Deep Reinforcement Learning Control of Hand-Eye Coordination with a Software Retina

Lewis Campbell Boyd
School of Computer Science
University of Glasgow
Glasgow, United Kingdom
l.boyd@strath.ac.uk

Vanja Popovic
School of Computer Science
University of Glasgow
Glasgow, United Kingdom
vanja.popovic@strath.ac.uk

Jan Paul Siebert
School of Computer Science
University of Glasgow
Glasgow, United Kingdom
Paul.Siebert@glasgow.ac.uk

***Abstract*—Deep Reinforcement Learning (DRL) has gained much attention for solving robotic hand-eye coordination tasks from raw pixel values. Despite promising results, training agents using images is hardware intensive often requiring millions of training steps to converge incurring long training times and increased risk of wear and tear on the robot. To speed up training, images are often cropped and downscaled resulting in a smaller field of view and loss of valuable high-frequency data. In this paper, we propose training the vision system using supervised learning prior to training robotic actuation using Deep Deterministic Policy Gradient (DDPG). The vision system uses a *software retina*, based on the mammalian retino-cortical transform, to preprocess full-size images to compress image data while preserving the full field of view and high-frequency visual information around the fixation point prior to processing by a Deep Convolutional Neural Network (DCNN) to extract visual state information. Using the vision system to preprocess the environment improves the agent's sample complexity and network update speed leading to significantly faster training with reduced image data loss. Our method is used to train a DRL system to control a real Baxter robot's arm, processing full-size images captured by an in-wrist camera to locate an object on a table and centre the camera over it by actuating the robot arm.**

***Index Terms*—Software Retina Preprocessor, Reinforcement Learning, Robotic Vision, CNN**

I. Introduction

Hand-eye coordination is the fundamental skill of combining visual processing with motor control to manipulate objects. The recent successes of deep learning have motivated an increasing interest in developing autonomous robots capable of more sophisticated behaviours [1], [2]. To facilitate this ability, it is necessary to have an efficient and reliable hand-eye coordination system that can carry out basic tasks such as reaching and grasping. Using DRL to train such systems is an attractive prospect for its ability to jointly learn the visual processing and robotic actuation skills from a single reward function.

In practice, training DRL agents to control robotic systems using raw pixel data is a challenging problem. Image data is a high dimensional state space for agents and the inclusion of convolutional layers further increases the number of parameters to learn. Robotic problems tend to involve controlling large numbers of joints giving a high dimensional continuous action space [3]. Training DRL agents in complex state and action spaces with bigger architectures often requires larger amounts of data, careful reward engineering and multiple training attempts to achieve agents with reasonable performance [3], [4].

One approach to easing the difficulty of vision based DRL tasks is to reduce the state space through cropping and downscaling the input images [5], [6]. While this can greatly reduce the state space it comes at the cost of reducing the field of view and lowering the image quality limiting the potential information that can be extracted from image processing. Reducing the size of the visual architecture can ease training by reducing the number of trainable parameters but also reduces the potential information that can be extracted from the images. Even when using images smaller than 100x100 pixels and small network architectures, DRL algorithms can still require millions of iterations to solve a task sufficiently [4], [5].

This work proposes training a retina based vision system to extract state information from the images using supervised learning as an approach to reducing the difficulty of the learning problem for DRL. The agent maintains the benefits of having fewer network parameters in a reduced state space to train without restricting the field of view, image quality and vision architecture. A training environment is developed in Gazebo with the Baxter robot where the goal is to centre the arm over an object on a table. The software retina is fixated on the centre of the in-wrist image feed to perform the retino-cortical transform prior to processing by a DCNN to predict the object location. Deep Deterministic Policy Gradient (DDPG) is used to successfully train agents to predict end-effector position movements which are then carried out using Inverse Kinematics (IK). Agents are trained using different vision systems to evaluate the effect on training using the predicted object location compared to feature vectors of different sizes obtained by removing the prediction layers from the DCNNs. All agents learn quickly to solve the task and when demonstrated on a real Baxter robot the system generalised to centre over novel objects placed on the table.

II. Background

A. Software Retina

Balasuriya [7] proposed a compressed representation for images inspired by the mammalian vision system. The retina is a

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

layer of different nerve cells at the back of the eye which turns light into neural impulses before sending them to the brain. Photoreceptor cells sample the retinal image most densely in the fovea of the eye, located around the fixation point, and are increasingly sparsely sample towards the periphery. Ganglion cells are connected to the photoreceptor cells to combine their responses to provide data compression. In the fovea they have a very small receptive field, sometimes only connecting to one photoreceptor, whilst they are increasingly larger towards the periphery of the visual field. This creates a blurring effect whereby visual information is preserved at the fixation point and image information towards the periphery becomes progressively more blurred. Balasuirya used self-similar neural networks to develop artificial retina tessellations overlayed with Gaussian receptive fields to replicate this effect [7], [8]. Figure 1 shows a 256 node tessellation, this paper uses a high-resolution 50k node tessellation.

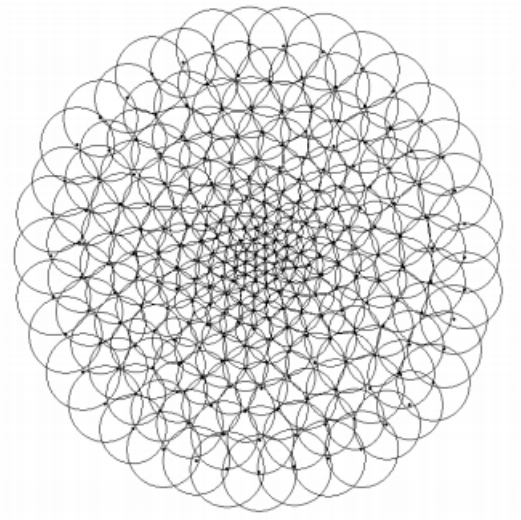

Fig. 1. 256 node tessellation with Gaussian receptive fields. Source: Balasuriya (2006)

Ozimek [8] extended Balasuriya's work by interpolating a *cortical image* from retinal samples and this image can be processed directly using conventional CNNs. The effect of the retina-cortex transform is to magnify the fovea within the cortical image whilst the periphery is progressively compressed with visual eccentricity from the optical axis. This sampling is manifest as a "zoom effect' enabling fine hand-eye coordination skills such as threading a needle, whilst also maintaining the full field of view [9]. Using cortical images for training CNNs provides shorter training times due to the increased image compression whilst also providing a degree of scale and rotation invariance because the cortical space is similar to the log-polar space [8], [9] where effect of input image scale and rotation is *quasi orthogonalised* in the cortical image axes. Figure 2 shows a starting image, a visualisation of the retinal blurring and the cortical image. Balog [10] further extended the software retina by implementing the retina-cortex transform on the Graphics Processing Unit (GPU) achieving a real-time performance of 15 frames per second when using an Nvidia GTX 1080 Ti.

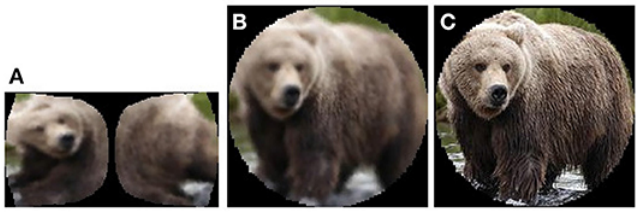

Fig. 2. The stages of the software retina from right to left. Image C shows the retina fixated on the centre, B is the backprojected retinal image and A is the cortical image. Source: Ozimek et al. (2019)

B. Deep Reinforcement Learning Hand-Eye Coordination

Using DRL to train general robotic grasping behaviours using image input has had many successes [1], [11]. Levine et al. [2], proposed training a DCNN on a large dataset of images and motor commands, consisting of 800k grasp attempts gathered over the course of 2 months, to predict their chance of grasp success then used the Cross-Entropy Method (CEM) to learn the optimal motor commands. Kalashnikov et al. [1], proposed the Qt-Opt algorithm which leveraged off-policy learning from real robot grasp attempts to train a large Q-network using 472x472 pixel images and CEM optimisation to find the action with the best Q-value. Their method achieves a 96% grasp success rate on unseen objects and performed sophisticated behaviours such as repositioning objects prior to grasping and regrasping. However, to gather the data took 800k robot hours and to train a sufficient Q-function took 5M-15M steps with further finetuning afterwards using a mix of off-policy and on-policy grasp attempts. The Q-network had 1.2M parameters which is considerable for a DRL problem but very small in comparison to architectures like ResNet trained solve challenging image dataset like ImageNet using supervised learning, which can have tens of millions of parameters [12]. Pore et al. [13] proposed improving the sample complexity of DRL by adopting a behavioural approach to grasping in the OpenAI FetchPickandPlace environment. Behaviour cloning was used to train three basic behaviours using expert demonstration: approach, grasp and retract, then DRL was used to train an LSTM to choreograph the behaviours deciding when they should run to successfully grasp a cube. To train the low-level behaviours and choreographer took under 9K episodes using kinematic and location data from the environment to reach 100% success rate compared to 95K for end-to-end learning using DDPG with Hindsight Experience Replay (HER). Our work is most closely related to the behaviour-based approach adopted by Pore [13] but with a retina vision system and a focus on training an approach like behaviour using DRL instead of behaviour cloning.

III. Methodology

A. Simulator Details

The Baxter robot was modelled inside Gazebo, a powerful 3D robotic simulator, with a blue cube on a table used as the target for the agent to centre on. Using a simulator made it possible to automate spawning the cube in random locations to create a more challenging learning environment and allowed safety checks to be turned off and joints to be moved at maximum speed without there being any risk of damage to the robot or anyone in the proximity of the robot. Accordingly, joint moves could be calculated and executed faster, both leading to shorter training times.

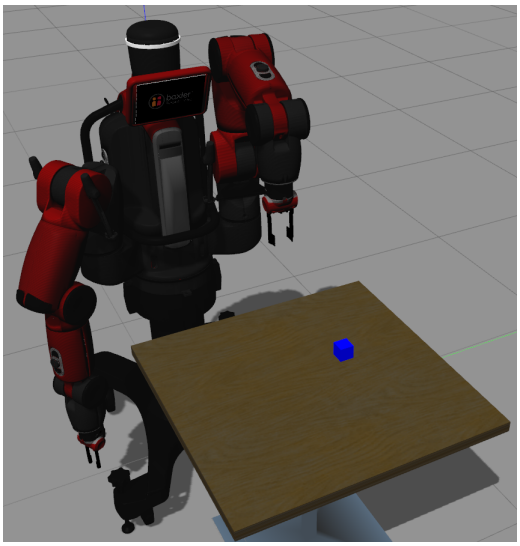

Fig. 3. Image of the simulated environment in Gazebo.

The goal in the environment is to move the Baxter robot's arm so that the object sits within the centre of the field of view of its in-wrist camera. Since the software retina will be used, this is somewhat equivalent to the mammalian eye fixating its fovea onto the object to extract its high-frequency visual information. Neurobiology suggests that when performing hand-eye coordination tasks, a human initially fixates on key locations before carrying out motor control [14]. To this end, the approach agent serves as a precursor to carrying out more complex manipulation tasks.

Our reward function comprises a negative value representing the distance of the object from the centre of the camera. The further away an object is from the visual field centre, the larger the negative reward. This encourages the agent to centre on the object as quickly as possible so as to accrue the least amount of negative reward. An episode reward of 0 would mean that the agent centred on the object perfectly in a single movement. The arm's endpoint is kept at a fixed rotation and depth as it would only become necessary to change these values were we attempting to manipulate the cube, which is not within the scope of this project. The inverse kinematics service supplied with Baxter was used so that the agent did not need to learn how to manipulate the joints directly. Instead, the agent had 2 actions which increased or decreased the x and y endpoint coordinates.

The maximum distance that the x and y end effector could move in a single action was limited 0.26 and 0.4 metres respectively as this is just large enough value to centre on any object in the field of view in a single action. The position of the end effector was clamped to be no more than 0.26 or 0.4 metres in the x or y distance from the object and thereby prevent the object being lost from sight during an episode. Originally, a larger negative reward was returned to discourage the agent from losing sight, but this was found to be very difficult to tune because too large a value seemed to destabilise the Q estimates, preventing learning. On the other hand, an insufficiently low reward made losing sight of the object too rewarding compared to exploring the environment and amassing negative rewards.

The environment and system state observations consist of the latest image taken by the in-wrist camera as well as the current x and y coordinates of wrist's endpoint. It was necessary to include the x and y coordinates so that the agent could learn the extent of its reachable area. Without this step, the agent would not be able to know whether an action could be executed or not and therefore could not predict reward values, which would preventing the critic from learning. The joint execution speed was set to the maximum possible values to maximise the rate at which actions could be executed. Unfortunately, this also introduced a noise component into the observations as the arm moved so quickly the wrist would exhibit under-damped residual oscillation after each movement was completed.

B. Vision System Details

The software retina is used to compress the full-size images from the in-wrist camera into smaller cortical images while maintaining high-frequency data at the centre of the camera. Two different DCNNs were implemented and investigated based on the successful ResNet architecture dubbed ResNet64, shown in table I, and ResNet128, shown in table II as they output 64 and 128 dimensional feature vectors respectively. Each architecture uses 5 convolutional layers with batch normalisation between each layer and global average pooling after the final layer. Batch normalisation is standard practice when training DCNNs and has been proven to improve performance and speed up training [15]. Two residual skip connections are used, one from before conv2_1 to after conv2_2 and one from before conv3_1 to after conv3_2 making conv2 and conv3 residual blocks. This provides many benefits including easier propagation of gradients to earlier layers, feature reuse from earlier layers and supports modelling of recurrent processes [12], [16]–[18].

To train the ResNet64 and ResNet128 architecture, the SGD optimisation algorithm with momentum was used. SGD with momentum was chosen over Adam because it tends to generalise better resulting in lower validation loss at the cost of a higher training loss and was used to successfully train

TABLE I
RESNET ARCHITECTURE WITH A 64 DIMENSION FEATURE VECTOR AND 155K PARAMETERS. REFERRED TO AS RESNET64.

Layer	*Kernel*	*Stride*	*Filters*	*Output*
Conv1	3x3	1	64	328x127x64
Conv2_1	3x3	1	64	328x127x64
Conv2_2	3x3	1	64	328x127x64
Conv3_1	3x3	2	64	164x64x64
Conv3_2	3x3	1	64	164x64x64
Avg Pool	-	-	-	64
Fc	-	-	-	2

TABLE II
RESNET ARCHITECTURE WITH A 128 DIMENSION FEATURE VECTOR AND 174K PARAMETERS. REFERRED TO AS RESNET128.

Layer	*Kernel*	*Stride*	*Filters*	*Output*
Conv1	3x3	1	18	328x127x18
Conv2_1	3x3	1	18	328x127x18
Conv2_2	3x3	1	18	328x127x18
Conv3_1	3x3	2	128	164x64x128
Conv3_2	3x3	1	128	164x64x128
Avg Pool	-	-	-	128
Fc	-	-	-	2

ResNet to achieve state of the art results on the ImageNet dataset [12], [16]–[18]. The learning rate starts at 0.01, then decays by a factor of 10 when the validation loss has improved for 10 epochs. A momentum of 0.9 was used and a weight decay of 0.00001 was included to help regularise the network following hyperparameters used for the ImageNet results [16]. Unfortunately, it was necessary to downscale the cortical images by 30% to allow for a reasonable batch size of 32 to be loaded into GPU memory. This is still quite a small batch size, compared to 512 used in the original ResNet paper [16], when using batch normalisation so it is likely that it introduced some noise into training.

Figure 4 shows the results of training the two different architectures on a 100k dataset gathered through random noise exploration. There were some unexplained fluctuations of the validation loss but they settled down after 30 epochs and both networks converged to very low loss values, with ResNet64 converging to a slightly lower loss.

C. Agent Details

The experimental details mostly follow those used by He et al. [4] due to its reported ability for effective learning in complex environments. Table III and table IV shows the network architectures used for the actor and critic respectively. The state was standardised before being processed by the network by maintaining a rolling estimate of the mean and standard deviation of each state dimension to assist learning [4]. The Adam optimisation algorithm was used with a learning rate of 0.0001 and 0.001 for the actor and critic respectively with an L2 decay of 0.01. A value of 0.001 was used for the soft network updates and the discount factor was set at 0.99 [4]. Additive action noise drawn from a normal distribution with a mean of 0 and a standard deviation of 0.2 was used to

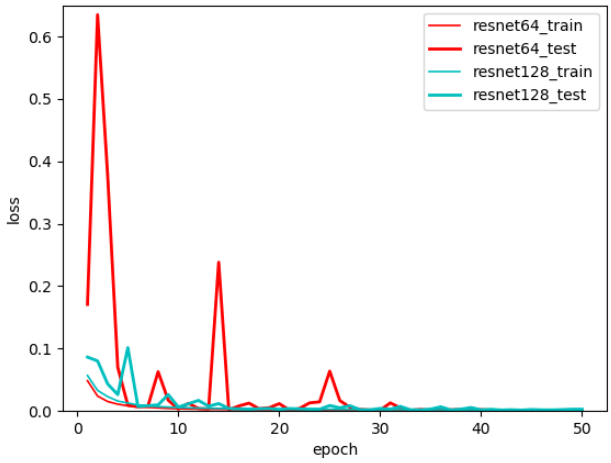

Fig. 4. Results of training the two different architectures from table I and table II using a dataset of cortical images from the Gazebo environment. 20k of the 100k labelled cortical images are reserved for testing.

encourage exploration. Over the first three-quarters of training, the noise was multiplied by a parameter starting at 1.0 to reducing to 0.02 to balance exploration with exploitation of the policy. Prior to training, 2k steps of random exploration using actions drawn from the noise function was carried to out to preprepopulate the experience replay [6]. To evaluate training progress, periodically 20 episodes were run without noise then the total episode rewards were averaged.

TABLE III
ACTOR ARCHITECTURE.

Layer	*Input*	*Activation*	*Output*
Fc1	State	Relu	400
Layer Normalisation	400	-	400
Fc2	400	Relu	300
Layer Normalisation	300	-	300
Actor Output	300	Tanh	Action

TABLE IV
CRITIC ARCHITECTURE.

Layer	*Input*	*Activation*	*Output*
Fc1	State	Relu	400
Layer Normalisation	400	-	400
Fc2	400 + Action	Relu	300
Layer Normalisation	300	-	300
Actor Output	300	Linear	1

IV. EVALUATION

A. Training on Environment Dynamics

To validate the simulated environment design and to provide a baseline performance measure, a DDPG agent was trained using the cube's centre location identified by colour thresholding. Figure 5 shows that the agent initially learns very quickly

to roughly centre on the object. After the initial spike, the agent slowly optimises its performance to centre on the object more accurately in fewer moves. After training for 100k timesteps, taking roughly 24 hours, the agent is able to centre accurately in one or two actions. If the cube is initially distant from the view-field centre, the agent takes a large action which roughly centres on the cube. A smaller action is then estimated and executed to refine the camera position with respect to the cube centre. It was observed that the agent continued to issue very small actions to try centre with increasing accuracy. However, these residual positional refinements could not be executed as the joints have an accuracy tolerance which prevents any update to their angles below this tolerance limit.

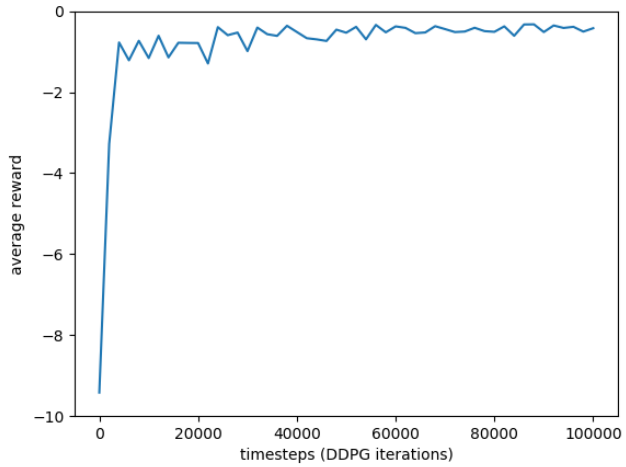

Fig. 5. Training performance using low state descriptor. Each timestep corresponds to carrying out one gripper move and subsequent network update.

B. Running Agent using Retina ResNet instead of Dynamics

In this project, the two ResNets that have been trained predict the dynamics of the environment with very high accuracy. This is less likely to be true in more realistic environments because they are much more complex containing many objects which may have never been seen in training as well as various sources of noise. Rather than train the DDPG agent on top of an imperfect vision system this method instead proposes that the agent is trained with perfect information, i.e., how the low state agent was trained. This allows the agent to understand in optimum circumstances how the information should be utilised to perform the task without incorrect information corrupting the agents understanding of the dynamics of the problem. This provides a clear separation of the two principal learnt tasks: how to use the visual information to perform the task and how to extract accurate estimates of the visual information. If a new vision system is developed which is more accurate then it can be integrated with the agent providing a performance improvement without the need to retrain the agent completely. If the agent was trained using imperfect information then it is not guaranteed that more accurate estimates would be of benefit, because the agent may have internalised a degree of noise into the policy.

To evaluate how effective the agent is when using the ResNet to predict the centre location instead of colour thresholding, the agent's performance was evaluated over 100 episodes of 5 timesteps in length. Table V shows that using the ResNet with the software retina incurs a slight performance hit but from inspection, this is due to the agent sometimes requiring an extra move instead of it being less accurate. It is suspected this is due to some of the predictions by the ResNet being slightly inaccurate in some specific circumstances. Figure 7 shows the predicted location using the cortical image in figure 6 is slightly too close to the centre of the camera because part of the cube is not visible in the cortical image, seen in figure 7. To compensate for the inaccuracies of the initial prediction, after the cube has moved closer to the retina's fovea generating a more accurate prediction the agent takes additional actions to better centre on the cube. This results in a slightly lower episode reward but that is inconsequential because the agent still achieves the goal of accurately centre on the object by utilising the software retina.

TABLE V
MEAN AND STANDARD DEVIATION OF THE EPISODE REWARDS OBTAINED BY AGENT USING THE ENVIRONMENT DYNAMICS COMPARED TO USING THE DYNAMICS AS PREDICTED BY THE VISION SYSTEM.

State Vector	*Performance (Avg. Reward)*
Dynamics	-0.163 $\pm$ 0.062
ResNet64 with Retina	-0.200 $\pm$ 0.074

Fig. 6. Cortical image generated by software retina using image from the in-wrist camera.

C. Training Agent using Retina Features

Having successfully trained the two ResNet architectures using the software retina, this section explores how well the DDPG agents can learn using the image features. Two different agents were trained by using the 64 and 128 dimensional features vectors produced by the ResNet's when their final fully connected layer was removed. The experimental details are the same as the previous DDPG experiments except it was decided that only the x and y endpoint location of the arm should be standardised using a rolling mean and standard deviation. This is due to the fact that the image

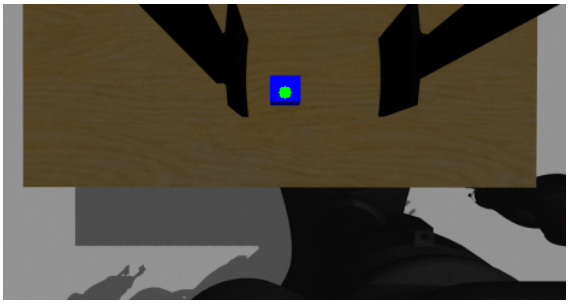

Fig. 7. Location predicted using cortical image in figure 6. Slightly underestimates vertical distance from the centre of the fovea.

features were batch normalised before being processed by the global averaging layer so the activations have already been normalised. To ensure that the training results are fair, all seeds for the random generators were set to the same value so that exploration noise, cube spawning locations and weight initialisation were the same for each agent.

This is a much more challenging environment for the agent to learn because its state dimension is much larger, increasing from 4 to 66 and 130 respectively. Instead of being given the exact central location of the object, the agent is given visual features which it must use to develop its own understanding of where the object is located and how that is affected by different actions. Figure 8 shows the relative training performance of the DDPG agents using the ResNet64 state, ResNet128 state and low state. Although the low state agent begins consistently outperforming the other agents after 50k timesteps, the difference is very small and both the ResNet64 and ResNet128 agents also learn successful policies. It was suspected that using a much larger state dimension would take longer for the agent to learn but, in practice, these results show little difference between the ResNet64 and ResNet128 agents.

It would have been interesting to have run the training for longer as it appears that all three agents are still learning by 100k timesteps but unfortunately there was not enough time available. However, the goal of this experiment was not to compare the final performance of the agents but to investigate a different method of integrating DCNNs and the software retina with DRL without greatly increasing sample complexity. To that end, these results show that agents can learn using the larger feature vectors extracted by a retina vision system trained on a task related dataset without greatly impacting training performance. To the best of the authors' knowledge, this is the first example of a DRL agent trained using image features extracted from images in a cortical space. Another important factor to note is that despite the use of a DCNN, large images and the software retina the training time was only increased by a couple hours compared to training the agent using the environment dynamics. Environment images are processed by the vision system whereupon only the state is stored in the agent's experience replay, greatly increasing the speed of off-policy sampling compared to end-to-end learning and compensating for the increased processing time of vision

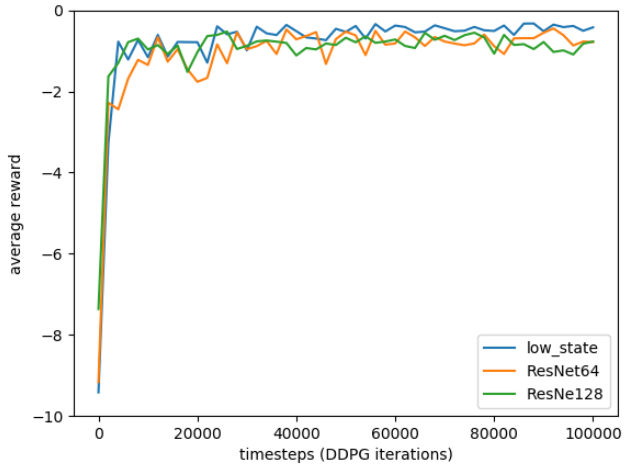

Fig. 8. Average episode reward during training using ResNet architectures in table I and table II without fully connected layer. Results training using low state used as baseline. Each timestep corresponds to carrying out one joint move and subsequent network update.

system.

D. Training using Dynamics or Image Features

Choosing whether to train the agent using the dynamics as predicted by the retina vision system or the image features is a question of trade-offs. When using the dynamics, the system is more accountable when a mistake is made in comparison to using image features. It can be clearly seen whether the issue was due to the vision system making a mistake resulting in incorrect information being used by the agent or whether the information was correct but the agent misunderstood how to use it. This could be very important in applications such as self-driving cars where there is a need for accountability because the costs of a mistake could be dire, i.e., a car crash. In addition, it may be undesirable to retrain the entire driving agent every time a better vision system is developed which would be required when using image features since a new vision system would be equipped with new features.

However, using dynamics is restrictive and may not always be applicable to the problem at hand. Perhaps there is information not included in the dynamics which is beneficial or vital to the agent's performance. By training using the image features the agent may be able to extract this information itself or learn an entirely different understanding of the environment which is better than the dynamics. In certain tasks, the dynamics may not be known, or it may be too difficult to gather a dataset for training a DCNN to predict it. In this case, there may be no other option than giving the agent access to the visual features and allowing it to figure it out itself. If some information is known to be helpful in solving part of the problem, a mix of dynamics and image features could be used to give the agent some prior higher-level knowledge to assist learning whilst the image features can be used to learn new information.

V. Demonstration on the real Robot

To transfer the vision system from simulation to the real world, attempts were made to freeze initial convolution layers to fine-tune the final connected layers. Here we discovered that because of the extreme difference in pictures, passing only a small amount of real-world data in the previously trained network, resulted in severe over-fitting. To address this, we increased the number of real-world images collected from 3200 to 30000, and employed only a slice of the images used to train in simulation. There is a visible similarity between the real-world training and training via the simulator, with fluctuations believed to be caused by the reality gap of the two worlds. However, with the added real-world dataset, the transfer succeeded in minimal time.

For the actuator part, it was perceived that due to the exponential growth in environment complexity between simulation and reality, the training time increased dramatically. Having such a long training time was deemed to be infeasible due to time limitations of the project. To combat this, the weights learned in the simulation were used as a starting point for the experimentation case. It was found out that these weights, although they provided a boost during training - by focusing in close proximity of the object - they were not optimal in the real world scenario, because of the differences in the two environments. Therefore, training was re-run to fine-tune the weights using one object location and subsequently, trained again using multiple cube locations. This has split the training into three phases, with each phase taking 13-24 hours to run. The resulting performance was slightly lower than the results in the simulation. The average reward during the final evaluation was -1.1 $\pm$ 1.34. In training, the rewards gained during the first 20000 time steps are fluctuating, but that can be justified as accustomization to the new environment. After 20000 time steps, the agent acting on the real-world shows similar results to the simulation, albeit with a moderately smaller reward gain.

Despite training the agent using a cube as an object, while allowing the vision system to use the full RGB colour space, the agent has shown interesting behaviours, such as generalising to follow any object. The agent experiences some confusion when presented with two objects, centring roughly in the middle of the two objects. The agent also exhibits object tracking capabilities, but is bottle-necked ed by the visual component, as it takes around one second for the image to be processed by the CNN.

VI. Future Work

A. Transfer Learning

The datasets used in this project were simple and geared towards the specific environments making the image features learned by the DCNNs more specialised. A more challenging problem would be training using large and more diverse datasets designed for different problems, e.g., motion and depth perception, object localisation etc. This would result in a more general and robust set of features giving the agent a more informative observation of its environment. This may allow the agent to learn how to solve many different problems using the same vision system removing the need for gathering a specialised dataset. This would be particularly beneficial for problems when the dynamics needed to solve the problem are unknown or too hard to train a DCNN to predict.

An issue of using transfer learning is that the powerful DCNNs that obtain state of the art results on large datasets often have very large feature vectors ranging from 512 to 2014 dimensions and higher [12], [16], [17]. This creates a much larger state space which significantly increases the complexity of the learning task. Although in this project using a feature vector of size 128 instead of 64 did not appear to have a large effect on training, this may be due to the feature vector being specialised towards the environment and a larger state space with more task-irrelevant information may degrade sample complexity and training time. State representation learning may be a promising approach for compressing the large state space to a smaller and more task-relevant representation before training the DRL agent [19].

B. Add Residual Block to Agent

In addition to removing the final fully connected layer, the global averaging layer could also be removed giving the agent access to the feature maps. Whilst the pooling layer is useful for culling down the state dimension that the agent needs to learn it may also lose important visual information that would be beneficial to training and the agent's final performance. Adding a residual block to the start of the agent would allow the agent to utilise the full visual information whilst also allowing it to learn higher-level features specific to its current environment. The use of a residual skip connection could allow the agent to decide to what extent it employs the use of the convolutional layers instead of reusing the features [18]. This would greatly increase the state dimension but since DCNNs typically extract more abstract information in later layers and have smaller feature maps, the training task may be easier than using raw pixel values.

VII. Conclusions

This paper investigated the difficulties of training the vision system and robotic actuation using end-to-end DRL. It was found that while agents have been successfully trained using larger networks and images, the poor sample complexity of DRL and hardware requirements of doing so limits its applicability [1]. Instead of using image downscaling and smaller DCNNs to ease training difficulty [4], [5], supervised learning was used to train a retina vision system to extract state information from the environment before training using DRL and was found to greatly improve the sample complexity and off-policy sampling speed of DDPG. A hand-eye coordination system was developed on a real Baxter robot which could track any object placed on the table and within the field of view of the in-wrist camera despite only training using cubes highlighting the potential for generalising robotic behaviours by developing general computer vision systems. To the best

of this authors knowledge, this is the first example of training a DRL agent on top of a retinal vision system.

REFERENCES

[1] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *CoRR*, vol. abs/1806.10293, 2018. [Online]. Available: http://arxiv.org/abs/1806.10293

[2] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *CoRR*, vol. abs/1603.02199, 2016. [Online]. Available: http://arxiv.org/abs/1603.02199

[3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning." *CoRR*, vol. abs/1312.5602, 2013.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[7] S. L. Balasuriya, "A computational model of space-variant vision based on a self-organised artificial retina tessellation," Ph.D. dissertation, University of Glasgow, 2006.

[8] P. Ozimek, "Integrating a biologically inspired software retina with convolutional neural networks," Master's thesis, University of Glasgow, 2017.

[9] P. Ozimek, N. Hristozova, L. Balog, and J. P. Siebert, "A space-variant visual pathway model for data efficient deep learning," *Frontiers in Cellular Neuroscience*, vol. 13, p. 36, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncel.2019.00036

[10] L. Balog, "A gpu accelerated software retina," Master's thesis, University of Glasgow, 2017.

[11] I. Popov, N. Heess, T. P. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. A. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *CoRR*, vol. abs/1704.03073, 2017. [Online]. Available: http://arxiv.org/abs/1704.03073

[12] S. Zagoruyko and N. Komodakis, "Wide residual networks," *CoRR*, vol. abs/1605.07146, 2016. [Online]. Available: http://arxiv.org/abs/1605.07146

[13] A. Pore and G. Aragon-Camarasa, "On simple reactive neural networks for behaviour-based reinforcement learning," in *In: International Conference on Robotics and Automation (ICRA 2020)*, 2020.

[14] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, "Eye–hand coordination in object manipulation," *Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, 2001. [Online]. Available: http://www.jneurosci.org/content/21/17/6917

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[17] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *CoRR*, vol. abs/1608.02908, 2016. [Online]. Available: http://arxiv.org/abs/1608.02908

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016. [Online]. Available: http://arxiv.org/abs/1603.05027

[19] T. Lesort, N. D. Rodríguez, J. Goudou, and D. Filliat, "State representation learning for control: An overview," *CoRR*, vol. abs/1802.04181, 2018. [Online]. Available: http://arxiv.org/abs/1802.04181