Aligning Agents like Large Language Models

Adam Jelley[†]
University of Edinburgh

Yuhan Cao
Microsoft Research Cambridge

Dave Bignell
Microsoft Research Cambridge

Sam Devlin
Microsoft Research Cambridge

Tabish Rashid
Microsoft Research Cambridge

Abstract

Training agents to behave as desired in complex 3D environments from high-dimensional sensory information is challenging. Imitation learning from diverse human behavior provides a scalable approach for training an agent with a sensible behavioral prior, but such an agent may not perform the specific behaviors of interest when deployed. To address this issue, we draw an analogy between the undesirable behaviors of imitation learning agents and the unhelpful responses of unaligned large language models (LLMs). We then investigate how the procedure for aligning LLMs can be applied to aligning agents in a 3D environment from pixels. For our analysis, we utilize an academically illustrative part of a modern console game in which the human behavior distribution is multi-modal, but we want our agent to imitate a single mode of this behavior. We demonstrate that we can align our agent to consistently perform the desired mode, while providing insights and advice for successfully applying this approach to training agents. Project webpage at: https://adamjelley.github.io/aligning-agents-like-llms/.

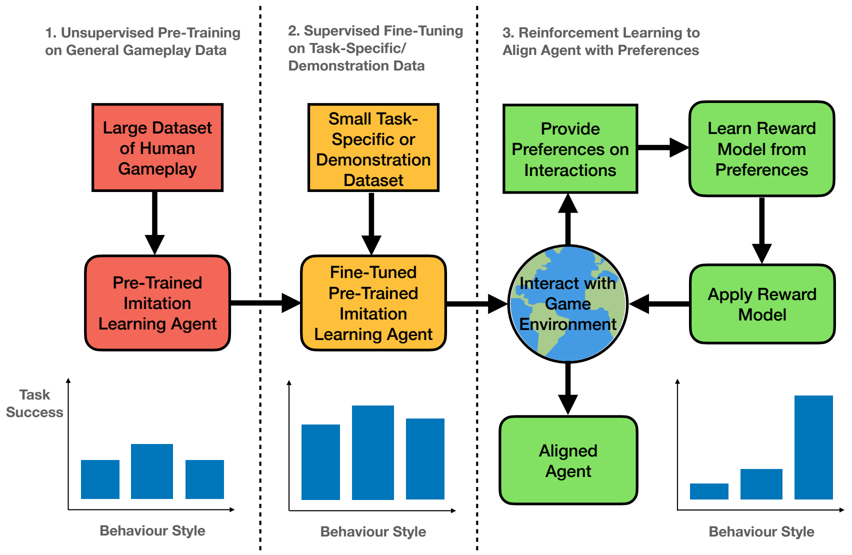

1. Unsupervised Pre-Training on General Gameplay Data
2. Supervised Fine-Tuning on Task-Specific/ Demonstration Data
3. Reinforcement Learning to Align Agent with Preferences
Large Dataset of Human Gameplay
Pre-Trained Imitation Learning Agent
Small Task-Specific or Demonstration Dataset
Fine-Tuned Pre-Trained Imitation Learning Agent
Provide Preferences on Interactions
Learn Reward Model from Preferences
Interact with Game Environment
Apply Reward Model
Aligned Agent
Task Success
Behaviour Style
Behaviour Style
Behaviour Style

Figure 1: Illustration of our approach for aligning a generally capable agent with a game designer's intended goals or preferences. A general agent pre-trained to imitate a large diverse dataset of human gameplay provides a base agent which can be more easily fine-tuned to imitate a smaller task or demonstration dataset. This agent can then be further fine-tuned with reinforcement learning using a reward model learned from preferences to reliably achieve the target behavior with the desired style. This approach is analogous to the alignment procedure used for recent large language models.

[†]Work done during internship at Microsoft Research Cambridge. Correspondence to: adam.jelley@ed.ac.uk

1 Introduction

The optimal approach for training competent agents to act in complex 3D environments without carefully crafted reward functions is an open question. Many modern console games provide such 3D environments, where the state space, action space and temporal horizons are large enough that learning from scratch is usually infeasible, even with a clearly defined goal. In these environments, a natural approach is to leverage a large dataset of general human behavior to pre-train an agent with imitation learning. This provides an agent with a general understanding of player behavior, and there is evidence of generalization benefits from scaling up data and compute (Baker et al., 2022; Reed et al., 2022; Brohan et al., 2023; Bousmalis et al., 2023; SIMA-Team et al., 2024).

However, such an agent will inevitably learn to imitate all behaviors found within the human gameplay, including undesirable behaviors of novice or malicious players, analogous to unhelpful or toxic responses of unaligned large language models (Ziegler et al., 2020). Additionally, game designers may have preferences for agents to act with a certain style or strategy, for which it may be difficult to codify a suitable reward function (Aytemiz et al., 2021; Devlin et al., 2021). There is a parallel between training useful agents and aligning large language models. In the same way that LLMs have been aligned to serve as customer service chatbots (Banerjee et al., 2023), search engines (Spathari-oti et al., 2023) and code generation assistants (Chen et al., 2021b), we can imagine aligning a large imitation learning agent with various objectives within a game. For example, we may wish to train agents that can act as allies, opponents (Vinyals et al., 2019), non-player-characters (NPCs) (Alonso et al., 2020), or even for quality assurance during development (Bergdahl et al., 2021).

In this work, we take inspiration from the current procedure for training large language models (LLMs) and investigate applying this procedure to train agents. We consider a situation where the human behavior distribution is distinctly multi-modal over non-negligible time horizons (of the order of ~ 10 seconds), but we desire our agent to imitate a single mode of this distribution. This particular mode of behavior may be difficult to learn from scratch with reinforcement learning since there is no clear reward function, but an imitation learning agent pre-trained on general gameplay data would only sometimes sample the desired behavior mode. At the same time, it may be infeasible to obtain sufficient clean demonstrations for robust imitation learning. To isolate this problem, we focus on an academically illustrative part of a modern console game where players must navigate from a randomly selected spawn point to one of three jumppads. We find that a base imitation learning agent reaches all three jumppads in similar proportions to the human data. We then demonstrate that we can align this agent to consistently reach a single preferred jumppad, using synthetic preference labelling to train a reward model followed by online reinforcement learning.

Our contributions include: 1) An analysis of the importance and potential difficulties of applying each stage of the current LLM training pipeline to agents, including unsupervised pre-training, supervised fine-tuning, preference modelling and online alignment, 2) Evidence of representation transfer between imitation learning and preference learning that highlights the benefits of using the base agent to initialize the reward model when learning preferences from pixels, and 3) The introduction of an additional training stage, *preference fine-tuning*, to substantially improve alignment efficiency. We perform the entire procedure from visual inputs with full controller action output, maintaining the generality of our approach to other game environments, and demonstrating that the modern LLM training paradigm can be successfully applied to aligning agents on a real 3D game.

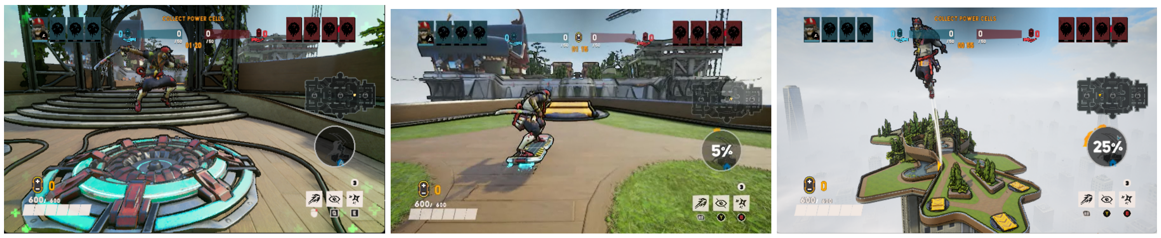
2 Game Environment and Alignment Goal

For this paper, we utilize the video game Bleeding Edge[1], which was launched in 2020 for Xbox One[2]. It is a team-based 4v4 online multi-player video game. Players select from thirteen possible heroes, each with different abilities. The game is played with a third-person view, with the camera angle controlled by the player, so the environment is partially observable. Here we focus on a single map, called Skygarden. This map is spread over three islands each with multiple elevation levels, including a main island and two launch islands (one for each team).

[1] https://www.bleedingedge.com/en [2] https://www.pcgamingwiki.com/wiki/Bleeding_Edge

2.1 Alignment Objective

At the beginning of the game, players spawn at one of four nearby points on their team's launch island. From the spawn location a player must navigate across the launch island to one of three jumppads that launch the player from their launch island onto the main island. Depending on the jumppad selected, the player will be launched onto different areas of the main island, so players may wish to take different jumppads during a match depending on the location of opposing team players. For the purpose of this work, we aim to train an agent to consistently navigate across the launch island to a single one of the three jumppads. Since this navigation task requires around 10 seconds to complete for an optimal agent, a random agent will rarely leave the spawn area. Additionally, without access to privileged information such as the agent location (we only provide visual input as below) and an externally shaped reward function, it would be difficult to train an agent to complete this task. On the other hand, an imitation learning agent will learn to reach all three of the jumppads, as in the human distribution. Therefore while the task is simple, it provides a clear motivation for alignment and a setting which we can quantitatively analyse.



(a) (b) (c)

Figure 2: Screenshots of the agent at a spawn point (2a), heading towards the middle jumppad (2b), and looking back at the launch island after having launched from the right jumppad (2c).

2.2 Observation and Action Spaces

To maintain the generality of our approach to any 3D environment, we take visual gameplay observations $\mathbf{o}_t \in \mathbb{R}^{H\times W\times 3}$ as input, sampled at 10Hz. We do not provide any privileged information, so the agent only has access to input information available to human players. The action space consists of 12 buttons and 2 joysticks. The left joystick controls the movement of the agent, while the right joystick controls the camera angle. We decompose each joystick into an x and a y component which are independently discretised into 11 buckets. This creates a $12 + 2 + 2 = 16$ dimensional discrete action space in total, although the joystick dimensions are of most relevance to our objective.

3 Implementation and Analysis

3.1 Training a Base Imitation Learning Model

We now follow the general procedure outlined in Figure 1 and Appendix A and describe our specific implementation of this procedure for training an agent aligned to reliably reach a preferred jumppad. We begin by performing imitation learning from visual observations to gamepad actions on general gameplay data to provide a behavioral prior with a general understanding of human gameplay.

Dataset: For general pre-training, a dataset was extracted from recorded human gameplay, as described in Appendix B. This large unfiltered dataset consists of 71,940 individual player trajectories from 8788 matches recorded between 09-02-2020 and 10-19-2022, which amounts to 9875 hours (1.12 years) of individual gameplay. For the purposes of this work, we use an agent that was trained for roughly one epoch on this dataset. Considering the success of recent foundation models, we note that given our unified observation and action spaces, it may also be possible to train across games to obtain a base model, as explored in previous work (Reed et al., 2022; SIMA-Team et al., 2024).

Architecture and Training: For the policy, we use a GPT-2 (Radford et al., 2019) causal transformer architecture with 103M parameters, similar to that used by VPT (Baker et al., 2022). Observations from the human gameplay $\mathbf{o}_t \in \mathbb{R}^{H \times W \times 3}$ are taken directly as input to a convolutional encoder to give observation embeddings $\mathbf{z}_t$. The transformer is trained with a context window of $H = 32$ timesteps (corresponding to around 3s of gameplay given the 10Hz sampling). The context window is important since the game is partially observable: as the context window is increased, the agent is able to better capture the state of the environment and take more informed actions (i.e. with a more Markovian state) at the cost of computational complexity. The output corresponds to the 16 discrete action dimensions. The transformer and convolutional encoder are both trained end to end with a cross-entropy loss over all output action components to provide a policy $\pi(a_t|o_t, ...o_{t-H})$. Full architecture and training details are provided in Appendix C.

Evaluation: Once trained, we ran our agent online in the game environment and recorded which jumppad was reached for 1000 episodes. We initialized the agent using the *Ninja* character with an empty context buffer at one of the spawn points at random. We ran the agent until it reached one of the jumppads or timed out. The jumppad distribution reached is demonstrated in Figure 3 below.

We find that our base model reaches a jumppad 56% of the time, and has a bias towards the middle jumppad, reflecting human behavior. Our base model therefore provides a reasonable behavioral prior that is much better than random exploration of the launch island (which has a 0% success rate). However, the success rate could be improved, which is likely due to distribution shift between the offline data and the online environment. For example, the base model was trained on data containing all thirteen possible characters, while our online evaluation only uses the default *Ninja* character. Additionally, we initialize our agents online at the spawn point with an empty context buffer, while in training the agent will have access to context including a spawn animation and prior gameplay. While measures can be taken to avoid distribution shift, offline only learning is inherently challenging (Ostrovski et al., 2021) and some distribution shift is usually inevitable, as we discuss further in Appendix D.

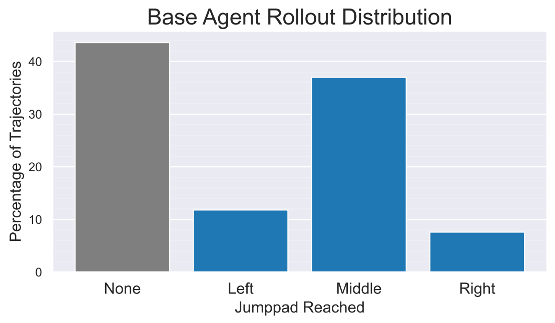


Figure 3: Distribution of jumppads reached by the base imitation learning agent.

3.2 Supervised Fine-Tuning on Task Relevant Dataset

Following the LLM pipeline, we now fine-tune our agent on a smaller curated dataset. Here we fine-tune on a high quality subset of the pre-training data. However, more generally this could consist of fine-tuning on demonstrations of desireable behavior by the game designer.

Dataset: We curated 300 successful trajectories (100 per jumppad) for fine-tuning, all of which involved the character Ninja and were filtered to contain only the first part of the trajectory until a jumppad was reached. We fine-tune until convergence as described in Appendix C.

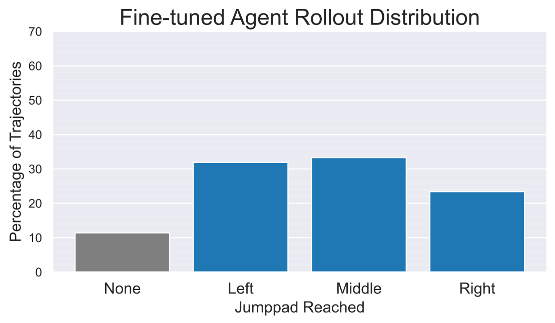


Figure 4: Distribution of jumppads reached by the fine-tuned imitation learning agent.

Results: We evaluate our agent following the same procedure as before. We see in Figure 4 that the success rate of reaching a jumppad is now substantially higher (89%), and that the agent more evenly reaches all three jumppads (corresponding to the balanced fine-tuning dataset). At this stage, it is natural to ask whether the general pre-training was beneficial or whether the model could have been trained from scratch on our task relevant dataset directly. We find that the agent without pre-training has a less diverse jumppad distribution, and a higher failure rate (around 20% of trajectories fail to reach a jumppad compared to only around 10% for the pre-trained agent). Interestingly, from inspecting agent behaviors we found that the pre-trained agent appeared to be more robust since it can often better 'self-correct' its trajectories if it drifts out-of-distribution of the fine-tuning dataset. Full details of our pre-training ablation and an example of this self-correction behavior are provided in Appendix E.1. These results corroborate similar findings for LLMs, in which pre-training transformer models has been shown to enable more effective fine-tuning, effectively increasing the size of the fine-tuning data compared to training from scratch (Hernandez et al., 2021).

We also perform a preliminary investigation of model scaling by training additional models from scratch on this smaller task-specific dataset in Appendix E.2. We find that smaller models have a higher failure rate, even trained only on this relatively small fine-tuning dataset. While larger models than we consider may perform even better (particularly with pre-training), this would increase the inference cost which has a substantial impact on online alignment speed (and the potential for the final agent to behave in real-time), so we proceed with the 103M parameter model described above.

3.3 Obtaining Preference Data on Online Rollouts

We now deploy the fine-tuned agent in the environment to collect trajectories for preference labelling. This is analogous to generating multiple LLM responses to a prompt, but in this context the prompt becomes the initial observation and optional context of any previous observations and actions provided. Similarly to LLMs, there are tradeoffs to be made in the diversity, quality and quantity of preferences when provided by humans (discussed further in Appendix K), but here we avoid such issues by utilizing synthetic preferences to isolate the effect of quantity.

Online Rollouts: We initialize the Ninja character with an empty context buffer at a random spawn point, equivalent to the evaluation procedure above. We repeated this procedure to generate a video dataset of 2400 on-policy trajectories, divided into 1000 trajectories for training and 1400 for evaluation. Similarly to LLMs, the temperature of the softmax sampling of the policy for action selection can be increased to generate more diverse behaviors from the agent for easier comparison, but here we found the behavior to be diverse enough with a default temperature parameter of 1.

Generating Preferences: We utilize synthetic preferences based on the primary criteria:

$$\textit{Preferred Jumppad Reached} > \textit{Other Jumppad Reached} > \textit{No Jumppad Reached}$$

Within each of these primary categories, we further rank trajectories by their duration, with shorter trajectories being preferred. By selecting subsets of trajectories in the training dataset, we are able to investigate how reward model performance scales with number of comparisons (which is a proxy for the human labelling time requirement, often the main bottleneck for reinforcement learning from human feedback, or RLHF). A preference dataset P is then created by considering all pairwise comparisons of trajectories τ using the criteria above, i.e. $(\tau_A \succ \tau_B) \in P \ \forall \ \tau_A, \tau_B \in \tau$ if $\tau_A \succ \tau_B$.

3.4 Training a Reward Model on Preferences

A reward model is then trained on this dataset to predict rewards in accordance with preferences as described below. Previous work on RLHF for agents has generally relied on simple (often linear) reward models (Knox & Stone, 2008; Christiano et al., 2017). However recent work on LLMs has demonstrated that reward models that utilize the pre-trained or fine-tuned policy model with the action classification head replaced with a scalar regression head generally perform better (Stiennon et al., 2020), and also improve with scale (Ouyang et al., 2022; Touvron et al., 2023). It is hypothesised that this is because this allows the reward model to benefit from the pre-training, which helps to the reward model to better distinguish behaviors for reward assignment, since the reward model shares the agent's knowledge of the behavior space (Ziegler et al., 2020). We investigate this potential for representation transfer in the context of agents in this section.

Procedure: We follow the standard Bradley-Terry (Bradley & Terry, 1952) model procedure to train a reward model $\hat{r}$ from these pairwise preferences, as introduced by Christiano et al. (2017). Specifically, we interpret trajectory rewards as preference rankings analogous to Elo ratings (Elo, 1978) developed for chess, such that the annotator's probability of preferring a trajectory depends exponentially on the trajectory reward difference. We can then fit the reward model by minimising the cross-entropy between these probabilities and the preference labels, which gives the loss function:

$$\mathcal{L}(\hat{r}) = \sum_{(\tau_w,\tau_l)\in P} -\log\left(\sigma\left(\hat{r}(\tau_w) - \hat{r}(\tau_l)\right)\right) \tag{1}$$

where σ is the sigmoid function and (τ_w, τ_l) are the trajectories being compared, with τ_w being the winning (preferred) trajectory and τ_l being the losing trajectory. Since the reward model is only trained on comparisons, the scale of the predicted rewards is arbitrary. As a result, we found the need to apply a small amount of $L2$ regularisation to prevent the scale of the rewards becoming overly large for the best and worst trajectories (overfitting). We further empirically normalise the reward model after training using the max and min of predicted rewards over the training trajectories to scale our reward model output to be in the range $\hat{r} \in [0, 1]$ (a crucial but often overlooked detail).

Architecture: Following the LLM procedure to use the agent model for the reward model initialization, we take the outputs of the fine-tuned agent's transformer as embeddings for each timestep, which are fed into an MLP reward model to produce a scalar return for the trajectory which is then input to the loss function in Equation 1. As an ablation, to investigate whether these observation representations used by the agent for imitation learning are also beneficial for reward modelling (i.e. predicting preferences), we consider a reward model with an equivalent architecture but with a randomly initialized linear encoder to provide random projections as embeddings for each timestep. We also investigate how reward model performance scales with number of comparisons, to obtain an estimate of the human time required to provide sufficient feedback for training the reward model.

Results: To measure the performance of our reward models, we apply each reward model to our held-out test trajectories. We compute pairwise preferences according to the reward models, and compare to the ground truth pairwise preferences to obtain a test preference accuracy. Reward model performances shown in Figure 5.

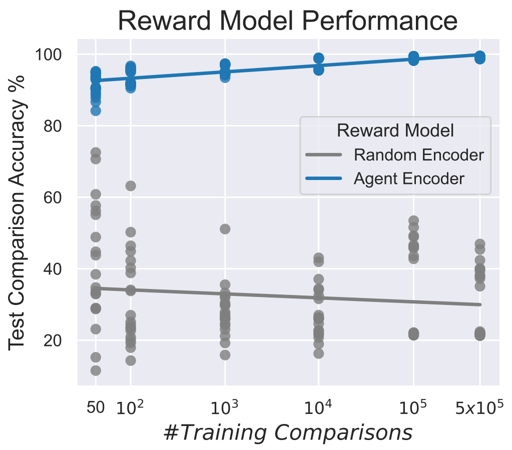


Figure 5: Reward model test performances.

We find that reward model accuracy generally increases with number of comparisons, although the random projection reward models have high variance. Most strikingly, the reward models utilizing the agent model perform better than the reward models using random projections across the full range of comparison sizes, suggesting that the imitation learning agent's representations of the observations contains information beneficial to predicting preferences. We further find that when using this encoder it is possible to train a reward model from visual input to achieve over 90% preference accuracy with only $\sim$ 100 comparisons. While we use synthetic preferences, we note that this would correspond to less than 1 hour of labelling time for our task.

3.5 Aligning the Fine-tuned Model with the Reward Model

Procedure: Finally, we align our agent with our preferences as captured by our reward models. We run our fine-tuned agent in the environment as described in Section 2 to generate online trajectories. After each trajectory is complete, we apply our reward model to the trajectory to generate a reward corresponding to that trajectory. We use this reward as the return for that trajectory and update our agent policy π using an undiscounted REINFORCE (Williams, 1992) loss. While much of the RLHF literature uses PPO (Schulman et al., 2017), we note that concurrent work on LLMs has found PPO to be unnecessarily complex for RLHF (Ahmadian et al., 2024). This is due to the fact that in the standard LLM framework the reward is only provided at the end of the trajectory, meaning that we generally just want to reinforce entire trajectories rather than attempt credit assignment at each timestep using an advantage function. Additionally, Ahmadian et al. (2024) found that clipping rarely occurs in practice since the initial fine-tuned policy is close to the final aligned policy. This is reflected by the use of REINFORCE in recent state-of-the-art LLMs (Google Deepmind, 2024).

For our experiments we ran our agent online for 9600 episodes (corresponding to around 1 day of real time gameplay), using a batch size of 16 to give 600 parameter updates. We fine-tune the last layer only since we want to maintain the general pre-trained representations, but note that approaches such as Low-Rank Adaption (LoRA) (Hu et al., 2021) used for fine-tuning LLMs could also be used here. We also investigated the use of an additional KL divergence term to regularise the optimized policy towards the initial fine-tuned policy, as is commonly used in LLM alignment (Touvron et al., 2023; Bai et al., 2022), but found it unnecessary. This is likely due to the fact that we fine-tune only the last layer rather than the entire network, which mitigates policy divergence.

3.5.1 Aligning Agent Towards Left Jumppad

We begin by focusing on aligning our agent towards the left jumppad. We plot the average success rate of reaching the left jumppad against the number of episodes with reward models that have been trained on 100 up to 500k comparisons in Figure 6. We see that all of our reward models are sufficient to align our agent to consistently reach the left jumppad, with reward models trained on more data (with higher test performances) generally leading to better alignment.

However, we see that the agents take most of the 600 updates to fully align, corresponding over a day of real time training. To improve this efficiency, we consider the addition of a *preference fine-tuning* phase in which we first further fine-tune on the preferred trajectories. Specifically, we apply the reward model to the training trajectories and take the top 20% of trajectories by reward (the preferred trajectories) and perform additional fine-tuning on these trajectories. Similar approaches have recently been introduced to improve the simplicity and efficiency of language model alignment (Gulcehre et al., 2023). We find that this improves performance for all reward models, shown in Figure 6.

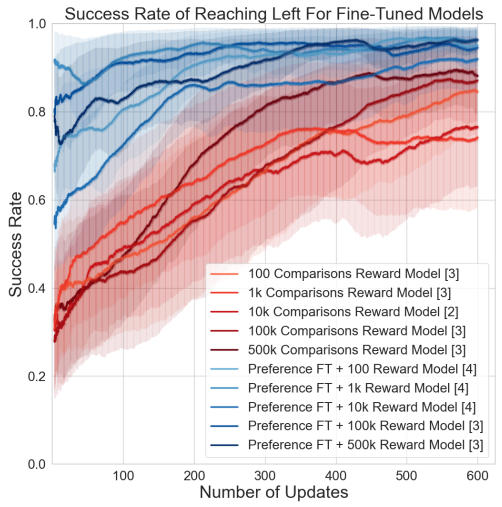


Figure 6: Left jumppad success rate during online alignment. Standard errors shaded, and number of seeds in legend. We see that: 1) Higher accuracy reward models (darker) generally lead to better alignment, 2) Preference fine-tuning (blue) on preferred trajectories before online alignment improves performance for all reward models.

3.5.2 Aligning Agent Towards Right Jumppad

We now consider aligning our agent towards the right jumppad. Since this task is seemingly of identical difficulty, we would expect to be able to align the agent similarly. However, we found that our agents did not align as quickly towards the right jumppad, and in the absence of preference fine-tuning only reached around a $\sim 40\%$ success rate after one day of training, as shown in Figure 7.

In order to investigate this discrepancy, we analyzed the rollouts of the fine-tuned agent, shown in Figure 8. We see that the trajectories that reach the left jumppad start relatively evenly from the 4 spawn locations. However, of the trajectories that successfully reach the right jumppad, only 1% originated from the spawns on the right hand side. Therefore it is unsurprising that it is more difficult for the agent to learn to go right, given that in half the episodes (corresponding to the right spawn locations) the agent rarely navigates to the right jumppad to receive reward to reinforce its behavior.

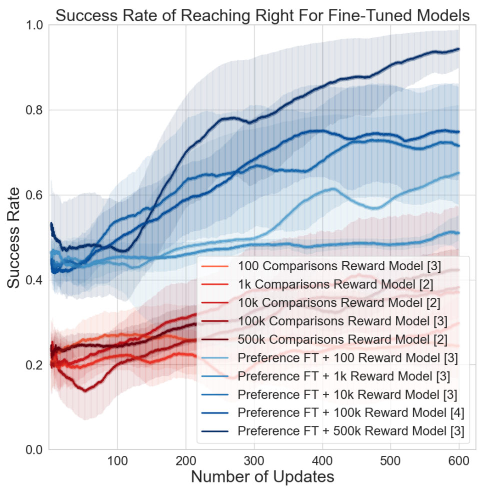


Figure 7: Right jumppad success rate during online alignment. Standard errors shaded, and number of seeds in legend. Similar trends hold to Figure 6, but alignment is less effective.

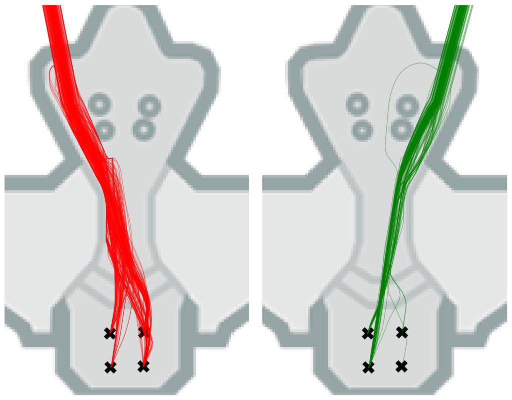


Figure 8: Fine-tuned agent trajectories which reached the left (red) or right (green) jumppads. While fine-tuning enables more efficient preference labelling, it reduces diversity, potentially limiting alignment.

To confirm this hypothesis, we instead attempted to align the base agent (before fine-tuning), and found that it can indeed be aligned more symmetrically, although with a lower final success rate, as discussed in Appendix G. The addition of preference fine-tuning before online alignment helps to mitigate this imbalance, as shown in Figure 7, since it modifies the agent to produce more of the preferred behaviors that achieve reward before online alignment. This enabled us to completely align the agent with the same training budget, as shown by the final jumppad distributions in Appendix H. This further demonstrates the benefits of our proposed additional stage of preference fine-tuning.

More generally, these results highlight the importance of maintaining diversity of behaviour when attempting to align agents (Touvron et al., 2023; Casper et al., 2023). At every stage of the alignment pipeline, the agent becomes increasingly specialised and less diverse in its behavior. While this generally leads to more efficient learning and better alignment, it can also lead to less general and more brittle policies. Additional research on improving the robustness and efficiency of this procedure for the more limited and unbalanced datasets available in the context of agents will be important for practical application of the LLM paradigm to gameplaying agents and beyond.

A heatmap illustrating our agent's behaviour at each stage of alignment is shown in Appendix I. Additional related work is covered in Appendix J. Finally, videos of our agent during each stage of alignment are provided at: `https://adamjelley.github.io/aligning-agents-like-llms/`.

4 Conclusion

In this paper, we have drawn an analogy between training agents and training large language models. We demonstrated that the current LLM training procedure can be used to align agents to reliably perform desired behaviors in complex environments on a modern console game, with only a small amount of task/demonstration data. This enabled us to train an agent to achieve specific modes of behavior in the game that would be difficult to achieve reliably with imitation learning, reinforcement learning or RLHF alone. Our analysis shows that many of the recent developments in the current procedure for training large language models, such as general pre-training and initializing reward models from the pre-trained agent, can be applied and have similar benefits for training agents. We also demonstrated improved alignment efficiency through the inclusion of an additional preference fine-tuning stage. We hope that our work encourages further communication and collaboration between the large language model and gaming communities, to enable shared insights and provide a path towards practical and reliable deployment of reinforcement learning agents in modern games.

Broader Impact Statement

This work considers aligning imitation learning agents with human preferences. The preferences considered in this work are synthetic for analysis purposes, but in general the source of the preference data is an important consideration to understand any biases in alignment.

More generally, research into aligning agents with preferences is important to ensure that agents are helpful and harmless. However, this procedure has various known open problems and limitations (Casper et al., 2023). Video games therefore provide an important test bed for such research, helping to mitigate risks and provide insights in a safe environment that may generalize to other more real-world applications such as robotics.

Acknowledgments

This project was done during an internship with the Game Intelligence Team at Microsoft Research Cambridge. We are grateful to Ninja Theory and Microsoft for their support with the project. Thank you to Yuhan Cao and Dave Bignell for implementation and infrastructure support, Sam Devlin for advice and guidance, and particularly Tabish Rashid for supervising the internship and seeing the project through to completion. In addition to the authors, thank you to the whole team, including Anssi Kanervisto, Gunshi Gupta, Katja Hofmann, Lukas Schäfer, Raluca Georgescu, Sergio Valcarcel Macua, Shanzheng Tan, Tarun Gupta, Tim Pearce and Eloi Alonso for a great internship. Thank you also to Amos Storkey, Tom Lee, Trevor McInroe and Rich Turner for insightful discussions. Outside of the internship, Adam Jelley is supported by Microsoft Research and EPSRC through Microsoft's PhD Scholarship Programme.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Twenty-first international conference on Machine learning - ICML '04*, pp. 1, Banff, Alberta, Canada, 2004. ACM Press. doi: 10.1145/1015330.1015430. URL http://portal.acm.org/citation.cfm?doid=1015330.1015430.

Josh Abramson, Arun Ahuja, Federico Carnevale, Petko Georgiev, Alex Goldin, Alden Hung, Jessica Landon, Jirka Lhotka, Timothy Lillicrap, Alistair Muldal, George Powell, Adam Santoro, Guy Scully, Sanjana Srivastava, Tamara von Glehn, Greg Wayne, Nathaniel Wong, Chen Yan, and Rui Zhu. Improving Multimodal Interactive Agents with Reinforcement Learning from Human Feedback, November 2022. URL http://arxiv.org/abs/2211.11602. arXiv:2211.11602 [cs].

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs, February 2024. URL http://arxiv.org/abs/2402.14740. arXiv:2402.14740 [cs].

Eloi Alonso, Maxim Peter, David Goumard, and Joshua Romoff. Deep Reinforcement Learning for Navigation in AAA Video Games, November 2020. URL http://arxiv.org/abs/2011.04764. arXiv:2011.04764 [cs].

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, August 2021. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103500. URL https://www.sciencedirect.com/science/article/pii/S0004370221000515.

Batu Aytemiz, Mikhail Jacob, and Sam Devlin. Acting with Style: Towards Designer-centred Reinforcement Learning for the Video Games Industry. In *CHI 2021 Workshop on Reinforcement Learning for Humans, Computer, and Interaction (RL4HCI)*. Association for Computing Machinery (ACM), May 2021. URL https://www.microsoft.com/en-us/research/publication/acting-with-style-towards-designer-centred-reinforcement-learning-for-the-video-games-industry/.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. URL http://arxiv.org/abs/1607.06450. arXiv:1607.06450 [cs, stat].

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, April 2022. URL http://arxiv.org/abs/2204.05862. arXiv:2204.05862 [cs].

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos, June 2022. URL http://arxiv.org/abs/2206.11795. arXiv:2206.11795 [cs].

Debarag Banerjee, Pooja Singh, Arjun Avadhanam, and Saksham Srivastava. Benchmarking LLM powered Chatbots: Methods and Metrics, August 2023. URL http://arxiv.org/abs/2308.04624. arXiv:2308.04624 [cs].

James Bennett, Stanley Lanning, and Netflix Netflix. The Netflix Prize. January 2009.

Joakim Bergdahl, Camilo Gordillo, Konrad Tollmar, and Linus Gisslén. Augmenting Automated Game Testing with Deep Reinforcement Learning, March 2021. URL http://arxiv.org/abs/2103.15819. arXiv:2103.15819 [cs].

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. RoboCat: A Self-Improving Generalist Agent for Robotic Manipulation, December 2023. URL http://arxiv.org/abs/2306.11706. arXiv:2306.11706 [cs].

Ralph Allan Bradley and Milton E. Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 0006-3444. doi: 10.2307/2334029. URL https://www.jstor.org/stable/2334029. Publisher: [Oxford University Press, Biometrika Trust].

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale, December 2022. URL http://arxiv.org/abs/2212.06817. arXiv:2212.06817 [cs].

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, July 2023. URL http://arxiv.org/abs/2307.15818. arXiv:2307.15818 [cs].

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback, July 2023. URL http://arxiv.org/abs/2307.15217. arXiv:2307.15217 [cs].

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. *arXiv:2106.01345 [cs]*, June 2021a. URL http://arxiv.org/abs/2106.01345. arXiv: 2106.01345.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, July 2021b. URL http://arxiv.org/abs/2107.03374. arXiv:2107.03374 [cs].

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, July 2017. URL http://arxiv.org/abs/1706.03741. arXiv:1706.03741 [cs, stat].

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling Instruction-Finetuned Language Models, December 2022. URL http://arxiv.org/abs/2210.11416. arXiv:2210.11416 [cs].

Sam Devlin, Raluca Georgescu, Ida Momennejad, Jaroslaw Rzepecki, Evelyn Zuniga, Gavin Costello, Guy Leroy, Ali Shaw, and Katja Hofmann. Navigation Turing Test (NTT): Learning to Evaluate Human-Like Navigation, July 2021. URL http://arxiv.org/abs/2105.09637. arXiv:2105.09637.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present.* Arco Pub., 1978. ISBN 978-0-668-04721-0. Google-Books-ID: 8pMnAQAAMAAJ.

Gemma Team Google Deepmind. Gemma: Open Models Based on Gemini Research and Technology. Technical report, February 2024. URL https://storage.googleapis.com/deepmind-media/gemma/gemma-report.pdf.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced Self-Training (ReST) for Language Modeling, August 2023. URL http://arxiv.org/abs/2308.08998. arXiv:2308.08998 [cs].

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv: 1512.03385.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), June 2023. URL http://arxiv.org/abs/1606.08415. arXiv:1606.08415 [cs].

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling Laws for Transfer, February 2021. URL http://arxiv.org/abs/2102.01293. arXiv:2102.01293 [cs].

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from Demonstrations, November 2017. URL http://arxiv.org/abs/1704.03732. arXiv:1704.03732 [cs].

Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning, June 2016. URL http://arxiv.org/abs/1606.03476. arXiv:1606.03476 [cs].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. URL http://arxiv.org/abs/2106.09685. arXiv:2106.09685 [cs].

Jian Hu, Li Tao, June Yang, and Chandler Zhou. Aligning Language Models with Offline Reinforcement Learning from Human Feedback, August 2023. URL http://arxiv.org/abs/2308.12050. arXiv:2308.12050 [cs] version: 1.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/8cbe9ce23f42628c98f80fa0fac8b19a-Abstract.html.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. URL http://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs, stat].

Changyeon Kim, Jongjin Park, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Preference Transformer: Modeling Human Preferences using Transformers for RL, March 2023. URL http://arxiv.org/abs/2303.00957. arXiv:2303.00957 [cs].

Bradley Knox and Peter Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pp. 292–297, Monterey, CA, August 2008. IEEE. ISBN 978-1-4244-2661-4. doi: 10.1109/DEVLRN.2008.4640845. URL http://ieeexplore.ieee.org/document/4640845/.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline Q-Learning on Diverse Multi-Task Data Both Scales And Generalizes, November 2022. URL http://arxiv.org/abs/2211.15144. arXiv:2211.15144 [cs].

Kimin Lee, Laura Smith, and Pieter Abbeel. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training, June 2021. URL http://arxiv.org/abs/2106.05091. arXiv:2106.05091 [cs].

Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, and Igor Mordatch. Multi-Game Decision Transformers, October 2022. URL http://arxiv.org/abs/2205.15241. arXiv:2205.15241.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, November 2020. URL http://arxiv.org/abs/2005.01643. arXiv:2005.01643 [cs, stat].

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s, March 2022. URL http://arxiv.org/abs/2201.03545. arXiv:2201.03545 [cs].

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs, math].

Stephanie Milani, Anssi Kanervisto, Karolis Ramanauskas, Sander Schulhoff, Brandon Houghton, Sharada Mohanty, Byron Galbraith, Ke Chen, Yan Song, Tianze Zhou, Bingquan Yu, He Liu, Kai Guan, Yujing Hu, Tangjie Lv, Federico Malato, Florian Leopold, Amogh Raut, Ville Hautamäki, Andrew Melnik, Shu Ishida, João F. Henriques, Robert Klassert, Walter Laurito, Ellen Novoseller, Vinicius G. Goecks, Nicholas Waytowich, David Watkins, Josh Miller, and Rohin Shah. Towards Solving Fuzzy Tasks with Human Feedback: A Retrospective of the MineRL BASALT 2022 Competition, March 2023. URL http://arxiv.org/abs/2303.13512. arXiv:2303.13512 [cs].

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the seventeenth international conference on machine learning*, pp. 663–670, 2000.

OpenAI. Introducing ChatGPT, 2022. URL https://openai.com/blog/chatgpt.

OpenAI. GPT-4 Technical Report, March 2023. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs].

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The Difficulty of Passive Learning in Deep Reinforcement Learning, October 2021. URL http://arxiv.org/abs/2110.14020. arXiv:2110.14020 [cs].

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, March 2022. URL http://arxiv.org/abs/2203.02155. arXiv:2203.02155 [cs].

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating Human Behaviour with Diffusion Models, March 2023. URL http://arxiv.org/abs/2301.10677. arXiv:2301.10677 [cs, stat].

Dean A. Pomerleau. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation*, 3(1):88–97, March 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.88. Conference Name: Neural Computation.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. February 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, May 2023. URL http://arxiv.org/abs/2305.18290. arXiv:2305.18290 [cs].

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent. Technical Report arXiv:2205.06175, arXiv, May 2022. URL http://arxiv.org/abs/2205.06175. arXiv:2205.06175 [cs] type: article.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, March 2011. URL http://arxiv.org/abs/1011.0686. arXiv:1011.0686 [cs, stat].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv: 1707.06347.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL https://www.nature.com/articles/nature16961. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 7587 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Computational science;Computer science;Reward Subject_term_id: computational-science;computer-science;reward.

SIMA-Team, Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, Stephanie C. Y. Chan, Jeff Clune, Adrian Collister, Vikki Copeman, Alex Cullum, Ishita Dasgupta, Dario de Cesare, Julia Di Trapani, Yani Donchev, Emma Dunleavy, Martin Engelcke, Ryan Faulkner, Frankie Garcia, Charles Gbadamosi, Zhitao Gong, Lucy Gonzales, Kshitij Gupta, Karol Gregor, Arne Olav Hallingstad, Tim Harley, Sam Haves, Felix Hill, Ed Hirst, Drew A. Hudson, Jony Hudson, Steph Hughes-Fitt, Danilo J. Rezende, Mimi Jasarevic, Laura Kampis, Rosemary Ke, Thomas Keck, Junkyung Kim, Oscar Knagg, Kavya Kopparapu, Andrew Lampinen, Shane Legg, Alexander Lerchner, Marjorie Limont, Yulan Liu, Maria Loks-Thompson, Joseph Marino, Kathryn Martin Cussons, Loic Matthey, Siobhan Mcloughlin, Piermaria Mendolicchio, Hamza Merzic, Anna Mitenkova, Alexandre Moufarek, Valeria Oliveira, Yanko Oliveira, Hannah Openshaw, Renke Pan, Aneesh Pappu, Alex Platonov, Ollie Purkiss, David Reichert, John Reid, Pierre Harvey Richemond, Tyson Roberts, Giles Ruscoe, Jaume Sanchez Elias, Tasha Sandars, Daniel P. Sawyer, Tim Scholtes, Guy Simmons, Daniel Slater, Hubert Soyer, Heiko Strathmann, Peter Stys, Allison C. Tam, Denis Teplyashin, Tayfun Terzi, Davide Vercelli, Bojan Vujatovic, Marcus Wainwright, Jane X. Wang, Zhengdong Wang, Daan Wierstra, Duncan Williams, Nathaniel Wong, Sarah York, and Nick Young. Scaling Instructable Agents Across Many Simulated Worlds, April 2024. URL http://arxiv.org/abs/2404.10179. arXiv:2404.10179 [cs].

Sofia Eleni Spatharioti, David M. Rothschild, Daniel G. Goldstein, and Jake M. Hofman. Comparing Traditional and LLM-based Search for Consumer Choice: A Randomized Experiment, July 2023. URL http://arxiv.org/abs/2307.03744. arXiv:2307.03744 [cs].

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/1f89885d556929e98d3ef9b86448f951-Abstract.html.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023. URL http://arxiv.org/abs/2307.09288. arXiv:2307.09288 [cs].

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards, October 2018. URL http://arxiv.org/abs/1707.08817. arXiv:1707.08817 [cs].

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, November 2019. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-019-1724-z. URL http://www.nature.com/articles/s41586-019-1724-z.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A Survey of Preference-Based Reinforcement Learning Methods. 2017.

Ruohan Zhang, Faraz Torabi, Garrett Warnell, and Peter Stone. Recent Advances in Leveraging Human Guidance for Sequential Decision-Making Tasks. *Autonomous Agents and Multi-Agent Systems*, 35(2):31, October 2021. ISSN 1387-2532, 1573-7454. doi: 10.1007/s10458-021-09514-w. URL http://arxiv.org/abs/2107.05825. arXiv:2107.05825 [cs].

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-Tuning Language Models from Human Preferences, January 2020. URL http://arxiv.org/abs/1909.08593. arXiv:1909.08593 [cs, stat].

A Discussion of General Procedure for Aligning Agents

We break down our procedure for aligning agents with preferences (see Figure 1) into five steps:

1. **Train a Base Imitation Learning Policy**
   The first ingredient for training large language models is to train a large, scalable transformer architecture with self-supervised next-token prediction on a diverse dataset of human text, to obtain a language prior. In the context of agents on modern console games, we interpret this as imitation learning to predict the next action taken in human gameplay data, to obtain a behavioural prior. Specifically this involves training the transformer autoregressively to learn a policy $p(a_t|o_t, ..., o_{t-H})$. For the purpose of this work, we consider an agent trained on diverse data within a particular game, but note that given our unified observation and action spaces (visual observations and gamepad actions), it would also be possible to train across games, as explored in previous work (Reed et al., 2022; Lee et al., 2022).

2. **Supervised Fine-Tune on a Task Relevant Dataset**
   The next step in the current LLM pipeline is to fine-tune the pre-trained 'foundation' model on task relevant data, such as instruction data (Chung et al., 2022). Pre-trained transformer models have been shown to fine-tune more effectively, essentially increasing the size of the fine-tuning data compared to training from scratch (Hernandez et al., 2021). For decision-making agents this involves fine-tuning by imitation learning on the task (or game) of interest. For specific behaviours, this could also consist of fine-tuning on demonstrations of the desired behaviour by the game designer. The training procedure is equivalent to step 1.

3. **Generate Preference Data on Online Trajectories**
   Fine-tuned LLMs are subsequently prompted to generate multiple responses which are compared by human labellers to provide preferences. In the context of agents, the prompt becomes the initial observation (and optionally the context of previous observations and actions). The agent is then rolled out from a given initial start state multiple times to collect multiple trajectories. Similarly to LLMs, the temperature of the softmax sampling of the policy for action selection can be increased to generate more diverse behaviours from the agent for easier comparison. A human (e.g. game designer) then provides preferences on these trajectories, such as preferring trajectories where the agent plays in a certain style.

4. **Train a Reward Model on Preferences**
   A reward model is then trained on these online trajectories, commonly using a Bradley Terry model (Bradley & Terry, 1952) for pairwise comparisons, so that the reward model provides higher reward for preferred trajectories. While this reward model is usually trained from scratch in the context of agents, the modern LLM procedure utilises the pre-trained or fine-tuned policy model by replacing the action classification head with a scalar regression head. This enables the reward model to also benefit from the pre-training, and share the same knowledge as the agent to reduce out-of-distribution behaviours (Ziegler et al., 2020).

5. **Align the Fine-tuned Model with the Reward Model**
   Finally the agent can be trained with online reinforcement learning, to maximise the reward provided by the reward model, thereby aligning the agent with the game designer's preferences. Since online reinforcement learning can be inefficient, we find that additional *preference fine-tuning* can be performed on the high reward trajectories before deploying it online to get the agent closer to the desired behaviour. A common failure mode at this stage is reward model over-optimisation, where the agent performs behaviours that maximise the reward model output but are not aligned with preferences (also known as reward hacking). If this occurs, regularisation towards the original policy can be added or steps 3-5 can be repeated to generate new preferences on the reward hacking behaviour which can be used to re-train the reward model. This may take multiple iterations (e.g. 5 reward model iterations were used for Llama2 (Touvron et al., 2023)), to eventually obtain an aligned agent.

This procedure combines the benefits of large scale pre-training to obtain a widely generalisable agent, with the benefits of reinforcement learning from preferences to obtain an aligned agent. Furthermore, it also combines the benefits of offline learning to avoid the well-known sample inefficiency of online reinforcement learning, with online fine-tuning to alleviate the well-known problems associated with imitation learning agents going out of distribution (Ross et al., 2011) by enabling active online learning (Ostrovski et al., 2021).

Our procedure for aligning agents with preferences can also be summarised in algorithmic form:

Algorithm 1 Aligning Agents with Preferred Behaviours

```
Require: Large diverse dataset $D$, task specific dataset $T$, environment $E$, transformer policy $\pi$
1: Train policy $\pi$ on large diverse dataset $D$ with imitation learning (Pomerleau, 1991)
       $\mathcal{L}_U(\pi) = \mathbb{E}_D\left[-\log \pi(a_t^\tau | o_t^\tau, o_{t-1}^\tau, ...)\right]$
2: Fine-tune policy $\pi$ on task specific dataset or demonstrations $T$
       $\mathcal{L}_{FT}(\pi) = \mathbb{E}_T\left[-\log \pi(a_t^\tau | o_t^\tau, o_{t-1}^\tau, ...)\right]$
3: Run fine-tuned policy $\pi$ in environment and provide preferences on pairwise comparisons on
   online trajectories to obtain preference data $(\tau_A \succ \tau_B) \in P$
4: Initialise additional reward model $\hat{r}$ from $\pi$ and train on preferences $P$ using Bradley-Terry
   model (Bradley & Terry, 1952)
       $\mathcal{L}(\hat{r}) = \sum_{(\tau_w, \tau_l) \in P} -\log\left(\sigma\left(\hat{r}(\tau_w) - \hat{r}(\tau_l)\right)\right)$
5: Train fine-tuned policy $\pi_{RL}$ using online reinforcement learning, using reward model $\hat{r}$ to provide
   return for online trajectory $\tau$ to align agent with preferences. For example, using REINFORCE
   (Williams, 1992):
       $\mathcal{L}_{RL}(\pi_{RL}) = -\mathbb{E}_{\pi_{RL}}\left[\sum_{t=1}^{T} \gamma^t \hat{r}(\tau) \log \pi_{RL}(a_t | o_t, ...)\right]$
```

B Bleeding Edge Game Human Data Collection

Human gameplay data was recorded as part of the regular gameplay process, in order to enable in-game functionality as well as to support research. In game, recordings allow players to view their past games to improve their skills and for entertainment. Games were recorded on the servers that hosted the games in the form of replay files, which include a representation of the internal game state and controller actions of all players.

Data collection was covered by an End User License Agreement (EULA) to which players agreed when logging in to play the game for the first time. Our use of the recorded human gameplay data for this specific research was governed by a data sharing agreement with the game studio, and approved by our institution's IRB. To minimize risks regarding data privacy, any personally identifiable information (Xbox user ID) was removed when extracting the data used for this study from the original replays.

C Architectures and Training Details

C.1 Base Model

For our base model ($\sim$ 103M parameters) we use a GPT-2 causal transformer architecture with 8 layers with 1024 hidden dim. Each attention layer has 8 heads, and the feedforward layers have a hidden dim of 4096.

Each image is resized to be of shape $128 \times 128 \times 3$, divided by 255 to put its values in $[0, 1]$, and is then fed into a convolutional encoder to map it to a 1024 dimensional vector.

The first layer of the conv net has kernels of shape 8×8, with a stride of 4, and a padding of 3 and maps to 16 channel dimension. This is followed by 4 lots of ConvNext (Liu et al., 2022) and downsampling blocks (kernel of shape 3×3, stride of 2, padding of 1, doubling the channel dimension). Finally, a LayerNorm (Ba et al., 2016) is applied to the output.

The transformer operates on sequences of 32 timesteps using learnt positional encodings. The output of the transformer is layernormed, and then fed into an MLP with a single hidden layer of 1024 dimensions with a GELU (Hendrycks & Gimpel, 2023) non-linearity.

For our optimiser we use AdamW (Loshchilov & Hutter, 2019) with a learning rate of $1e-4$ and a weight decay of $1e-4$. We use a batch size of 256 with a learning rate warmup period of 1000 updates and a gradient clipping value of 1. We train with the same image augmentations as used by (Baker et al., 2022), and filter out all no-op actions.

We used 8 32GB Nvidia V100 GPUs for approximately 4 days to train the base model.

C.2 Fine-Tuning of Base Model

We train for 1500 batches of size 128 with a learning rate of $1e-6$ with the same image augmentations and no-op filtering as for pre-training with 200 warmup steps.

We used 4 48GB Nvidia A6000 GPUs for less than 1 hour for fine-tuning.

C.3 Reward Models

Each trajectory is padded up to the maximum length of 100 (with black images) before being fed into the reward models.

For the `Random Encoder` model we randomly initialise a linear layer to randomly project the flattened values of the image to a 512 dimensional vector. This linear layer is not trained.

For the `Agent Encoder` model, we feed the trajectory into the fine-tuned agent and take the layernormed output of the transformer, corresponding to timesteps 0 up to 100, as 1024 dimensional embeddings. This is larger since it must capture the 32 context steps rather than just a single observation. The parameters of the fine-tuned agent are not trained.

For both models we then feed these vectors into an MLP with a GELU non-linearity and a hidden layer of 256 and and output dimension of 3. Each of the 100 3-dimensional vectors are concatenated together and then fed into another MLP with a GELU, 256 hidden dimension, and an output of 1.

To train the reward models we use a minibatch of size 2048, learning rate of $1e-4$, and an L_2 regularisation penalty of 0.1. We train all models for 200 epochs, except for the largest training set size of 1000 trajectories which we train for 50 epochs.

After training, we compute the minimum and maximum outputs of the reward model on the training set. These are then used to normalise the output of the reward model to lie within $[0, 1]$.

We used 4 48GB Nvidia A6000 GPUs for a few hours in total for training all reward models.

C.4 Alignment Training

Preference Fine-Tuning: after training the reward model on its dataset of M trajectories (which result in a dataset of up to $N = (M)(M-1)/2$ comparisions), we compute the reward for each of these M trajectories. We then sort them by magnitude, and take the top 20% of these as a smaller dataset to perform behaviour cloning on.

For this final step of BC, we use a learning rate of $1e-5$ with 1000 updates on minibatches of size 256. We only train the parameters of the MLP after the transformer layers.

We again used 4 48GB Nvidia A6000 GPUs for less than an hour for preference fine-tuning.

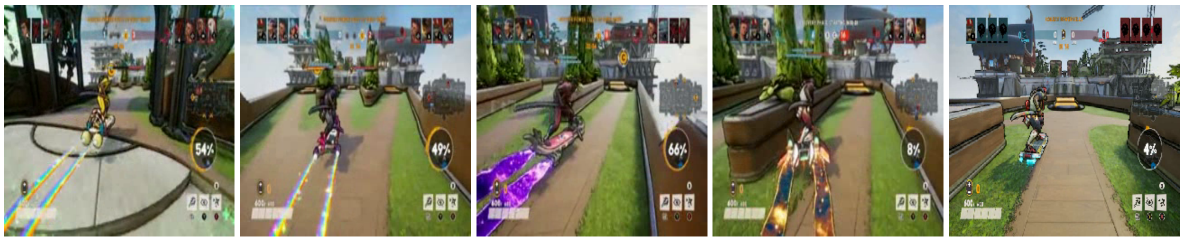
Reinforcement Learning: in our experiments we use an undiscounted REINFORCE loss on batches of 16 episodes of up to 100 timesteps. We use a learning rate of $1e-4$ and once again only train the parameters of the MLP after the transformer layers. If an error occurs during an episode's rollout, we simply drop the samples from that episode and subsequently use a smaller batch size for the update.

We used 16 Standard NV12 machines (each with 2 Nvidia Tesla M60 GPUs for game rendering) on Azure Batch, requiring one machine around one day per seed for agent alignment.

The total compute used for the entire research project therefore represents of the order of weeks of total compute time, with preliminary experiments not included of the order of days of compute.

D Discussion of Offline to Online Distribution Shift

As we mention in Section 3, the nature of using a real AAA video game is such that there is significant offline to online distribution shift between our offline training data and our online evaluation. One such source is due to character selection and customisable visual modifications, as shown below in Figure 9.



(a) (b) (c) (d) (e)

Figure 9: Screenshots of various characters with visual modifications contained within the general pre-training data. All are completely representative of the input provided to our agent (only 256×256 rather than 128×128 resolution). Figure (e) demonstrates the agent we use for online evaluation.

It is well-known that offline imitation learning often suffers from online distribution drift, particularly when learning from pixels in a continuous partially observable environment, due to accumulating errors (Ross et al., 2011). The visual distribution shift in our environment shown in Figure 9 only exacerbates these issues. As a result, imitation learning agents may not consistently perform imitated behaviours online even if the loss has converged on the offline data. This challenge is fundamental to offline learning (Ostrovski et al., 2021). This motivates the need to have some online fine-tuning (in our case from preferences) to refine the policy to consistently perform the desired behaviour online.

Figure 4 demonstrates that even after the supervised fine-tuning stage on a dataset of trajectories that always reach a jumppad and involve the same character, a significant proportion of trajectories ($\sim 10\%$) do not reach any jumppad. However, after online fine-tuning, we are able to increase the overall success rate to nearly 100% as shown in Figure 18.

E Ablation of Unsupervised Pre-Training and Model Scaling

E.1 Ablation of Unsupervised Pre-Training

We see from the results in Sections 3.1 and 3.2 that fine-tuning improves the task-specific performance of our pre-trained agent. However, to determine whether fine-tuning our base model is beneficial over simply training a model from scratch on our curated task-specific dataset, we also ablate the unsupervised pre-training stage. We train our agent from scratch for 20k updates, using the same procedure as used previously for fine-tuning (see Appendix C) until the loss appeared to converge. We subsequently evaluated the agent in the environment for 1000 episodes following the same procedure described in Section 3. The distribution of jumppads reached by the agent trained from scratch on the fine-tuning dataset is shown below (left) for comparison with the original pre-trained+fine-tuned agent (right) in Figure 10.

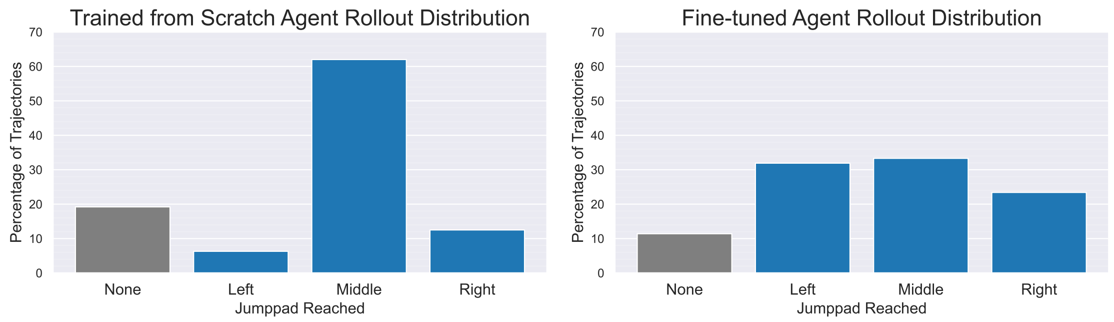


Figure 10: Distribution of jumppads reached by an imitation learning agent trained from scratch on the task-specific dataset used for fine-tuning the base model (left) compared to the base agent that was fine-tuned on this dataset.

We find that this agent has a greater failure rate (around 20% for the fine-tuned only agent fail to reach any jumppad compared to only around 10% for the pre-trained+fine-tuned agent) and a much narrower distribution of jumppads reached. This is surprising given the task-specific dataset consists of successful trajectories evenly split between the jumppads.

As additional anecdotal evidence for the benefit of unsupervised pre-training, we also noticed a fine-tuned agent miss a jumppad, hit the wall behind and then turn around to hit the jumppad. Since the fine-tuning dataset consists of only curated trajectories that directly hit the dataset, this behaviour was not present in the fine-tuning dataset, and agents trained from scratch on the fine-tuned dataset are found to continuously run into the wall if they miss the jumppad, as they have never seen that observation before. This behavior is shown on our project webpage at: `https://adamjelley.github.io/aligning-agents-like-llms/`.

This provides further anecdotal but intuitive evidence for the benefits of unsupervised pre-training.

E.2 Preliminary Model Scaling Analysis

To further justify the model size and investigate scaling properties of our transformer policy, we also trained smaller models of 4M and 25M parameters on our task-specific dataset, using the same procedure as above. The architecture of our 2 smaller models is identical to that of the base model described in Section C, except for:

- **∼4M:** 4 layers with 256 hidden dims and 4 heads for each attention layer.
- **∼25M:** 8 layers with 512 hidden dims and 8 heads for each attention layer.
- (**∼103M:** 8 layers with 1024 hidden dims and 8 heads for each attention layer.)

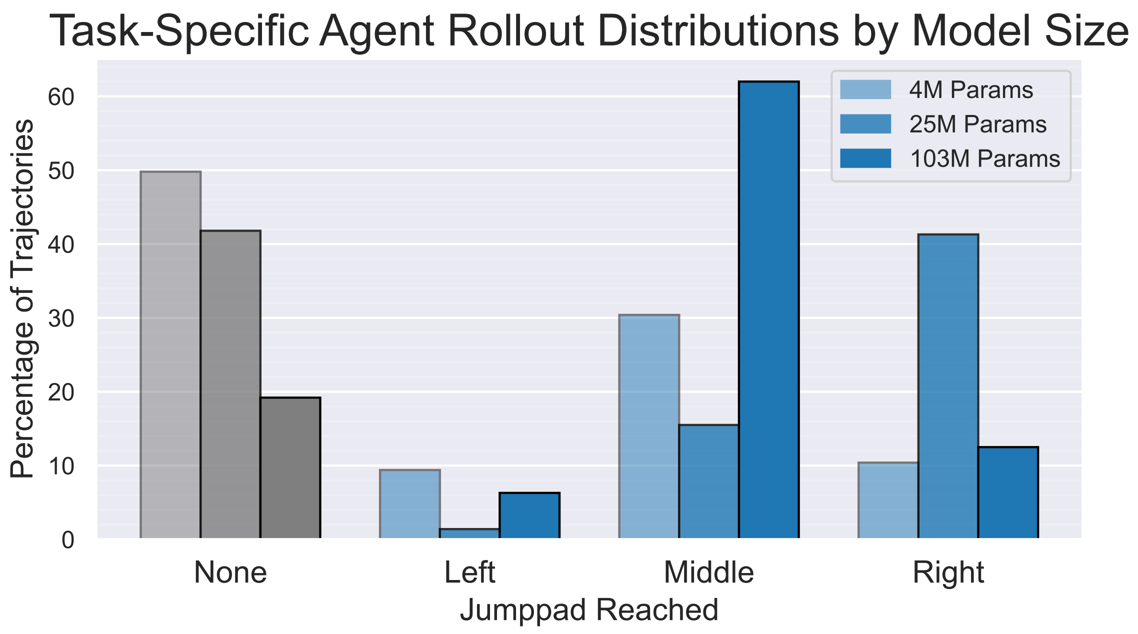


Figure 11: Distribution of jumppads reached by various size models (4M, 25M and 103M parameters) trained from scratch on the task-specific dataset used for fine-tuning the base model. Importantly, we see that task failure rates (the grey bars on the left) decrease as the number of parameters increases, even given the same size of dataset and number of training updates.

The jumppad distributions for these models are provided in Figure 11. We see that these smaller models have much greater failure rates that the larger 103M parameter models shown in Figure 10, demonstrating that larger models are beneficial for imitation learning from pixels even on this relatively small task-specific dataset of 300 trajectories. While larger models still may be beneficial (particularly with pre-training on our large unsupervised dataset described in Section 3), larger models would further increase the crucial inference cost and we find that 103M parameters is sufficient for further alignment, as demonstrated in Figure 10.

E.3 Alignment of Models Trained from Scratch by Size

To complete our ablation, we now investigate how pre-training and model size affects online alignment. To do so we followed the procedure in Appendix A to generated preferences as in Section 3. We then trained corresponding reward models and compare aligning these models trained from scratch to our pre-trained and fine-tuned model, as shown below in Figure 12.

Overall, the results are relatively high variance due to the small dataset these results are based on. However, bearing that in mind, we can see some general trends emerging nonetheless. We see that our pre-trained model aligns significantly better than the equivalent size model trained from scratch when aligned left, but worse when aligned right (due to the lack of diversity in the pre-trained+fine-tuned model as explained in Section 3.5.2). We also see that larger models generally align better (although the bias of the 25M parameter model towards going right makes this trend less clear) since the success rate generally increases more quickly during alignment for the larger models.

We also note that in this experiment we only considered using the reward models trained on 500k comparisons, in order to see whether it was possible to successfully align a model trained solely on our small task-specific dataset. As noted in the previous section, these models have significantly less diversity in their behaviour than the model first pre-trained and then fine-tuned. This makes it much more costly (in terms of human time) to generate sufficient labelled examples of the desired behaviour to properly train the corresponding reward model (for example, less than 10% of trajectories reach

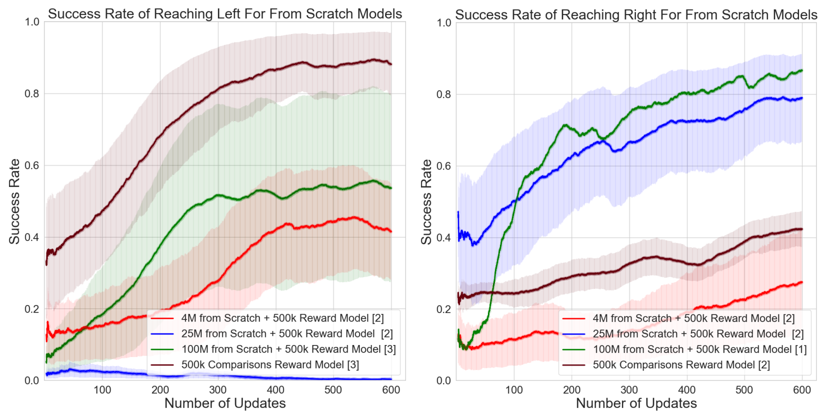


Figure 12: Left and right jumppad success rate during online alignment for models of different sizes trained from scratch using corresponding reward models trained on 500k comparisons.

the left jumppad for the model trained from scratch compared to over 30% for the pre-trained + fine-tuned model). This means that while these results show an upper bound in terms of reward model performance, we would expect the larger and pre-trained models to have less degradation in their alignment when using reward models trained on smaller quantities of preference data.

F Reward Model Performances by Jumppad

Test reward model performances against number of comparisons used for training by jumppad are shown below in Figure 13.

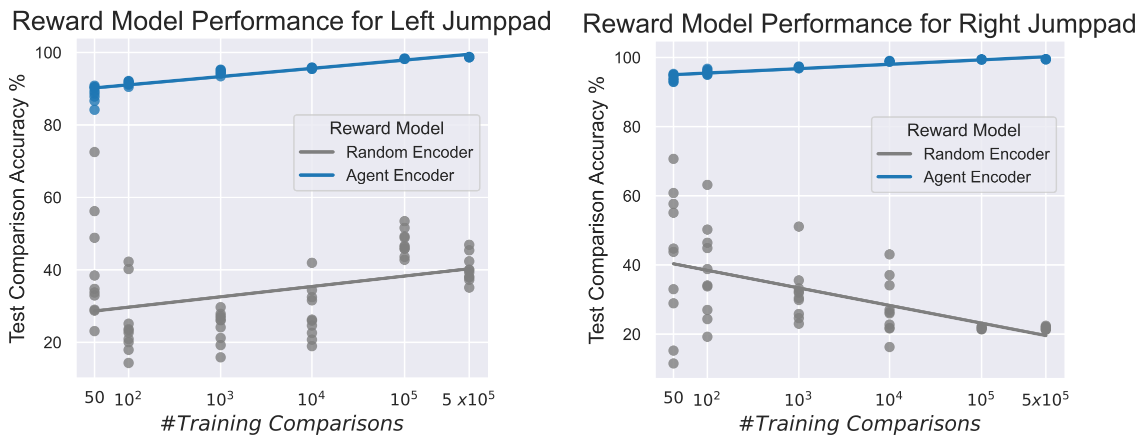


Figure 13: Test reward model performances against number of training comparisons by jumppad.

Both jumppads show the same trend for the agent encoder, reaching $\sim$ 90% performance for 100 comparisons and 100% performance by 500k comparisons. The random encoder is high variance, but generally increases in performance with training comparisons for the left jumppad, while decreasing in performance for the right jumppad. This surprising trend for the right jumppad could be because the reward model begins to overfit to the imbalanced trajectories discussed in Section 3.5.2 and shown in Figure 8. In other words, the reward model misinterprets the preferences and instead learns to prefer shorter trajectories that begin at the bottom left spawn point (from which 84% of the successful right-going trajectories arise) leading to the $\sim$ 20% accuracy that we see for higher numbers of comparisons. The agent encoder based reward model is better able to distinguish these trajectories and infer the true underlying preference that better fits the data.

G Alignment of Base Model for Comparison with Fine-Tuned Model

To understand how trajectory diversity affects alignment, we also consider aligning the base agent using the same reward models (trained on the fine-tuned agent). By plotting the successful trajectories that reach the left and right jumppads we see that the base agent (left) has a greater diversity than the fine-tuned agent (right). However, we note that the success rate for reaching the left and right jumppads is lower for the base model (as in Figures 3 and 4), so there are fewer trajectories.

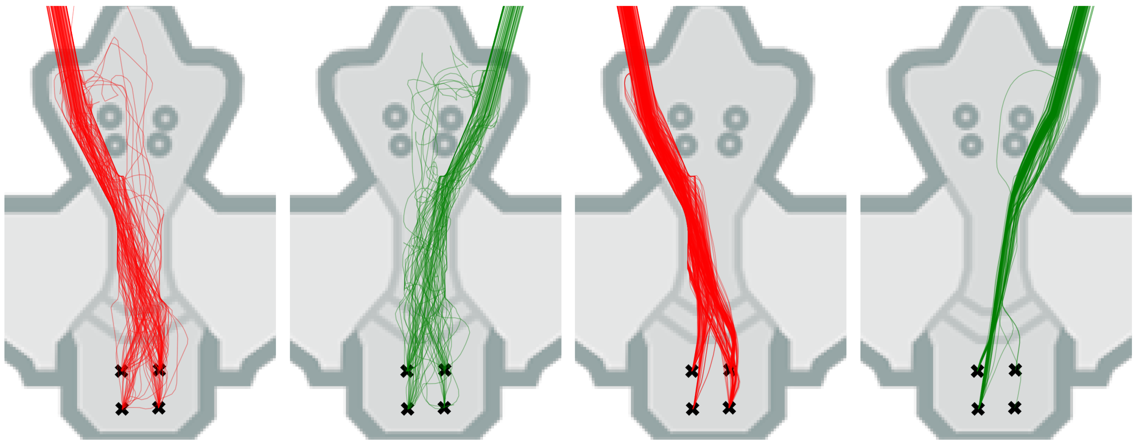


Figure 14: Base agent (left) and fine-tuned agent (right) trajectories for which the agents successfully reached the left (red) or right (green) jumppads. While fine-tuning provides a greater success rate, enabling more efficient preference labelling, it also reduces diversity of trajectories.

We now align the base agent to reach the left and right jumppads using the reward models trained on the fine-tuned agent, as shown below. We find that the agent can be aligned more symmetrically with both the left and the right jumppads, although alignment with the left jumppad still has a slightly better performance. However, in comparison to alignment of the fine-tuned agent (Figures 6 and 7), we see that the alignment is much slower, starting at a lower success rate, not increasing as quickly during training, and reaching a lower final success rate. As before, we see the same general trend that higher accuracy reward models (using more preference data) lead to better alignment.

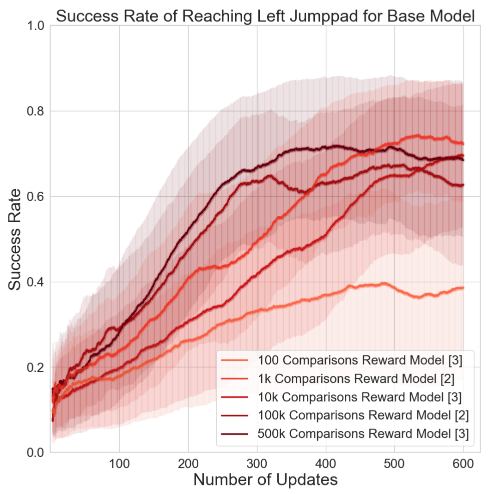


Figure 15: Left jumppad success rate for base agent using fine-tuned agent reward models.

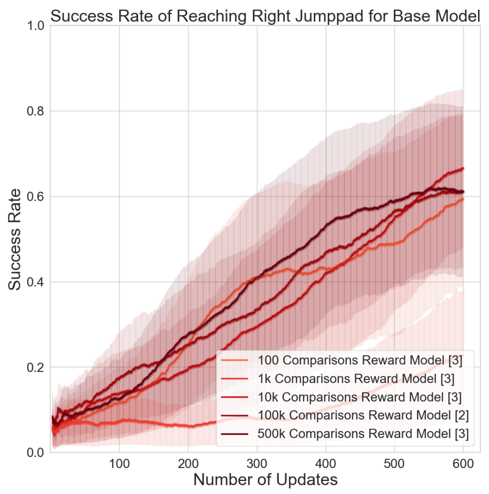


Figure 16: Right jumppad success rate for base agent using fine-tuned agent reward models

H Final Aligned Agent Jumppad Distributions

Jumppad distributions for our final agents aligned to go left and right using preference fine-tuning and online alignment with reward models trained on 500k preferences are shown below.

First we partially align our agents with preference fine-tuning using our (500k comparison) reward models, so that the behaviour distributions are closer to the desired behaviour distributions to reduce the alignment required online, as shown in Figure 17.

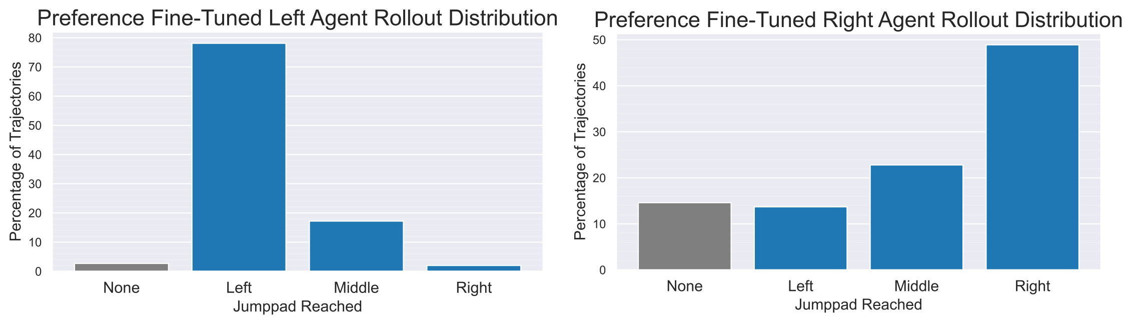


Figure 17: Left and right jumppad success rate for agents partially aligned with preference fine-tuning with reward models trained on 500k preferences.

We see that preference fine-tuning starts to align our agents towards the desired behaviour, but does not fully align our agents. Therefore we then perform online reinforcement learning using our (500k comparison) reward models until our agents are fully aligned.

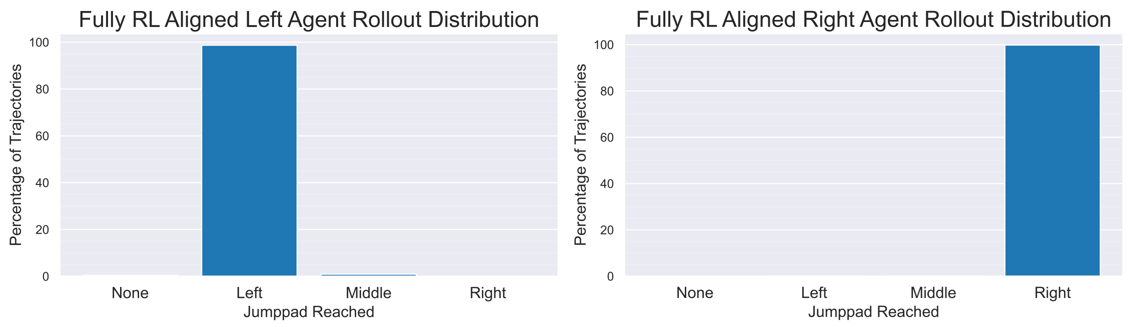


Figure 18: Left and right jumppad success rate for our fully aligned fine-tuned agents using preference fine-tuning and online reinforcement learning with reward models trained on 500k preferences.

Final evaluation shows that our agents have now been effectively fully aligned with our desired behaviour.

I Heatmap and Videos of Gradual Alignment of Agents

To help visualise the gradual alignment of our agent, we provide a heatmap of the agent trajectories at each stage of our alignment pipeline below in Figure 19.

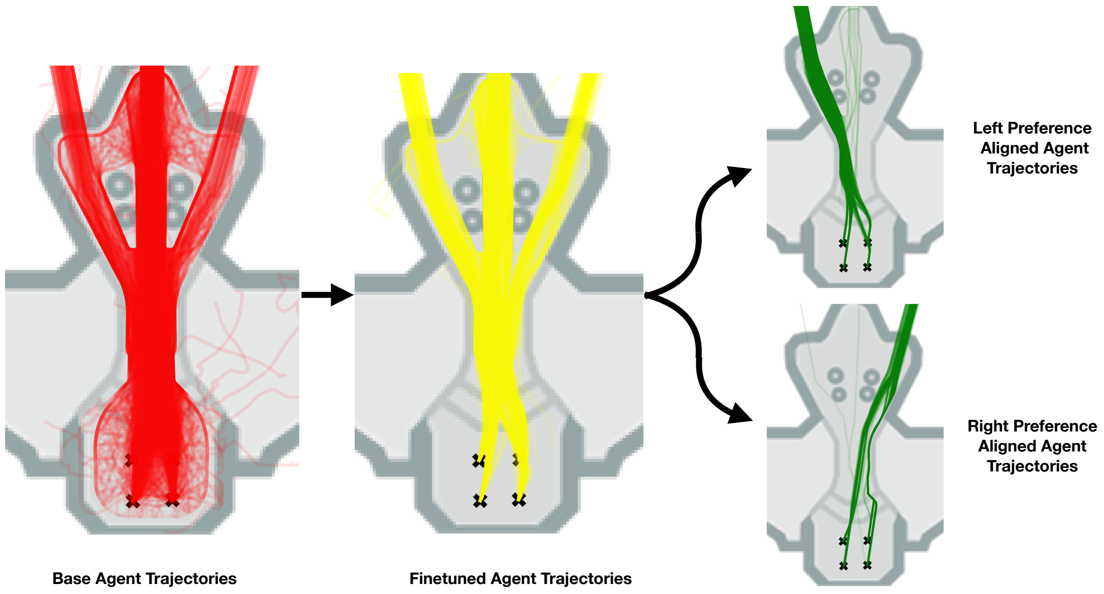


Figure 19: Heatmap of agent trajectories at each stage of our alignment pipeline. 1000 rollouts shown in each figure.

We see that the base agent trajectories are relatively diverse, capturing a variety of human behaviours, including those that do not reach a jumppad, or take a very indirect route to do so. The fine-tuned agent which has been refined on curated task-specific/demonstration trajectories shows more direct trajectories to the jumppads, but still does not incorporate any preference regarding the the jumppad which we would like our agent to reach. Finally, by training a reward model on these fine-tuned agent trajectories to capture our preferences, we are able to use that reward model for preference fine-tuning and subsequent online reinforcement learning to align our agent to reliably perform the desired behaviour, be it to reach the left or the right jumppad.

Videos of the behavior of our agent at each stage of this alignment pipeline are provided on the project webpage at: `https://adamjelley.github.io/aligning-agents-like-llms/`.

J Additional Related Work

J.1 Large Scale Imitation Learning Agents

Imitation learning and offline reinforcement learning (Levine et al., 2020) have recently gained popularity with the aim of demonstrating similar scaling laws to LLMs (Kaplan et al., 2020; Hernandez et al., 2021) to obtain more general agents. Decision Transformer (Chen et al., 2021a) proposed imitation learning with a transformer policy that can be conditioned on a desired return. Multi-Game Decison Transformers (Lee et al., 2022) extended this approach to learn multiple game policies with a single model. GATO (Reed et al., 2022) and RoboCat (Bousmalis et al., 2023) demonstrated large scale multi-task imitation learning could enable a single model to learn and improve on hundreds of diverse tasks. RT1 (Brohan et al., 2022) shows similar scaling potential for transformers on robotics tasks, while RT2 (Brohan et al., 2023) integrated vision-language models to help with zero-shot generalisation to new tasks. In addition to transformer policies, Kumar et al. (2022) show promising scaling for offline Q-learning with ResNet architectures (He et al., 2015), while Pearce et al. (2023) demonstrate that diffusion models can also scale to capture complex human action distributions. More recently, SIMA (SIMA-Team et al., 2024) is a general agent with language input and keyboard output which shows promising generalization to new video games, and further demonstrates the potential for scaling imitation learning to large, diverse datasets.

J.2 Inverse Reinforcement Learning

As an alternative to the behavior cloning (Pomerleau, 1991) approach to imitation learning that we take in this work (along with most other recent large scale imitation learning agents, such as those discussed in Section J.1), another approach is to use inverse reinforcement learning (Ng & Russell, 2000; Arora & Doshi, 2021) in which the reward function of a demonstrator is inferred from demonstrations. An agent can then be trained with reinforcement learning using this reward function to imitate the demonstrator (Abbeel & Ng, 2004), or an equivalent policy can be learned from the demonstrations directly (Ho & Ermon, 2016). However, these approaches assume that the demonstrator is an expert that only attempts to perform the desired behaviour. We instead consider the case where the offline data is not optimal, consisting of general multi-modal gameplay data, in which we would like our agent to imitate a particular mode of this behavior distribution. While we may be able to provide some demonstrations, we assume that we do not have sufficient clean demonstrations for robust inverse reinforcement learning. Therefore, in this work we instead aim to adapt and refine the behavior of an agent trained on more general data according to our preferences.

J.3 Fine-tuning with Reinforcement Learning

The use of imitation learning as pre-training for reinforcement learning was first investigated to improve the sample efficiency of deep RL algorithms by reducing the exploration space (Hester et al., 2017; Vecerik et al., 2018). AlphaGo (Silver et al., 2016) and subsequently AlphaStar (Vinyals et al., 2019) demonstrated that imitation learning on human data could provide a strong behavior prior for environments like StarCraft where reinforcement learning from scratch is infeasible. VPT (Baker et al., 2022) extended this paradigm to web-scale data by performing imitation learning on 70k hours of Minecraft videos labelled by an inverse dynamics model, before fine-tuning with reinforcement learning on task-specific rewards. While these works demonstrate the potential of fine-tuning large imitation models, they use hard-coded reward functions to maximise agent performance rather than align an agent's behavior with subjective preferences.

J.4 Reinforcement Learning from Human Feedback for Agents

Training agents with human preferences has a long history, as reviewed by Wirth et al. (2017) and Zhang et al. (2021) (including notably Bennett et al. (2009) and Knox & Stone (2008)), leading to the popular modern formulation of RLHF for deep learning proposed by Christiano et al. (2017)

that we use in our work (see Section 3.4). This idea was extended by Ibarz et al. (2018) to include imitation learning as pre-training to improve the efficiency of preference learning. PEBBLE (Lee et al., 2021) instead utilises unsupervised pre-training to increase the diversity of initial behaviors, and incorporates reward relabelling to enable off-policy learning for greater feedback efficiency. Preference Transformer (Kim et al., 2023) investigates the use of a transformer as a reward model, and finds that bidrectional self-attention can better capture temporal dependencies in human preferences. Abramson et al. (2022) and Milani et al. (2023) are closest to our work in demonstrating the potential of RLHF for improving the performance of large imitation agents in 3D simulated worlds, but do not analyze the full LLM training pipeline.

J.5 Large Language Models

Radford et al. (2018) demonstrated that pre-training with an unsupervised generative task (next token prediction) on a large diverse corpus of text, followed by fine-tuning on a specific task provided benefits compared fine-tuning alone. Stiennon et al. (2020) (following Ziegler et al. (2020)) then popularised the use of RLHF to further fine-tune these models to align their responses with human preferences. This procedure led to InstructGPT (Ouyang et al., 2022), Chat-GPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023) as well as open-source models, such as Claude (Bai et al., 2022), Llama 2 and 3 (Touvron et al., 2023) and Gemma (Google Deepmind, 2024). The success of these models suggests that it is worth investigating whether their training procedure can be transferred to other domains. While components of our procedure have been applied both individually and in combinations to agents, our work is the first to investigate the benefits of applying the full LLM training procedure to agents trained end to end from visual inputs to a unified action space.

K Limitations and Further Work

One limitation of our work is that we assume access to a large amount of human data on the game of interest to pre-train a base model. In practice however, when designing a game, large amounts of human data may not be available, even for existing games. A potential solution may be to use a 'gaming foundation model' trained on similar games (with pixel input and action controller output), as investigated by other works (Lee et al., 2022; Reed et al., 2022; SIMA-Team et al., 2024).

Another limitation of our approach is that we utilize synthetic preferences. While this may be possible for many use cases, real human feedback from a game designer will be required in general. Given the time expense of providing feedback, especially for agents on modern console games which must often be run in real-time in order to be rendered, this process could be costly. Fortunately, supervised fine-tuning on relevant behaviours can enable more efficient preference labelling. Additionally, knowledge transfer from LLMs could again be relevant here. Recent work on providing human feedback for LLMs has used a hybrid form of feedback that combines preference and evaluative feedback, in which responses are grouped into a batch of size N and simultaneously compared on a preference scale of size P, which enables larger numbers of comparisons to be extracted from a given number of responses, improving the time efficiency of providing feedback. For example, InstructGPT (Ouyang et al., 2022) uses $4 \leq N \leq 9$, $P = 7$ while Llama2 (Touvron et al., 2023) uses $N = 2$, $P = 4$. As N and P increase, more information can be extracted from the provided human feedback. For example, with $N = 5$, $P = 5$ and assuming no category duplication (no trajectories are considered equal), $\binom{5}{2} = 10$ comparisons can be extracted, which is equivalent to 2 bits of information per trajectory watched, compared to just 0.5 bits per trajectory for default pairwise comparisons with $N = 2$, $P = 2$. This results in a $4\times$ improvement in feedback efficiency for human labellers, at the cost of potential label noise due to the additional mental overhead required. Similar strategies could be applied to providing feedback to agents to make the process more time efficient.

A final limitation is that running agents online in modern console games can be challenging, and may be prohibitively expensive for more complex behaviors. Fortunately, developments in RLHF for language models can also be utilised here. We demonstrated in Section 3 that an intial step of preference fine-tuning could improve the efficiency of alignment, and explained that this corresponds to Reinforced Self-Training (ReST) (Gulcehre et al., 2023) with a single iteration. However, the full procedure using multiple iterations could be used to further improve the sample efficiency of aligning with preferences while maintaining the benefits of online exploration. Other recent work has investigated aligning models completely offline. Direct Preference Optimisation (Rafailov et al., 2023) provides an approach for optimising a model to align with preferences without the need for a reward model or online training, while (Hu et al., 2023) uses offline reinforcement learning with pre-generated samples and rewards. Additionally, efficient fine-tuning strategies such as LoRA (Hu et al., 2021) could be used to reduce the hardware requirements for game designers to fine-tune a large base model. These new approaches provide promising directions for further work in the context of agents to improve the efficiency or potentially even remove the need for online training completely. However, the extent to which active online learning is required for true generality is still an open question.