552
[b] https://orcid.org/0000-0002-8425-9065

Flyr, T. and Parsons, S.
Improving Learning in a Mobile Robot using Adversarial Training.
DOI: 10.5220/0010107100820089
In *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems (ROBOVIS 2020)*, pages 82-89
ISBN: 978-989-758-479-4

for the robot thus reduces to deciding where to position itself, given the position of ball and target, in order to strike the ball effectively. This is a task that is complex enough to be somewhat challenging if the robot were to be programmed directly, but also simple enough to be feasible without an expensive lab setup.

The remainder of this paper is structured as follows. Section 2 describes related work. Then Section 3 describes the approach that we adopted. Section 4 describes the specific experiments that we carried out, and Section 5 presents the results that we obtained. Section 6 discusses the results while Section 7 concludes with pointers to future work.

2 RELATED WORK

There is a large literature applying machine learning to robotics. Here we briefly survey the work that is most related to our own.

This paper builds on our work which looked at the use of adversarial training for a robot that was learning to hit a ball a target (Flyr and Parsons, 2019) . We tested whether the training set for a neural network controller could be usefully extended by examples generated by a GAN which was trained on the same data set. This paper starts from the same position, and then examines the extent to which data generated by the GAN makes it possible for the controller to generalise, first to scenarios outside the range of the training data, and second to cope with a very minimal training set.

This approach to data augmentation with a GAN relates to (Salimans et al., 2016). In that case, the GAN is used to generate an extra class of data in addition to their canonical list of classes, K. As this GAN generated trial data is the $K+1$th class, it is effectively unlabeled. (Salimans et al., 2016) then show how this can be used to improve training on the labeled data, similar to how semi-supervised learning works via exposing a neural network to related, but unlabeled data. Our approach differs in that we simulate the entire system of inputs and outputs, including a measure of success, and train the discriminator to simply assert whether or not that combination is plausible.

This interest in a controller's ability to make predictions outside the range of its experience is inspired by Bongard's concept of *prospection* (Bongard, 2015). This is the ability to predict future actions, in particular the ability to mentally simulate the actions in contexts that have never been previously encountered.

For us, this connects with the use of the GAN, with its ability to generate data from which the controller can learn standing as a simple form of prospection by providing the controller with imagined scenarios and the action that should be taken in those scenarios.

(Lee and Ryoo, 2017) developed a method for training a large CNN combined with an autoencoder to learn from video of humans manipulating objects with their hands. (Lee and Ryoo, 2017) involves a form of prediction where a sequence of video frames is mapped to 1-2 seconds of further video, thus "predicting" what will be seen a short distance into the future. This ability can be used, for example, to identify the point at which a person carrying a box will want to put it down, allowing a robot to anticipate the need to clear a space for the box.

The approach to prediction from (Lee and Ryoo, 2017) has subsequently been applied to provide a robot with the ability to engage in learning through self simulation. In this work, a controller for a Turtlebot is trained to recognize an object, and decide whether to approach it or avoid it, by using a "dreaming model". The process starts with the robot wandering randomly in the space taking pictures. Then a variational autoencoder with the prediction mechanism from (Lee and Ryoo, 2017) is used to produce sequences of images that make up a realistic path to (or away from) the object. A reinforcement learning algorithm is then used to learn a policy that will take the robot along the relevant path. After the pictures are taken, the whole process is completed without any real-world trials, and so it represents a form of imagining how to achieve a simple goal.

Finally, we note that (Gupta et al., 2018) use a GAN framework to improve models for predicting trajectories when navigating in a crowd. Like our work, (Gupta et al., 2018) shows what can be achieved by using a GAN architecture when learning with a robot. However, unlike our work, there is no physical robot.

3 APPROACH

This section explains the setup for our experimental work. There are three aspects — the physical setup of the robot and the task it is learning, the way that data is collected, and the design and training of the controllers. We discuss these in turn.

3.1 Physical Setup

The physical setup replicates that from our earlier work in (Flyr and Parsons, 2019), though our arenas

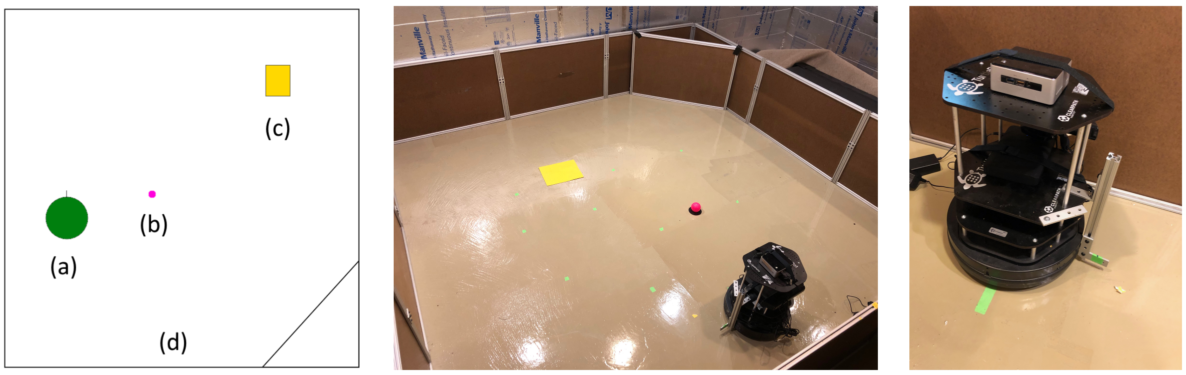


Figure 1: The physical setup. (i) is a diagrammatic view of the arena: *(a)* The turtlebot with the putter head attached to the side; the ball—in an initial position; *(c)* the target—a piece of colored paper, the robot must locate the ball and the target and then properly position itself to line up the shot; *(d)* the arena walls. (ii) is a picture of the the arena. (iii) is a photograph of the modified Turtlebot.

are larger at about 3 metres square.

The physical setup is shown in Figure 1. The target is a coloured piece of paper on the floor of the arena. The robot is a Turtlebot 2, modified by attaching a "putter" to the side of the robot. The robot starts at a position in the arena that is somewhat distant from the ball. It approaches the ball, and strikes the ball the ball by rotating so that the putter comes into contact with it. If the ball passes over the target, we consider that a "hit". Otherwise we consider that the ball missed the target.

3.2 Data Collection

In order to develop a dataset, trial data was collected from a number of scenes. We use the term *scene* to mean a unique robot initial position, ball initial position, and target location. For each scene, we collected data for several trials. A *trial* is an attempt for the robot to hit the ball to the target in a given scene.

Unlike our work in (Flyr and Parsons, 2019), where we used a rule-based system to collect trial data, we controlled the robot by hand during data collection. Data was collected as follows. First, a scene was selected and the robot was driven to a suitable position from which to hit the ball. The robot localized, and recorded its pose, we call this the demonstrated position for the scene. Trials were then run for this scene. Each trial involved the following. The robot localized, and recorded its pose in its initial position. The robot also detected, using its on-board RGB-D camera, the location of the ball and the target. These are the ball and target locations recorded for the trial. The robot then drove to the demonstrated position, using its on-board localization to determine its final pose — this was recorded as the robot's final position. The robot then attempted to hit the ball (by rotating and swinging the putter at the ball). The on-board camera then tracked the ball and determined whether or not the target was hit. If the target was not hit, then the closest approach of the ball to the target was computed and recorded. As a result, the data for each trial consists of the robot's estimates of:

$$\langle target(x,y), ball(x,y),\\ robot\_initial\_pose(position(x,y), yaw))\rangle$$

and which form the input to the neural network, along with the robot's estimates of its final position:

$$\langle position(x,y), yaw\rangle$$

that is the position from which it should attempt to hit the ball. This final position is what the neural network is trained to predict given the input. In order to learn the optimal settings for the task, we added in the terms for the minimum distances of the task goals: $inv_{bt\_min}(x,y)$ and $inv_{rb\_min}(x,y)$ where *bt* is the *ball-target* distance and *rb* is the $robot_{final\_pose} - ball_{initial\_position}$ distance . Because of noise in the robot's estimates, and because the target position was being generated by the human operator, a given scene could generate several different trials. For Experiments 1 and 2 we collected sufficient data that we had several successful and unsuccessful trials for each scene. For Experiments 3 and 4, where (see below) we were interested in establishing a minimal training set, we continued trials until the target was hit, and kept just that trial.

It is worth noting that the only human involvement collecting the training data — and hence in the entire training process — is to pilot the robot to its final position before carrying out trials. Identifying the positions of all the relevant objects is carried out by the robot. As a result, we consider that this work falls into the category of learning by demonstration, with each trial being a single, simple, demonstration.

3.3 Training the Controllers

The process for training the GAN and the MLP controllers were identical to the training in (Flyr and Parsons, 2019).

Each experiment compares two controllers. One is a neural network learnt directly from the training data collected as above what we call the *initial* training data. This neural network is a fully connected feed forward network with two hidden layers. The network topology was $11 \times 120 \times 84 \times 3$. Note that training only considered the yaw (rotation on the z-axis) of the robot. Thus the input to the controller is

$$\langle target(x,y), ball(x,y),\\ robot\_initial\_pose(position(x,y), yaw),\\ inv_{bt\_min}(x,y), inv_{rb\_min}(x,y))\rangle$$

and the output from the neural network is:

$$\langle position(x,y), yaw\rangle$$

This is an estimate of the position from which the robot should hit the ball. We call the controller that is learnt from the initial training data the MLP controller.

The second controller is a neural network with the same architecture as the MLP controller, but which makes use of additional training data. This additional data is generated using a WGAN (Arjovsky et al., 2017). The GAN is used in a standard way — it is made up of a a generator, *G*, and a critic, *D*. The generator, *G*, takes noise as input, and generates data that simulates trials of the robot. *G* is trained against the critic *D*. *D* takes *G*'s outputs as input, and only outputs whether or not each output from *G* looks like a genuine trial run. *G* uses the results from the critic to learn how to output data that looks exactly like the trial data in the initial training set, and at the same time *D* learns, based on the initial training data, how to recognise data that is close to that in the training set.

The generator network had a topology of $8 \times 96 \times 106 \times 106 \times 106 \times 106 \times 96 \times 14$ while the critic had a topology of $14 \times 64 \times 74 \times 74 \times 74 \times 74 \times 64 \times 1$. The generator simulates the entire system, thus the 11 inputs and the 3 outputs are combined for the 14 outputs of the generator.

Once trained, the GAN was used to generate some additional trials that were added to the initial dataset to create an *augmented* training set. Another neural network was then trained using the augmented training set. We call this controller the MLP/GAN. The main difference between Experiments 1 and 2, on one hand, and Experiments 3 and 4, on the other, is in the number of trials in the initial and augmented datasets. We give these details in the description of the individual experiments below.

3.4 Carrying out Experiments

Once the controllers were trained, we carried out trials in which the robot, completely autonomously, attempted to hit the ball to the target. The robot, ball and target were set up as a scene. The robot then self-localized, identified the positions of ball and target, and used its neural network (either MLP or MLP/GAN), to identify the pose from which it should hit the ball. The robot then used the standard ROS navigation stack to move to the pose indicated by the neural network. Once there, the robot swung and tried to hit the ball, tracked the ball, and recorded if there was a hit, and if not recorded the closest approach to the target.

4 EXPERIMENTS

In this section we describe the set of experiments that we carried out.

4.1 Overview

We present results from four experiments. In each experiment we compared two robot controllers, the MLP and MLP/GAN described above. For each experiment, we ran a number of trials.

An example of the data collected for a trial is given in Figure 2a. This shows initial and final positions of the robot (green and blue squares, respectively), the expected final position of the robot (blue star), the initial position of the ball (red circle), the target position (yellow square), and the tracked location of the ball. What is recorded are the earliest point at which the moving ball is observed (magenta circle), around one meter from the robot, where the ball comes to rest (magenta triangle), and the track (blue line). From this we can compute the closest approach of the ball to the center of the target.

Table 1: The composition of the full dataset.

Hit	Hit Ball/Missed Target	Missed Ball	Total
54	90	10	154

The experiments tested two parameters, each of which had two possible values. (Hence four experiments.) One parameter was the size of the dataset that we used to train the controllers. This parameter either had value 154, or had value nine, the minimum number of trials for which we could generate sensible robot behaviour. We call the first of these the *full dataset*, and the second the *minimal dataset*. The other parameter relates to the scenes, and has values *in-range* and *out-of-range*. In range scenes are those in which the initial position of the robot, the initial position of the ball and the position of the target all lie in the range of positions used in constructing the initial training set. In out-of-range scenes, one or more of these positions is allowed to range across the entire arena. The range of positions for the initial training dataset in Experiments 1 and 2 is shown in Figure 2b. The four experiments are thus:

- Experiment 1: Full dataset, in-range.
- Experiment 2: Full dataset, out-of-range.
- Experiment 3: Minimal dataset, in-range.
- Experiment 4: Minimal dataset, out-of-range.

The first experiment replicates our work in (Flyr and Parsons, 2019), though with a much larger dataset — both more scenes and more trials. Looking at the results of the second experiment alongside this should tell us whether a training dataset augmented with data generated by a GAN that was trained on the same dataset can be used to improve the ability of the robot to hit the target when the robot is presented with initial conditions outside the range of conditions that it has seen before. In other words, this is a test of the controller's ability to generalize. The second two experiments then allow us to examine how this ability to generalize is affected by starting from a much reduced — we would argue a bare minimum — dataset.

4.2 Experimental Detail

4.2.1 Experiment 1

For the first experiment,we ran 154 trials with the Turtlebot, as described above. The breakdown of the data, in terms of how many trials resulted in the ball hitting the target, the robot hitting the ball but missing the target, and the robot missing the ball, is given in Table 1.

The MLP was learnt from this data. The GAN was trained on these same 154 trials, and used to generate the data for 200 more. The combined dataset of 354 trials is the augmented training set. The MLP/GAN was then trained on the augmented set. The controllers were then tested on a small number of trials where the scenes were chosen from those used to generate the training data. (This is not the same as saying the controllers were tested on the training data — see the discussion in Section 5). Care was taken that both controllers were tested on a similar pattern of scenes.

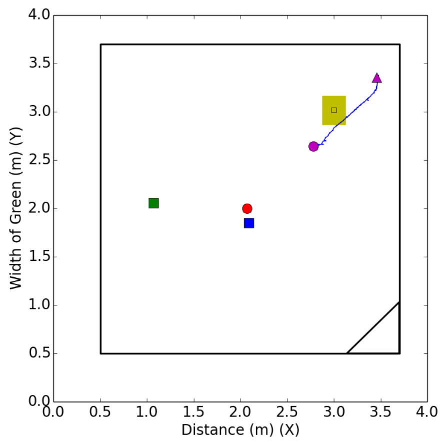
4.2.2 Experiment 2

The second experiment was intended to test whether, in addition to improving the performance of the controller on data similar to that in the initial training set, the MLP/GAN was able to generalize to scenes outside the range of the initial training set. Normally, we would expect that presenting a neural network with inputs that are outside of the ranges on the data on which it was trained would fail, and so it was our expectation that the MLP controller would perform badly in this experiment. However, the use of the GAN can expand the range of possible inputs to some degree via a logical extrapolation from the existing data, so it was our hypothesis that in this case, the MLP/GAN would function better than the MLP. The experiment took the same MLP and MLP/GAN controllers from Experiment 1, and tested them on such scenes — for example scenes where the ball is at the extreme right of the robot and the target is at the extreme left. Again, the pattern of scenes presented to both controllers was similar.

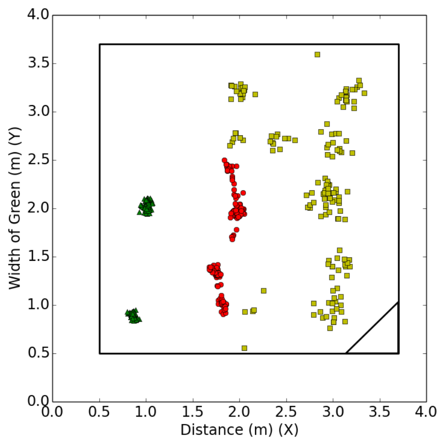
4.2.3 Experiment 3

The third experiment was intended to test how well the GAN can augment the scene dataset starting from an initial training set that was as small as possible. To create a minimal dataset, a single robot position was used, position (1.0, 2.0) (see the coordinate system in Figure 2a). A set of ball and target positions were chosen that covered the breadth of the arena, in particular to ensure that they include every combination of robot movement to the ball (move straight, turn left, turn right) and rotation to face the target from the ball (none, clockwise, counter-clockwise). This resulted in the nine scenes. For each scene, trials were repeated until a hit was recorded, and the data from the trial that led to the hit was added to the initial training set. The initial training set was thus made up of nine trials, each reflecting a different scene. The MLP and MLP/GAN networks were trained as before with the GAN generating an additional 200 scenes so that the augmented dataset was made up of 209 scenes.

For Experiment 3, both controllers were then tested using the same nine scenes. Each scene was used eight times to create 72 trials.



(a)



(b)

Figure 2: Examples of data from our experiments: (a) data recorded in a trial, and (b) scenes for some of the experiments. In (a), the initial position of the robot is in green, and its final position is in blue. The initial position of the ball is in red, the blue line indicates where the ball has been tracked — from the magenta circle to the magenta triangle. The target is the big yellow square. In (b), robot positions are green triangles, ball positions are red circles and target positions are yellow squares.

4.2.4 Experiment 4

The fourth experiment was intended to test generalization in the minimal dataset scenario. The MLP and MLP/GAN controllers from Experiment 3 were again tested by making two trials from each of eight scenes, just as in Experiment 3. However, these scenes were novel, that is they were not included in the training data. Some scenes were in the range of the training data and other scenes were outside that range.

5 RESULTS

5.1 Experiments 1 and 2

The first two sections of Table 2 summarize the results of the experiments. For both the MLP and MLP/GAN controllers, we report the total number of trials (second column), the number of times the target was hit (third column), the number of times that the ball was hit but the target was missed (fourth column), and the number of times that the ball was missed (ninth column). When the ball was hit but the target was missed, we report the average and median closest approach that the robot made to the ball (fifth and sixth columns) and that the ball made to the target (seventh and eighth columns). When the ball was missed, we report the average and median distance of the robot from the ball (tenth and eleventh columns).

The results of Experiment 1 suggest improved performance for the MLP/GAN. While the MLP controller is not able to hit the target at all, the MLP/GAN manages to do this around 30% of the time. When hitting the ball but not the target, the MLP/GAN hits the ball closer, on average, to the target. However, when the robot misses the ball, the MLP controller has it come closer to the ball, on average, than the MLP/GAN. The median approach to robot to ball is lower for the MLP than the MLP/GAN. We carried out Barnard's exact test on the numbers of hits and misses achieved by the MLP and MLP/GAN. This produces a p-value of $P = 0.1163$, indicating a probability of 0.8837 that the MLP/GAN performs better on this measure.

The scenes used for the trials in Table 2 were taken from the set of scenes that were used in generating the training data. Consequently, these results might be considered to have been obtained from the training data. However, using a scene from the training data to test the performance of the controller learned from the training data does not mean the controller is being presented with exactly the same instance that it was trained on. It is considerably different than, for example, testing a neural network-based vision system on one of the images that it was trained on. This is because the scene only describes the physical setup at the start of the trial — it identifies the location of the robot, ball and target as positioned by the experimenters. It does not exactly fix the data that is fed into the controller, since this is computed by the robot from its localization component, and data extracted from its camera. Though we would expect that the data generated by the robot from two instances of the same scene to be close, the noise means that two instances will generate different data. As a result, we argue that testing with in-range scenes is not the same as using training data for testing, but equally does tell us much about the controllers' ability to generalize. What it does allow us to show, is that in terms of hitting the target, which is the task it is trained for, the MLP/GAN likely performs better than the MLP, exactly as we might expect.

The results of Experiment 2 (Table 2, second section) extend our work in (Flyr and Parsons, 2019), showing that the MLP/GAN is able to generalize to scenes that are outside the range of the training set. While it does not manage to hit the target, the MLP/GAN controller is able to hit the ball a few times while MLP controller, as one might expect, was not able to hit the ball in eight attempts. Comparing the counts of hits and misses by the two controllers using Barnard's exact test gives $P = 1.0$ as those counts were identical. But comparing whether or not the robot hit the ball gives $P = 0.2118$ as the MLP/GAN actually hit the ball twice. Thus it is likely that there is a measurable improvement in generalization from the use of the GAN even in this more extreme scenario.

5.2 Experiments 3 and 4

The results of Experiments 3 and 4 are given in the second half of Table 2.

Experiment 3 shows that both MLP and MLP/GAN controllers outperform their counterparts from Experiments 1 and 2. Further, in Experiment 3, as in Experiment 1, the MLP/GAN controller performs much better than the MLP controller, on the same metrics — it hits the target more often, and when it misses the target, the mean approach is closer (however, the median approach of the robot to the ball is again slightly higher). Comparing counts of hits and misses by the two controllers using Barnard's exact test gives $P = 0.0383$ so the difference is statistically significant. While again, especially in the light of Experiment 1, this might be expected, it is notable that even when trained with so little data, the GAN can generate additional data that improves the performance of the network trained from it.

Experiment 4 shows the performance of the two controllers on novel scenes. Here the performance of the MLP/GAN in the novel out of range scenes is marginally worse than the non-augmented MLP. Comparing counts of hits and misses by both controllers yields $P = 1.0$, as in Exp 2 — there is no difference in performance. Indeed, while both controllers had an equal number of hits on the target, the MLP/GAN controller missed the ball more frequently, and only outperformed the MLP in the sense that it came closer, on average, to the target in the "hit ball/missed target" category, a difference that is not statistically significant. However, it is worth noting that both controllers show good generalization here, hitting the target more than a third of the time even though ball, target, or both are positioned outside the range of locations seen in the training data. We discuss why the MLP/GAN did not perform better in Section 6 below.

6 DISCUSSION

While the previous section shows that GAN augmentation improves that robot's ability to learn the task, we do not suppose that what we have is necessarily the best solution. But it does illustrate how GAN augmentation may improve performance on a task for which a neural network is a good solution apart for the need of a large number of physical trials. Furthermore, with a carefully tailored dataset, as in Experiment 3, we can obtain reasonable performance for what is quite hard task when all the noise is taken into account. This suggests that it is possible to train a mobile robot on a physical task when it is difficult or expensive to obtain a large corpus of training data.

It is interesting to reflect on the question of what the GAN is actually generating. Upon review, the GAN generated trials follow the same general pattern of the physical trials.

However, the GAN generates some trials that stretch the dimensions of each element of the scene, while maintaining the proportional relationships between these elements established in the training data. IT also "flips" some trials into the negative X direction. These additional trials are sufficient to improve performance. In Experiment 2, for example, the flipped trials expanded the range enough to enable the robot to at least hit the ball when placed in a trial which was well outside of the scene data.

A follow-up question is why the MLP/GAN did not perform well in Experiment 4. A review of the GAN generated data for Experiment 4 showed a form of mode collapse. It seems likely that this explains the poorer performance of the MLP/GAN in out-of-range scenes.

7 CONCLUSIONS

While the work presented here gives clear evidence that the GAN-augmentation of a small dataset composed of physical trials can improve performance, there are some possibilities for improvements. (Salimans et al., 2016) suggests one avenue, which is

Table 2: Results for all experiments. Experiment 1: Full dataset, in-range scenes. Experiment 2: Full dataset, out-of-range scenes. Experiment 3: Minimal dataset, in-range scenes. Experiment 4: Minimal dataset, out of range scenes. The subscript *rb* indicates robot/ball and *t* indicates target.

Type	Total	Hit Target	Hit Ball/Missed Target					Missed Ball		
	#	#	#	$Mean_{rb}$	Med_{rb}	$Mean_t$	Med_t	#	$Mean_{rb}$	Med_{rb}
Experiment 1										
MLP	9	0	5	0.1438	0.1404	0.4504	0.4607	4	0.1375	0.1287
MLP/GAN	11	3	4	0.2023	0.2022	0.3246	0.3772	4	0.2069	0.2382
Experiment 2										
MLP	8	0	0	-	-	-	-	8	0.2985	0.3312
MLP/GAN	8	0	2	0.2816	0.2816	0.5370	0.5370	6	0.3022	0.3395
Experiment 3										
MLP	36	12	11	0.2106	0.1990	0.3368	0.2989	13	0.2467	0.2785
MLP/GAN	36	21	7	0.2254	0.2223	0.3854	0.3321	8	0.2524	0.2432
Experiment 4										
MLP	16	6	4	0.1791	0.1898	0.3399	0.2624	6	0.43	0.3977
MLP/GAN	16	6	1	0.2128*	0.2128*	0.3039*	0.3039*	9	0.3439	0.2427

*The mean and median results match as there is only a single trial where the ball was hit but the target was missed.

to train the generator on an intermediate layer of the discriminator, which is the actual features of the dataset, as opposed to a singular output that inadequately combines those features.

Another line of inquiry is to use a more recent GAN architecture. We picked the WGAN (Arjovsky et al., 2017) to minimize mode-collapse. However, mode-collapse was still a problem in Experiment 4. A natural response is to use a GAN such as that described in (Gulrajani et al., 2017), which incorporates further technical improvements.

Regardless of these future developments, we believe that we have shown GAN-augmentation is a viable path for developing robust controllers for physical robots in scenarios where a sufficient number of the physical trials is difficult to obtain.

ACKNOWLEDGEMENTS

These experiments were carried out while Todd Flyr was a visiting student at King's College London. We are grateful to King's for hosting him, and for allowing us to use the robotics facilities in the Department of Engineering.

REFERENCES

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia. PMLR.

Bongard, J. C. (2015). Using robots to investigate the evolution of adaptive behavior. *Current Opinion in Behavioral Sciences*, 6:168–173.

Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw, J., and Rueckert, D. (2018). GAN augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

Flyr, T. W. and Parsons, S. (2019). Towards adversarial training for mobile robots. In *Proceedings of the 20th Towards Autonomous Robotic Systems Conference*, London.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of Wasserstein GANs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social GAN: socially acceptable trajectories with generative adversarial networks. *CoRR*, abs/1803.10892.

Lee, J. and Ryoo, M. S. (2017). Learning robot activities from first-person human videos using convolutional future regression. *CoRR*, abs/1703.01040.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. (2016). Improved techniques for training GANs. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc.