Behaviourally Cloning River Raid Agents

Laurens Diels
imec-TELIN-URC-IPI, Ghent University
St-Pietersnieuwstraat 41, 9000 Ghent, Belgium
laurens.diels@ugent.be

Hussain Kazmi
ESAT-ELECTA, KU Leuven
Kasteelpark Arenberg 10, 3000 Leuven, Belgium
hussainsyedkazmi@kuleuven.be

Abstract

We investigate the feasibility and difficulties of using behavioural cloning to obtain player models using the 1982 video game River Raid. We attempt to clone both virtual game-playing agents (a fixed (non-improving) reinforcement learning agent and a random agent sampling actions uniformly) as well as an actual human agent. The behavioural clones' performance is evaluated on the micro-level through comparison of the state-conditioned and unconditional action distributions, and on the macro-level by comparing the (cloned) agents' survival time and score per episode. Using our methodology, cloning virtual agents seems feasible to varying extents, even with somewhat limited amounts of data. However, our method fails to create reliable behavioural clones of human players. We conclude with a discussion of some of the more important reasons that might cause this: a lack of training data, the problem of covariate shift, and improving and inconsistent play-style over time.

Introduction

There has been considerable amounts of research efforts concerning reinforcement learning in video games, especially since DeepMind's breakthrough paper *Playing atari with deep reinforcement learning* (Mnih et al. 2013). Here they used their technique of deep Q-networks to play games for the Atari 2600, such as River Raid.

Traditionally in reinforcement learning the goal of an agent is to choose actions that maximise its long-term rewards. In many applications such as energy management this makes perfect sense. However, in video games, which are mostly just meant to be fun, playing at a superhuman level might not be optimal. Indeed, for most humans, playing against such foes will be very frustrating. In this setting it might then be more interesting to attempt to be able to generate human-like opponents (or allies), or even to capture specific players. The use cases can be diverse (Pfau, Smeddinck, and Malaka 2018). One could for example analyse the player models to create personalised challenges, or, when detecting a mismatch between player and model, to detect cheating or identity fraud. Or in multiplayer games human allies and opponents can be replaced by similarly playing bots. Research has shown that player engagement is higher when playing against more human-like bots than those programmed using fuzzy finite state machines (Soni and Hingston 2008).

There are multiple ways to capture player models, such as inverse reinforcement learning (IRL) and behavioural cloning (BC). To capture the use cases highlighted above, we will focus on the latter approach. The idea is as follows. A player model should be able to predict which action the player will choose in a given state of the game. Behavioural cloning just views this as a multiclass classification problem which can be solved by collecting many examples of the player's input-output pairs.

More concretely, we will use behavioural cloning to attempt to capture player models in the game of River Raid. This is a 1982 vertical autoscroller for the Atari 2600, where the goal is to survive for as long as possible and obtain a high score. The game uses frames of size 160×210 with 3 colour channels, at 60 frames per second. See Figure 1 for an example frame together with some explanation of what is happening in the game. Although we will emulate the game in the Arcade Learning Environment (ALE) (Bellemare et al. 2013) integrated into OpenAI Gym (Brockman et al. 2016), like DeepMind we will only consider the game frames as inputs, and not for example the (emulated) console's internal RAM of 100 bytes. As outputs we will use the 18 possible combinations of 9 movement directions (cardinal, intercardinal and no movement) and a binary choice of shooting or not.

Methodology

Game-playing agents While our ultimate goal is to model human players, there are a number of challenges associated with behavioural cloning of such players. First and foremost, humans tend to play quite inconsistently and improve over time. Moreover we are likely to have only limited amounts of training data for a single human player. Therefore we opted to start with computer agents as base agents to clone. This allows for the collection of as much data as needed, as well as an estimation of how BC improves with increasing amounts of (stationary) data.

We first consider a reinforcement learning agent in a fixed state. That is, we made sure it was no longer improving after the training phase, which consisted of playing for 1000 episodes using Proximal Policy Optimisation (PPO) (Schulman et al. 2017), as implemented in Keras Gym (Holsheimer

Figure 1: A greyscaled frame from River Raid (emulated). The player controls the aircraft at the bottom of the screen. Here it is moving to the left after having fired a shot which is about to hit the second enemy ship. The aircraft always moves upwards, but the player can control its speed to some extent. The current score is 2690 which will increase after destroying the ship. Further there is a fuel meter which can be refilled by the fuel pad at the bottom left. The player has two remaining lives (not including the current one). The black borders are part of the image and will not be cut off.

2019). The reinforcement learning agent will play quite predictably. Completely opposite to this we will also consider a random agent which simply samples actions uniformly, irrespective of the current game state. Finally, we will also consider data from a human agent, namely the first named author.

Training The data collection consists of letting the various agents play the game of River Raid for a number of episodes. Next, we train a model that can predict the actions the base agent would choose given the current game state. In fact, as input for our model, we will use stacked game frames (in chronological order), to provide a local history. Since consecutive frame stacks will be extremely similar, we only use every third frame stack for training. For the model architecture, we chose a neural network consisting of two convolutional layers of 128 filters of size 3×3, a max pooling and flatten layer, and finally three dense layers of 128, 128 and 18 nodes. We also included Monte Carlo dropout ($p = 0.5$) layers (Gal and Ghahramani 2016) in front of the flatten and dense layers. These serve the dual purpose of preventing the model from overfitting, and introducing stochasticity in the (trained) model, i.e. the same input stack can lead to different outputs during different forward passes. As an additional way to prevent overfitting we also use L^2-regularisation (weight decay).

We used Nadam (Dozat 2016) as the optimiser, and the hyperopt package (Bergstra, Yamins, and Cox 2013) for optimising over the hyperparameters, namely the learning rate, L^2-regularisation strength, and number of frames to stack using the tree-structured Parzen estimator approach (TPE) (Bergstra et al. 2011). We used a loguniform distribution between 10^{-5} and 10^{-3} for the learning rate, a loguniform distribution between 10^{-4} and 10^{-2} for the regularisation constant and a uniform distribution on $\{2, 3, \dots, 25\}$ for the number of frames to stack. We let the tuning algorithm run for 50 combinations of hyperparameters (or 44 in the human case due to practical considerations of computation time), evaluated using the average validation accuracy over the last training epoch of our model. For each combination of hyperparameters we trained the model for 10 epochs with 50 training episodes and 25 validation episodes from the base agent. These optimised hyperparameters were then used to train the final models used for BC. We opted to always use 25 epochs.

Evaluation Given a behavioural clone of a game-playing agent, we will evaluate it in two different ways: (1) by comparing action distributions, and (2) by comparing play-style data. Part of the distribution comparison will only work for computer agents (or, more generally, agents that we can easily sample at will).

We will start with comparing local action distributions, which is only possible when the game-playing base agent is a computer agent. Since both this agent and its clone are stochastic, we can sample probability distributions of actions in a certain frame stack. This way we obtain empirical distributions. The Kullback-Leibler (KL) divergence is a popular choice to compare distributions. However, this is ill-defined when one (the second) distribution vanishes while the other (the first) does not. Therefore we opted to simply use the L^1-distance to compare these empirical conditional action distributions. The frame stacks we condition on were obtained by considering every tenth frame stack from an episode from the human player. This yielded a total of 115 frame stacks. For each of these and for each base agent and clone we sampled 100 actions. This results in 115 empirical conditional action distributions per agent and consequently 115 L^1-distances for each base agent - clone pair. To aggregate these into one number we simply took the mean. Thus in this manner we obtain what we call the *mean local (or conditional) L^1-distance* between a computer agent and its clone.

We can also approximate the global unconditional action distribution of an agent by taking all the 11500 sampled actions. We can then compare the global distributions of a game-playing base agent and its clone by taking the L^1-distance. This results in the *global L^1-distance* between the agent and its clone.

For human agents, however, we can not rely on local distributions, as these agents cannot (reliably) be sampled in a given state. There is also an additional issue that the previous performance metrics can at times be considered quite artificial, as the states we condition on might never be encountered by an actual agent. For example, if an agent never makes it very far into the game, the relevance of its hypothetical performance at this stage in the game is disputable. Therefore we will also compare a base agent and its clone solely by looking at played episodes.

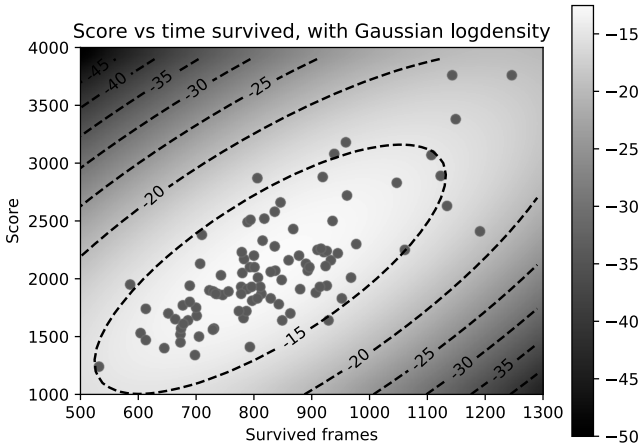

Figure 2: The time-score scatter plot from the reinforcement learning agent with overlaid Gaussian density and contour lines. The colour intensity of the background, quantified in the colour bar at the right, indicates the logdensity of the Gaussian distribution fitted using maximum likelihood estimation. Contour lines of this logdensity are also shown.

After completing an episode we can extract the number of frames the agent survived, and the score the agent obtained in this time. In this way we can represent every episode as a point in 2D (survival time, score) space. Playing for many episodes yields a point cloud (scatter plot), and we can compare such a point cloud of a base agent with that of a clone.

The problem now shifts to comparing these point clouds. We will consider two methods. In the first one we fit a 2D Gaussian distribution onto a point cloud using maximum likelihood parameters. See Figure 2 for an example. Then we compare the fitted Gaussians of the base agent with the fitted Gaussian of a clone using the KL divergence. We will refer to the resulting scalar metric as the Fitted Gaussian KL Divergence or *FGKLD*. This approach of course assumes that the point clouds are approximately normally distributed. Based on Figure 2 this assumption seems reasonable, at least in this case.

The second method is non-parametric and is based on the Kolmogorov-Smirnov two-sample test, lifted to higher dimensions (Fasano and Franceschini 1987). Given two sets of points this test returns a p-value for the null hypothesis that both sets were sampled from the same underlying distribution. Therefore, similar point clouds will receive a high p-value near 1, and dissimilar point clouds a value near 0. These p-values can then be used to compare the point clouds of a base agent with that of its clone. In addition we get a normalised quantity, as opposed to the KL divergence. We refer to this value as the Kolmogorov-Smirnov p-value or simply the *KSp*.

Results and discussion

As explained in the methodology section we will start with the reinforcement learning agent, followed by a random agent sampling actions uniformly. Finally the case of behaviourally cloning the human agent will be examined.

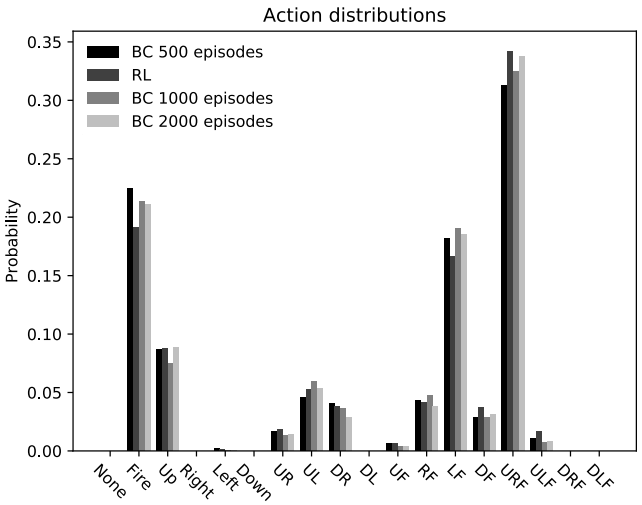

Figure 3: Histograms of the global action distributions of the reinforcement learning agent (RL) and some of its behavioural clones (BCs) trained using varying numbers of episodes. For the labels of combined actions we abbreviated Up by U, Right by R, Down by D, Left by L, and Fire by F.

Reinforcement learning agent For the purposes of cloning the reinforcement learning agent we let it play for 2571 episodes, which constitutes 2174108 frames and corresponds to about 10 hours of non-stop playing. To investigate the effect of the amount of training data on the clone's performance, we trained models using 1, 2, 5, 10, 25, 50, 125, 250, 500, 1000 and 2000 training episodes. (The number of validation episodes used was always a quarter of the number of training episodes, rounded up.) This resulted in 11 different behavioural clones for the reinforcement learning agent.

The performance of these clones is evaluated in Table 1, according to the metrics we previously discussed. For the creation of the time-score scatter plots we used 100 newly sampled episodes. For the metrics based on the action distributions we can see a general, albeit not monotonic, improving trend as the number of training episodes increases. As can also be seen in Figure 3, where we present histograms of the global action distributions, we also certainly end up with low (i.e. good) values for the L^1-distance. Based on the comparison of the distributions, it seems that using more than 250 episodes yields little improvement.

On the other hand, the measures based on the time-score scatter plots yield chaotic results: there is no clear trend to be found when increasing the amount of training data. Since the FGKLD, like all KL divergences, is unnormalised, it is not clear whether the results are generally good or generally poor. The KSp is more interpretable in comparison. Although quite low in general, with the exception of the case of 2000 episodes, from 50 episodes onwards it lies above 5%. This means we will not reject the null hypothesis of the point clouds being sampled from the same distribution at a confidence level of 95%.

Note the good results of the clone using 10 episodes for the FGKLD and KSp. Visual inspection shows that this

Training episodes	1	2	5	10	25	50	125	250	500	1000	2000
Global L^1-distance (↓)	0.2774	0.1341	0.1897	0.2287	0.2522	0.1595	0.1449	0.1297	0.1064	0.1167	0.07930
Mean local L^1-distance (↓)	1.099	0.8290	0.7313	0.7183	0.6496	0.6003	0.6365	0.4518	0.4979	0.4266	0.4534
FGKLD (↓)	1.218	4.324	0.3273	0.08024	1.882	0.1838	0.1056	0.2620	0.2131	0.4118	0.5033
KSp (↑)	$2.002 \cdot 10^{-6}$	0	0.02921	0.2670	0	0.1509	0.09667	0.07243	0.05845	0.1134	0.01617

Table 1: Evaluation of the behavioural clones of the reinforcement learning agent. Values of 0 for the KSp are due to limited numerical precision. Arrows next to the performance measures indicate whether they are subject to maximisation (↑) or minimisation (↓). For a point of comparison for the results in terms of the action distributions, an agent sampling actions uniformly has a global L^1-distance of 1.131 and an average local L^1-distance of 1.794 to the base reinforcement learning agent.

Figure 4: Histograms of the global action distributions of the random agent and its behavioural clone (BC).

clone does play quite similarly to the reinforcement learning agent. However, we found the same for the clone of 2000 episodes, despite the low KSp. It seems that from a low FGKLD and high KSp we can conclude that a clone performs well, but the converse does not hold: if a clone has a high FGKLD and/or a low KSp it might still play visually similarly to the base agent. These metrics therefore seem to have only limited discerning power about the quality of the behavioural clone.

Random agent We collected 511 episodes (521721 frames) from the random agent in action. The comparison of actions distributions between this agent and its behavioural clone trained on 400 of these episodes yielded a global L^1-distance of 0.3023 and a mean local L^1-distance of 1.050. Note that these values are somewhat comparable with the clone of the reinforcement learning agent using only 1 training episode, i.e. they are considerably worse than the clones using a decent amount of episodes in that case. Thus the action distributions are captured rather poorly. See also Figure 4.

For the scatter plot based metrics (see Figure 5) we again used 100 newly sampled episodes. Although the KSp is quite low at 0.0621, so is the FGKLD at 0.1969. So for the latter metric at least we score quite well, although it is the only one coming to this conclusion. When visually inspecting both the clone and the base agent, subtle differences are noticeable, but overall they feel quite similar in the sense that they both play very erratically. Although there are clearly objective differences between the random agent and its clone, these might be more or less problematic depending on the use case of the behavioural clone.

Human agent Finally, for the human agent, we collected 254 episodes, which corresponds to 531102 frames or about 2.5 hours of game-play. Note that such episodes are, on av-

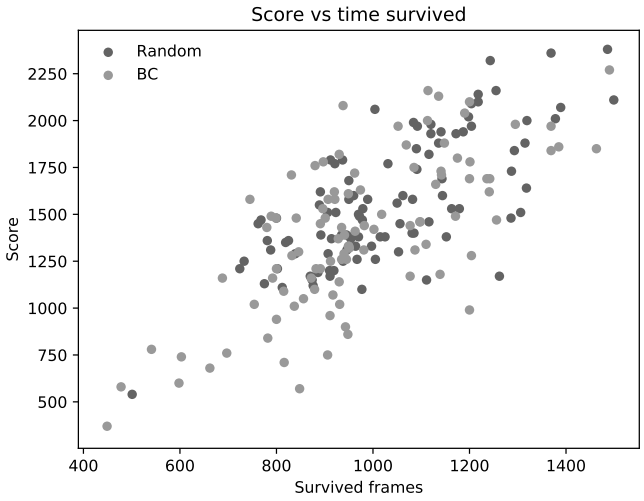

Figure 5: Time-score scatter plots of the random agent and its behavioural clone (BC).

erage, more than twice as long as those of the reinforcement learning agent. We used 203 episodes for training a behavioural clone. The other episodes were used for validation (25) and testing (26). The validation results showed that our approach of always training for 25 epochs led to slight overfitting in this case (and only this case), despite the dropout layers and L^2-regularisation. The validation accuracy at the end was about 3% lower than at its highest point. Therefore, in the future we would recommend early stopping. Additionally since for all the training processes in this paper the loss learning curves were reasonably convex, this would have reduced the computation time considerably, at no accuracy cost.

Recall that for evaluation of the clone we cannot use local action distributions, as the human agent cannot be sampled many times in a fixed state. But we can still compare the unconditional action distributions, now obtained by simply looking at the performed actions in the test set. For the clone we used 26 new episodes. The L^1-distance between the obtained unconditional action distributions was 0.5211 which is the highest value encountered so far, although we should point out that the way we obtained this global distribution differs from what we did for the previous cases (as explained in the methodology section). The poor result is confirmed by the histogram in Figure 6.

In terms of the time-score scatter plots (Figure 7) the results are worse by a considerable margin as well: the FGKLD was 24.97, the KSp turned out to be 0 (due to bounded numerical accuracy).

We will now attempt to explain the poor performance. Firstly, data is more limited. However, it is not clear whether having more training data would actually help. We attempted to use the reinforcement learning behavioural clone as an initialisation for the human agent clone, but this only made the situation worse. Secondly, there is an inherent problem with (naive) behavioural cloning: covariate shift. Since the clone is imperfect, it will sometimes move into states the

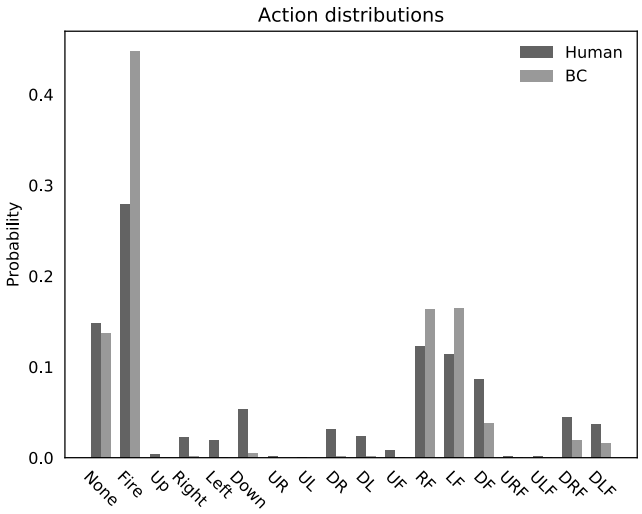

Figure 6: Histograms of the global action distributions of the human player and its behavioural clone (BC).

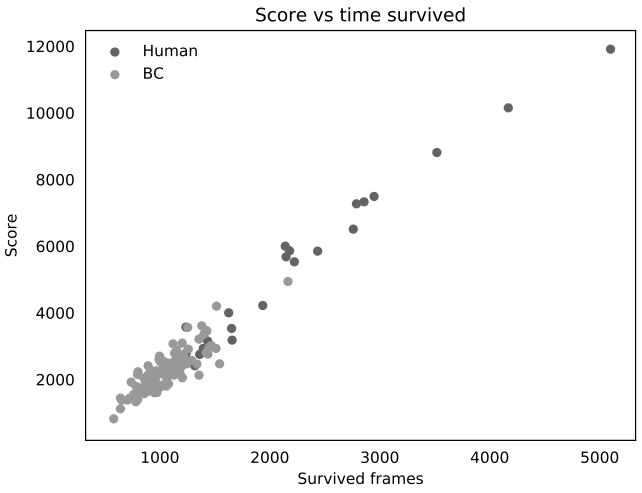

Figure 7: Time-score scatter plots of the human agent and its behavioural clone (BC).

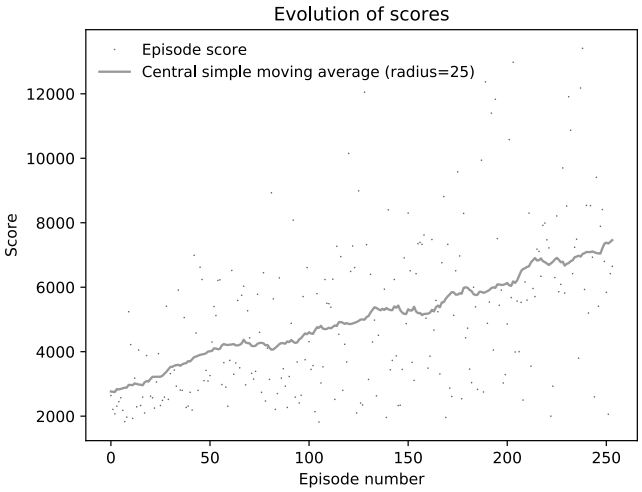

Figure 8: Improvement of the scores of the human player over time. The line is the central simple moving average of the sequence of scores (dots), for a smoothing radius of 25, i.e. if the n-th episode score is s_n for $1 \leq n \leq N$, where $N = 254$ is the total number of episodes, then the smoothed point on the curve is $t_n = \text{avg}\{s_k \mid 1 \leq k \leq N, |k-n| \leq 25\}$.

base agent would not. Lacking training data it might then continue to make unwise decisions. Thirdly, there is the issue of stationarity: humans improve over time while playing a game. To illustrate this, consider Figure 8, which shows the improvement of the human agent. During data collection the average score improved by a factor of more than two, which we cannot simply ignore as we have done in this paper. Finally, we claim that the human agent actually plays the most inconsistent out of all base agents, at least on the macro-level. Indeed, globally speaking the random agent simply moves forward, as on average its movements to the sides cancel out.

To quantify the last assertion of erratic game-play, we use a simple strategy. Consider one specific game-playing agent. Now let it play the game for many episodes. From each such episode, we collect the 60th frame. This frame number was chosen to be near the beginning of an episode, but still far enough into the game so that the agent's chosen actions can make a difference. We will remove the borders from these frames so as to only end up with the playable space, and not, for example, the score. This yields a set X of such cropped frames. We then calculate the medoid $m \in X$, i.e. some representative frame. For now we postpone the procedure for actually obtaining this medoid; at this time simply assume we have found it. Then the average deviation from this frame would give an indication of how unpredictably the agent plays. To calculate the deviation from a given frame $x \in X$ to the medoid m we will simply count the number of pixel locations where x and m differ. Thus, given a medoid we can now quantify the inconsistency of an agent in this manner. It then remains to find a medoid m. For this we will simply try all $m \in X$ and take the one leading to the smallest mean deviation. This relation between medoid and mean deviation is similar to the relation between the mean and (biased) variance of a numerical data set.

For the reinforcement learning agent and random agent we used 100 new episodes to collect X, for the human agent we used the 26 test episodes. The inconsistency scores were 519.5 for the reinforcement learning agent, 595.4 for the random agent, and 757.4 for the human agent. Using 50th, 55th, 65th and 70th frames always gave the same ordering, leading to the conclusion that the human agent plays the most inconsistent, followed by the random agent and finally the reinforcement learning agent. This then also shows that behaviourally cloning human agents is more difficult than the other agent types, because humans are the least consistent.

Conclusions

In this paper we attempted to capture players models in River Raid using behavioural cloning. We developed measures of comparing a base agent and its clone, namely the L^1-distances between the local and global action distributions, and the fitted Gaussian KL divergence FGKLD and Kolmogorov-Smirnov p-value KSp for comparing time-score scatter plots of the agents in action. Cloning our reinforcement learning agent gave hopeful results, both in terms of the captured distribution and the time-score scatter plots. The random agent proved much harder already, certainly at the level of distributions. On the level of play-style (time-score) we got mixed results with the FGKLD being quite good and the KSp quite poor. On a qualitative level it was still possible to distinguish the random agent from its clone, although the differences were not very large: both agents simply appeared erratic. This might already be sufficiently similar in certain use cases. Finally we got poor results for all metrics when cloning the human agent. The main reasons appeared to be the fact that humans improve over time and play inconsistently. Covariate shift is also a theoretical issue with our naive approach. The limited amounts of training data might be an issue as well, although this warrants further investigation.

It turned out our developed measures can only be used to conclude a clone performs well, but the converse does not hold: clones with poor performance metrics might qualitatively still play very similarly to be base agent, at least to our eyes.

Further research

We should investigate whether using more training data would help in the case of human agent cloning. Indeed, our guideline of 250 training episodes might only be valid for the easy case of reinforcement learning agents, or even only our specific reinforcement learning agent. However, we feel that increasing the amount of training data alone would not suffice in the case of the human agent.

As previously mentioned, covariate shift is an inherent problem with simple behavioural cloning. Luckily there are a number of strategies to deal with covariate shift. On-policy techniques like DAgger (Ross, Gordon, and Bagnell 2011)

require us to continually consult the supervisor (base agent) during training. With the ultimate goal of getting human player models, these techniques are consequently labour-intensive. On the other hand DART (Laskey et al. 2017) is an off-policy technique which during data collection injects noise into the supervisor's chosen actions. This forces the supervisor to also deal with the consequences of unwise decisions, even when the supervisor would never make them. By enforcing deviation from an optimal strategy it ensures more states are covered in the training data (which stores the real chosen actions). This way when a clone 'gets lost' due to suboptimal cloning performance, it has reference data to get 'back on track'. It is worth investigating whether DART would then be able to improve our cloning results. But we should point out that depending on the use case of the cloned models, intervening in the data collection phase is simply not possible or desirable.

When attempting to clone the human agent we should model the player improvement in some manner. It might be prudent to again first consider the easier case of reinforcement learning agents, but unlike in this paper allow them to improve during the data collection phase.

Although the performance measures we developed can be useful, they are not sufficient to characterise whether an agent performs well or not. It might be possible to use machine learning to obtain better measures. However, we should be wary of the required amount of data in the case of human agents, and should also take into account we do not just want to be able to distinguish a base agent from its clone, but judge whether they seem similar enough according to human judgement. This would require a lot of (expensive) human annotation. Alternatively such machine learned measures might still be useful to improve the training process, similar to generative adversarial networks.

We attempted to supply the models with some local history by stacking game frames. We found that compared to not using any history at all this certainly helped. Perhaps an even larger improvement might be gained by utilising recurrent neural networks, LSTMs or transformers.

Finally, it is possible that training agents using inverse reinforcement learning techniques might outperform the behavioural cloning approach highlighted in this paper.

The source code used for this project is available at https://github.com/LaurensDiels/Behaviourally-cloning-River-Raid-agents.

Acknowledgements

The authors would like to thank the Flemish Supercomputer Centre VSC and the KU Leuven for allowing the use of their servers through introductory credits. Most of the neural network training was done on these servers.

The authors are also grateful to an anonymous reviewer for pointing out the issue of covariate shift, and DAgger and DART's attempts to alleviate it, as well as for suggesting LSTMs and transformers instead of basic recurrent neural networks.

References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Bergstra, J. S.; Bardenet, R.; Bengio, Y.; and Kégl, B. 2011. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, 2546–2554.

Bergstra, J.; Yamins, D.; and Cox, D. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, 115–123.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

Dozat, T. 2016. Incorporating Nesterov momentum into Adam. *ICLR Workshop*.

Fasano, G., and Franceschini, A. 1987. A multidimensional version of the Kolmogorov–Smirnov test. *Monthly Notices of the Royal Astronomical Society* 225(1):155–170.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

Holsheimer, K. 2019. keras-gym. https://github.com/KristianHolsheimer/keras-gym.

Laskey, M.; Lee, J.; Fox, R.; Dragan, A.; and Goldberg, K. 2017. DART: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Pfau, J.; Smeddinck, J. D.; and Malaka, R. 2018. Towards deep player behavior models in MMORPGs. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, 381–392.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Soni, B., and Hingston, P. 2008. Bots trained to play like a human are more fun. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 363–369. IEEE.