ADAPTING A WORLD MODEL FOR TRAJECTORY FOLLOWING IN A 3D GAME

Marko Tot[1,2*†] **Shu Ishida**[1,3*†] **Abdelhak Lemkhenter**[1]

David Bignell[1] **Pallavi Choudhury**[1] **Chris Lovett**[1]

Luis França[1] **Matheus R. F. Mendonça**[1] **Tarun Gupta**[†] **Darren Gehring**[1]

Sam Devlin[1] **Sergio Valcarcel Macua**[1] **Raluca Stevenson**[1]

[1]**Microsoft Research** [2]**Queen Mary University of London** [3]**University of Oxford**

ABSTRACT

Imitation learning is a powerful tool for training agents by leveraging expert knowledge, and being able to replicate a given trajectory is an integral part of it. In complex environments, like modern 3D video games, distribution shift and stochasticity necessitate robust approaches beyond simple action replay. In this study, we apply Inverse Dynamics Models (IDM) with different encoders and policy heads to trajectory following in a modern 3D video game – Bleeding Edge. Additionally, we investigate several future alignment strategies that address the distribution shift caused by the aleatoric uncertainty and imperfections of the agent. We measure both the trajectory deviation distance and the first significant deviation point between the reference and the agent's trajectory and show that the optimal configuration depends on the chosen setting. Our results show that in a diverse data setting, a GPT-style policy head with an encoder trained from scratch performs the best, DINOv2 encoder with the GPT-style policy head gives the best results in the low data regime, and both GPT-style and MLP-style policy heads had comparable results when pre-trained on a diverse setting and fine-tuned for a specific behaviour setting.

1 INTRODUCTION

Using video games as a testbed for game-playing agents has been a thoroughly studied area. Although imitation learning and reinforcement learning have been applied, most of these algorithms (Vinyals et al., 2019a; Berner et al., 2019; Wurman et al., 2022) focused on superhuman behaviour, rather than matching human play style. Research on human-like play primarily leverages imitation learning, where the most popular techniques revolve around learning from demonstration (Abbeel & Ng, 2004; Ho & Ermon, 2016) and learning from observations (Torabi et al., 2018a; Yang et al., 2019).

In this work, we use learning from demonstrations to replicate a recorded trajectory in a complex 3D video game. In simple environments, trajectory replication can often be achieved by directly replaying recorded actions. However, in stochastic settings, naive action playback fails, as small variations in state transitions can lead to significant deviations from the intended trajectory.

To address this challenge, we adapt a pre-trained World Model to construct an Inverse Dynamics Model (IDM) (Lamb et al., 2023). We evaluate the effectiveness of world model embeddings compared to alternative encoders and explore future alignment strategies to improve the replication of

[*]Equal contribution

[†]This work was conducted while at Microsoft Research

recorded trajectories in the video game Bleeding Edge. Our evaluation involves two types of policy heads – an autoregressive transformer (Radford et al., 2019) and a feed-forward network – paired with three encoder types: a pre-trained game-specific world model, a general pre-trained encoder, and a ConvNeXt trained from scratch. This results in six distinct model configurations.

We evaluate these six model variants across three experimental settings: 1) General - trained on a large corpus of general gameplay trajectories, and evaluated on held-out trajectories, 2) Specific - trained on a small set of similar trajectories that exhibit the same behaviour, and evaluated on the same class, and 3) Fine-tuned - where we pre-train the model using 1) and then fine-tune and evaluate it using 2).

In summary, our contributions are as follows:

- Adapting a pre-trained world model for imitation learning in a downstream task of trajectory following.
- Conducting an empirical analysis of different model configurations across General, Specific and Fine-tuned settings in a complex 3D video game.
- Investigating the impact of design choices, such as using single observation vs sequence of observations as the model inputs, and the inclusion action inputs.
- Exploring different future conditioning strategies to mitigate distribution shifts and improve long-term trajectory alignment.

2 Related Work

World Models World Models (Ha & Schmidhuber, 2018) have shown strong capabilities in simulating environments. They have been successfully applied to games such as Doom (Valevski et al., 2024), Atari (Micheli et al., 2023) and Counter Strike (Alonso et al., 2024), allowing users to interact with the game without reliance on an underlying game engine. Recent studies have shown that world models follow similar scaling laws as Large Language Models (Pearce et al., 2024), exhibit zero-shot generalisation to unseen tasks (Xu et al., 2022), and can scale to multi-game environments (Bruce et al., 2024).

Imitation Learning Imitation learning enables agents to learn tasks by observing expert demonstrations rather than relying on reward signals or direct exploration, as in reinforcement learning. The goal is to replicate observed behaviours by learning policies that map states to actions. One of the most widely used approaches is Behavioural Cloning (BC) (Torabi et al., 2018b), which trains models on offline datasets to mimic human behaviour. BC has been applied in domains such as autonomous driving (Pomerleau, 1991), robotics (Florence et al., 2022) as well as game-playing agents in video games (Kanervisto et al., 2020; Vinyals et al., 2019b). While much of the research in imitation learning for games focuses on optimising policies for high performance (Vinyals et al., 2019b; Ross et al., 2010; Ho & Ermon, 2016; Pearce & Zhu, 2022), some works have explored capturing diverse played behaviours and playstyles. Ferguson et al. (2022) investigated the use of Dynamic Time Warping (Mü, 2007) to imitate a given playstyle, while Pearce et al. (2023) used Diffusion Models to capture multi-modal behavioural patterns. These studies highlight the potential of imitation learning to generate human-like gameplay beyond optimal strategies.

Inverse Dynamics Models Inverse Dynamics Model (IDM) is commonly used to condition agents by predicting the action required to transition from one state to another. Paster et al. (2021) applied an IDM to task condition the agent in a visual domain to predict a sequence of actions. Yang et al. (2019) used an IDM, as a tool, to minimise the disagreement between the expert and the agent during training. Pavse et al. (2019) combined an IDM with reinforcement learning to match the expert's trajectory. They use an inverse dynamics model to infer the action that should be taken to traverse from the learner's to the expert's trajectory in robotic control domains. While our approach shares similarities with Pavse et al. (2019), there are key distinctions. Our method focuses on directly matching reference trajectories, whereas their objective was to generate high-scoring trajectories when conditioned on the expert behaviour. Additionally, we train our IDM on offline datasets, avoiding the need for an RL feedback loop to avoid the time cost of querying the environment.

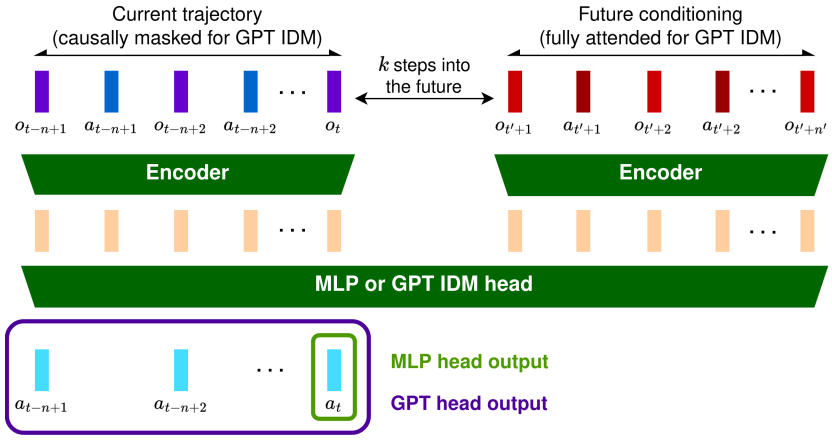


Figure 1: A high-level overview of the IDM model. We encode two distinct trajectories, the current trajectory of the agent, and the future conditioning. The resulting encodings are then passed into an IDM head to select which action should be performed.

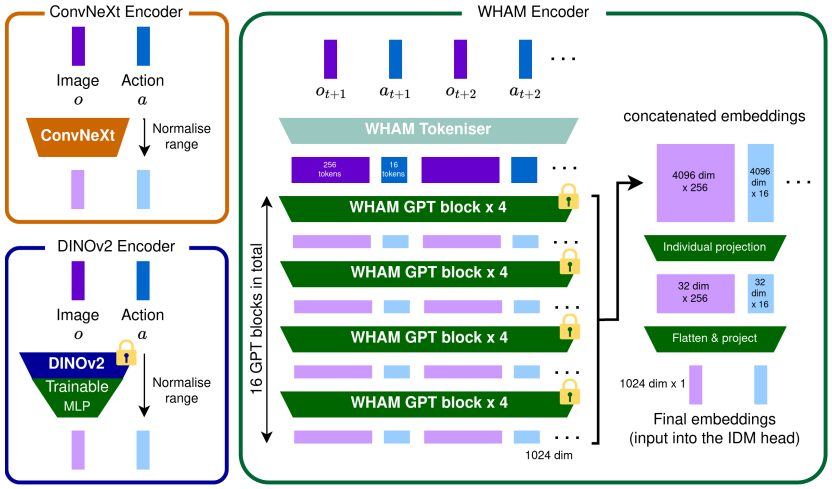


Figure 2: We evaluate three different encoders. A trained from scratch ConvNeXt encoder, a general pre-trained encoder DINOv2, and a game-specific pre-trained World and Human Action Model.

3 METHOD

A conventional IDM encodes the current observation and next observation to predict the action that has been taken. IDM-K generalises this approach, by shifting the future conditioning K steps ahead. Rather than relying on a single observation pair, we extend the notion of IDM-K by conditioning on past and future trajectory sequences - comprising of both observations and actions - to improve the temporal consistency and long-term dependency. This is particularly important in partially observable environments, such as video games, where single-frame conditioning can be insufficient for accurate decision-making.

3.1 MODELS

As shown in Figure 1, our generalised IDM-K model processes a past trajectory sequence up to time step t, and a future trajectory sequence from time step $t' + k$ onward. Given these inputs, the model predicts the action taken at t. Both observation and action sequences can be encoded independently or jointly, proving flexibility in representing past and future context. The encoded representations are then passed through a policy head to predict the action.

Visual Encoder We evaluate three different encoders (shown in Figure 2):

1. ConvNeXt (Liu et al., 2022): A convolutional network trained from scratch, combined with a normalised action vector.
2. DINOv2 (Oquab et al., 2024): A pre-trained general-purpose image encoder with a fine-tunable MLP head combined with a normalised action vector.
3. World and Human Action Model (WHAM) (Kanervisto et al., 2025): A pre-trained game-specific encoder optimised for auto-regressive next-token prediction, processing sequences of interleaved image and action tokens as context.

The WHAM encoder presents a unique challenge due to its tokenised representation. In its original form, a single observation and corresponding action are represented by 256 and 16 tokens, respectively. This results in a sequence length 272 greater than the actual trajectory length. Preliminary experiments showed that training the IDM-K to predict action vectors every 272 steps does not provide a sufficiently dense loss to the model. To address this, we project the sequential embeddings down to compact representations – one for the observation input and another for the action input before passing them through a secondary projection layer to obtain the final embeddings.

Decoder Since the model processes sequential data, we employ a transformer-based decoder for action prediction. Specifically, We use a GPT model (Radford et al., 2019) as the IDM head to predict the actions taken to arrive at the future trajectory. With a GPT head, it is possible to predict multiple actions corresponding to different time steps at which the past information is cut off. These additional losses on the sequential action outputs help robustify the IDM policy to varying lengths of past trajectory input. We compare the GPT head against a Multi-Layer Perceptron (MLP) IDM head. The MLP head flattens and concatenates the input encodings, offering a simpler alternative to the transformer-based approach.

We combine the decoders (MLP, GPT), with the encoders (ConvNeXt, DINOv2, and WHAM) resulting in six IDM-K variants to be evaluated.

3.2 Future selection strategies

Selecting an appropriate future conditioning strategy for IDM-K is crucial due to two primary challenges: (1) agent imperfections, and (2) environmental stochasticity.

- Agent Imperfections: The model's action predictions are not always accurate, leading to deviations from the reference trajectory. As these errors accumulate, the agent may drift too far from the expected path. The distance between the current position and the future conditioning can become too large, putting the agent out of the training distribution.
- Environmental Stochasticity: Certain game elements, such as randomly spawning firewalls, moving platforms, and health packs can alter the environment unpredictably, causing a further distribution shift.

To mitigate these issues, we explore four future conditioning strategies:

Static future conditioning In this approach, the future start timestep t'_{future} is determined solely by the current timestep t_{current}, independent of the agent's position. Given the number of frames to be skipped K, the future start timestep is computed as:

$$t'_{\text{future}} = t_{\text{current}} + K. \tag{1}$$

This method does not account for spatial deviations, making it susceptible to distribution drift when the agent strays from the intended trajectory.

Closest future conditioning This method selects the closest point in the reference trajectory based on the agent's current position, minimising spatial deviation. The future start timestep is given as:

$$t'_{\text{future}} = \arg\min_{t'} \|\tau(t') - \hat{\boldsymbol{x}}_t\| + K, \tag{2}$$

where $\tau(t)$ represents the (x, y, z) position in the reference trajectory at timestep t, $\hat{\boldsymbol{x}}_t$ represents the agent's current (x, y, z) position, and K represents the number of skipped frames.

While this strategy maintains spatial alignment, it introduces potential pitfalls in scenarios involving loops, or long stationary phases, where the agent may become trapped in an infinite cycle.

Radius future conditioning This method dynamically updates the future conditioning timestep based on the proximity to the reference trajectory. If the agent is within a predefined radius r, the future start timestep advances by one step, otherwise it remains unchanged.

$$t'_{\text{future}} = \begin{cases} t'_{\text{future}} + 1, & \text{if } \|\tau(t'_{\text{future}}) - \hat{\boldsymbol{x}}_t\| \leq r, \\ t'_{\text{future}}, & \text{otherwise,} \end{cases} \tag{3}$$

where $\tau(t)$ represents the (x, y, z) position in the reference trajectory at timestep t, $\hat{\boldsymbol{x}}_t$ represents the agent's current (x, y, z) position, and r represents the radius.

Inner-Outer future conditioning This method introduces two thresholds: an inner radius r_{in} and an outer radius r_{out}. The future timestep is updated based on the distance between the agent's position and the position of the current future conditioning frame.

- Smaller than r_{in} - The future timestep advances until it leaves the inner radius.
- Between r_{in} and r_{out} - The future advances by 1 step, same as the Radius strategy.
- Larger than r_{out} - The future timestep remains unchanged.

A full description of the conditioning strategies can be seen in Appendix A.3.

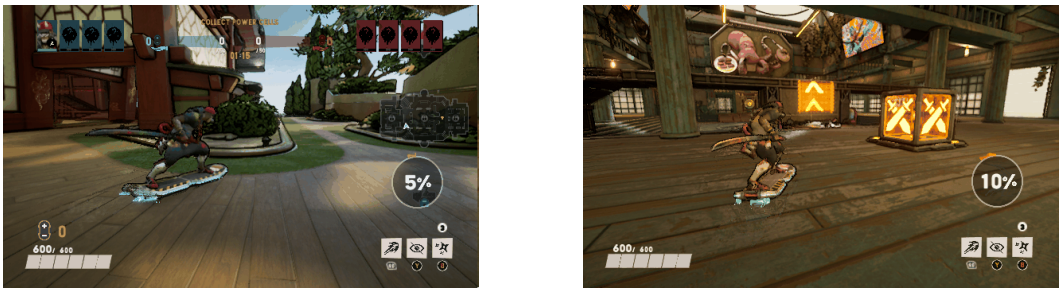
4 Experiments

The following section describes the environment used for the experiments, outlines the training and evaluation setup and provides and analyses the empirical results of the different models.

4.1 Environment

We evaluate our approach in Bleeding Edge[1], a third-person multiplayer game featuring various playable characters with unique abilities. This environment provides a challenging testbed for trajectory-following agents due to its dynamic gameplay and stochastic elements.

Our experiments focus on two maps: SkyGarden and the tutorial map Dojo (Figure 3). The agent's observations contain both the visual signal, representing the agent's camera viewport and the symbolic modality including the telemetry data for the agent. The action space is represented by the input of an Xbox controller, having a discrete set of 12 buttons and 2 continuous sticks, for movement and camera rotation, each including the x and y axes.



(a) Sky Garden (b) Dojo

Figure 3: Example images of the Sky Garden and Dojo maps used for training and evaluation.

[1]Official game website: https://www.bleedingedge.com/en

The dataset contains $71,940$ trajectories of recorded human gameplay on the Sky Garden map, from 8788 matches recorded between 02-09-2020 and 19-10-2022 (Jelley et al., 2024). This dataset represents approximately 1.12 years of human gameplay, and contains visual and telemetry data, as well as the actions the players took during the game.

4.2 SETUP

To unify the modality of the action space from the dataset, we normalise the continuous dimensions (stick inputs) of the continuous action space into a [-1, 1] range and then discretise it into 11 bins. All of the proposed models were trained for 200 epochs, each epoch with 1,000 updates with a batch size of 64. The complete list of training hyperparameters is provided in Appendix A.2.1.

Evaluation We evaluate model performance on 8 held-out trajectories, covering diverse tasks:

- Jumppad (3 trajectories): The agent must select the correct path at a crossroads
- Benchmark (3 trajectories): Involves complex turns throughout different map areas.
- Dojo (2 trajectories): Navigation through a tutorial environment.

These test trajectories vary in complexity and domain. Some originate from Sky Garden (the primary testing environment), while others come from Dojo, introducing an additional domain shift. The trajectories can be seen at https://adaptingworldmodels.github.io/.

Each agent is evaluated on 10 rollout seeds per trajectory. We assess the performance on two metrics:

- AUC (Area Under the Curve): Measures trajectory similarity via Dynamic Time Warping (DTW) (Mü, 2007) at varying radii.
- Future Index Ratio (FI): Captures the proportion of the trajectory followed before the first significant deviation from the reference path.

A detailed formulation for the AUC metric is provided in Appendix A.4. Unless otherwise stated, all reported results reflect the median score across 10 rollouts per model, per trajectory.

4.3 RESULTS

This section contains the empirical evaluation of the models and provides insight into the encoder and IDM head design choices. We split the section into two parts: (1) Evaluation in the General setting, and (2) Evaluation of Specific and Fine-tuning settings.

4.3.1 GENERAL SETTING

Table 1 summarises the performance of different architectures trained on a large, diverse dataset and evaluated zero-shot on unseen trajectories. Across all evaluated architectures, ConvNeXt outperforms the other encoders, regardless of whether paired with the MLP or GPT IDM head. The FI performance of the models is available in Appendix A.2. While the AUC provides a quantitative comparison between models, it does not capture if trajectory-following was successful.[2]

To provide a more intuitive understanding of model performance, Figure 4 visualises the Benchmark 1 trajectory, overlaying the agent's paths to the reference trajectory. Figure 4 reveals distinct performance differences among the models. ConvNeXt consistently follows the expected trajectory, with deviations only in a single rollout. DINOv2 exhibits occasional successful runs, recovering from minor deviations, while WHAM fails to follow the trajectory, displaying mostly arbitrary movement. These qualitative observations align will the quantitative AUC results presented in Table 1.

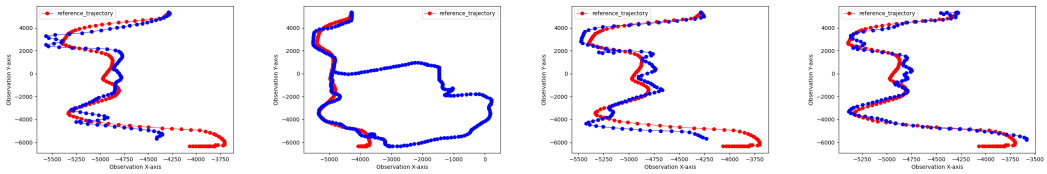
Our results suggest that training an encoder specifically for trajectory following is advantageous. Even the relatively simple ConvNeXt architecture produces more effective embeddings than large-scale pre-trained models like DINOv2 and WHAM, which were not explicitly trained for this task.

The performance difference between ConvNeXt, DINOv2 and WHAM likely comes from differences in their embedding strategies. ConvNeXt and DINOv2 encode observations independently,

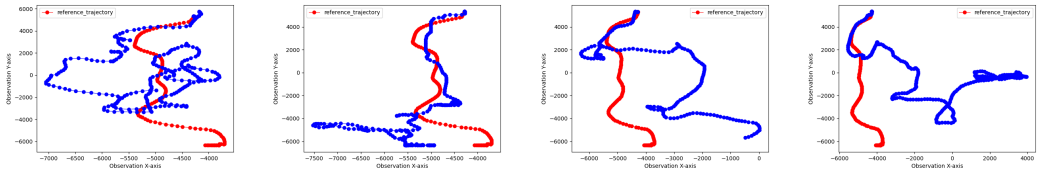
[2]Sample videos of the ConvNeXt agent are available at https://adaptingworldmodels.github.io/

Table 1: AUC for the general setting evaluation on 8 heldout trajectories. Higher AUC is better.

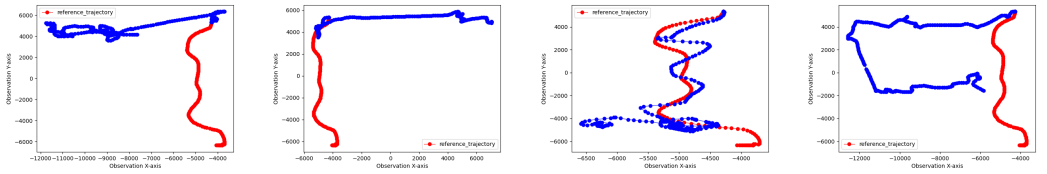
	MLP			GPT		
Trajectory	ConvNeXt	DINOv2	WHAM	ConvNeXt	DINOv2	WHAM
Jumppad left	0.99	0.92	0.84	**0.99**	**0.99**	0.87
Jumppad right	0.89	0.88	0.91	**0.99**	**0.99**	0.92
Jumppad mid	0.99	0.81	0.83	**0.99**	**0.99**	0.90
Benchmark 0	0.33	0.33	0.17	**0.64**	0.59	0.29
Benchmark 1	0.81	0.50	0.67	**0.98**	0.88	0.57
Benchmark 2	0.73	0.56	0.26	**0.95**	0.89	0.50
Dojo Ramp	0.65	0.40	0.67	**0.69**	**0.69**	0.68
Dojo Gong	0.54	0.42	0.39	**0.66**	0.65	0.64
Mean	0.74	0.60	0.59	**0.86**	0.84	0.67



(a) Samples of ConvNeXt-GPT rollouts on the Benchmark 1 trajectory.



(b) Samples of DINOv2-GPT rollouts on the Benchmark 1 trajectory.



(c) Samples of WHAM-GPT rollouts on the Benchmark 1 trajectory.

Figure 4: Sampled rollouts from Benchmark 1. The red line shows the reference trajectory, while the blue line shows the agent's path. x and y axes represent the coordinates of the agent.

with actions processed separately, while WHAM uses a causal transformer to jointly encode both modalities. This early-stage entanglement of observations and actions may limit the model's ability to generalise effectively to trajectory-following tasks.

While WHAM-GPT achieves lower training loss compared to ConvNeXt and DINOv2, this advantage does not translate to better evaluation performance, where WHAM underperforms relative to other variants. This discrepancy likely originates from the difference between training and evaluation conditions. During training, past and future trajectories are perfectly aligned both spatially and temporally, as they originate from the same recorded trajectory. However, at evaluation time, past trajectories are generated dynamically by the agent, while future trajectories remain fixed from recording. The training curves for different models can be seen in Appendix A.2.2.

General Setting Ablations To understand the benefit of having different input modalities, we conduct ablations by selectively removing observations or actions. Table 2 presents the performance comparison for the ConvNeXt-GPT model across these variations.

Results indicate that visual inputs are essential for accurate trajectory following, while including action inputs provides only marginal gains. In simple scenarios (e.g. Jumppad trajectories), action inputs alone suffice. The observation-only model almost matches the performance of the full model. The evaluation shows that having the visual modality is crucial while adding the action modality gives only a slight overall improvement. Full results per trajectory can be seen in Appendix A.5.

Table 2: AUC and the FI of different inputs to the model - Mean results across all trajectories

	Observations Only		Actions Only		Full Model	
	AUC	FI	AUC	FI	AUC	FI
Mean	0.84	0.70	0.69	0.47	**0.86**	**0.73**

We also investigate the impact of sequence length on model performance. Table 3 reports results for different past and future sequence lengths, alongside a BC agent as a baseline (which lacks future conditioning by design). Full results per trajectory can be seen in Appendix A.5.

The results indicate that the BC agent struggles to complete any of the trajectories consistently; the only trajectory it performed well on was Jumppad mid, where the behaviour is to only move forward. Other models performed on par with each other, with giving full 10 observation-action frames for both the past and the future being slightly better. Full results per trajectory are in Appendix A.5.

Table 3: AUC and the FI of different future and past lengths - Mean results across all trajectories

	BC		1P-1F		10P-1F		1P-10F		Full model	
	AUC	FI	AUC	FI	AUC	FI	AUC	FI	AUC	FI
Mean	0.64	0.45	0.84	0.69	0.85	0.69	0.79	0.64	**0.86**	**0.73**

We present the results for the future selection strategies in Table 4. It is important to note we do not measure FI in this setting, as the radius values directly influence the allowed spatial distance between the agent and the future trajectory, making a fair assessment in this regard impractical. Among the four selection strategies *Closest* slightly outperforms the others, achieving the AUC of 0.877. While *Closest* was the best-performing strategy on the benchmark, we chose the *Radius* strategy for our models for two key reasons. First, the agent struggled to follow the Dojo trajectories, showing arbitrary movement regardless of the strategy. Excluding the Dojo trajectories, the *Radius* strategy yields a slightly better mean AUC than *Closest*. Second, the *Closest* strategy encounters theoretical issues when dealing with trajectories containing loops or extended sequences of no-ops - problems that were not present in the evaluation set, but may arise in other scenarios.

Table 4: Different future selection strategies. Results show the median AUC value across all trajectories. The radius R for Radius and Inner-Outer strategies has been determined through a parameter sweep for each trajectory.

	Static	Closest	Radius	Inner-Outer
Mean AUC	0.84	**0.88**	0.86	0.85

4.3.2 Specific and Fine-tuning

We trained models on 30 trajectories that exhibit near-identical behaviour (variations of Dojo Ramp). Table 5 presents the results. The evaluation procedure matches the General setting, the only difference being that the WHAM encoder was not evaluated. The reason for this is that the WHAM model requires the availability of the large dataset that these experiments assume is not available.

In the Specific setting, DINOv2-GPT achieves the best overall performance, closely followed by DINOv2-MLP. Both models achieve a perfect FI of 1.00, indicating a replicated trajectory.

Additionally, we would like to note the overall difference that the training on the specific behaviour makes for the evaluation on that behaviour. In the general setting, when we evaluated the agents

Table 5: AUC and FI in Dojo Ramp for models trained on the whole dataset and evaluated zero-shot on Dojo Ramp **(general training)**, models trained only on Dojo Ramp **(specific training)**, and models first trained on the whole dataset and fine-tuned on Dojo Ramp **(fine-tuning)**.

	ConvNeXt-MLP		DINO-MLP		ConvNeXt-GPT		DINO-GPT	
	AUC	FI	AUC	FI	AUC	FI	AUC	FI
General training	0.65	0.22	0.40	0.21	**0.69**	**0.29**	**0.69**	0.24
Specific training	0.70	0.26	0.96	**1.00**	0.94	0.98	**0.97**	**1.00**
Fine-tuning	**0.96**	**1.00**	0.83	0.67	**0.96**	**1.00**	0.93	0.92

on the previously unseen Dojo Ramp behaviour, none of the models could follow the trajectory, while when trained on the specific, Dojo Ramp trajectory, almost all were able to capture the full behaviour.

For the fine-tuning setting, we see that the ConvNeXt-MLP, ConvNeXt-GPT and DINOv2-GPT are all able to successfully follow the trajectory, showing that the ConvNeXt architecture, due to being trained from scratch benefits from the large initial dataset from the general setting, and ultimately slightly outperforming the DINOv2 models.

We also present the impact of the input sequence length to understand the impact of future conditioning in this setting. The results in Table 6 show that even the BC version of the DINOv2-GPT model, where we don't condition the agent at all, still performs optimally. This is due to the nature of the training and evaluation setting. For the specific setting, there isn't any variety in the exhibited behaviour, where all trajectories follow the same path. In such a scenario, knowing the past is sufficient to predict the next action.

Table 6: AUC and FI for different past and future lengths, for Specific trajectories.

	ConvNeXt-MLP		DINO-MLP		ConvNeXt-GPT		DINO-GPT	
	AUC	FI	AUC	FI	AUC	FI	AUC	FI
10P-0F (BC)	0.70	0.26	0.96	**1.00**	**0.97**	**1.00**	0.93	0.99
10P-1F	0.71	0.26	0.74	0.85	0.96	**1.00**	**0.98**	**1.00**
10P-10F	0.70	0.26	0.96	**1.00**	0.94	0.98	**0.97**	**1.00**

5 CONCLUSION

We evaluated different IDM architectures across three settings – General, Specific, and Finetuned – finding that different architectures excel in different scenarios. ConvNeXt-GPT performed best in the General setting, DINOv2 variants in the Specific setting, and ConvNeXt variants in the Finetuned setting. Additionally, we conducted extensive ablations to identify key design choices for inverse dynamics models in imitation learning. Additionally, we explored various future selection strategies to mitigate distribution shifts, a crucial challenge in trajectory-following tasks.

Despite the advancements in model architectures and design choices, trajectory-following remains a significant challenge. In the General setting, none of the agents successfully followed the most complex trajectories, highlighting persistent gaps in generalisation across different behaviours.

Limitations and Future Work We plan on further investigating model robustness against external perturbations, such as human intervention, by manually rotating the camera and inputting specific actions, to better understand recovery from out-of-distribution situations.

While our models were trained on all 13 Bleeding Edge characters and actions, our evaluation focused on movement trajectories, with a single character. Expanding testing to include behaviours such as combat interactions and specific character actions could provide a deeper insight into model capabilities and limitations.

ACKNOWLEDGEMENTS

The authors would like to thank Katja Hofmann, Tabish Rashid, Tim Pearce, Yuhan Cao, Chentian Jiang, Shanzheng Tan, and Linda Yilin Wen at the Microsoft Research Game Intelligence Team for their immense support and contribution throughout this research.

REFERENCES

Dynamic Time Warping, pp. 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74048-3. doi: 10.1007/978-3-540-74048-3_4. URL https://doi.org/10.1007/978-3-540-74048-3_4.

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, pp. 1, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015430. URL https://doi.org/10.1145/1015330.1015430.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=NadTwTODgC.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Mark Ferguson, Sam Devlin, Daniel Kudenko, and James Alfred Walker. Imitating playstyle with dynamic time warping imitation. In *The 17th International Conference on the Foundations of Digital Games (FDG) 2022*, United States, May 2022. ACM.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/florence22a.html.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

Adam Jelley, Yuhan Cao, Dave Bignell, Sam Devlin, and Tabish Rashid. Aligning agents like large language models, 2024. URL https://arxiv.org/abs/2406.04208.

Anssi Kanervisto, Joonas Pussinen, and Ville Hautamäki. Benchmarking end-to-end behavioural cloning on video games. *CoRR*, abs/2004.00981, 2020. URL https://arxiv.org/abs/2004.00981.

Anssi Kanervisto, Dave Bignell, Linda Yilin Wen, Martin Grayson, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Tabish Rashid, Tim Pearce, Yuhan Cao, et al. World and human action models towards gameplay ideation. *Nature*, 638(8051):656–663, 2025.

Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J Foster, Lekan P Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=TNocbXm5MZ.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11966–11976, 2022. doi: 10.1109/CVPR52688.2022.01167.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt.

Keiran Paster, Sheila A. McIlraith, and Jimmy Ba. Planning from pixels using inverse dynamics models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V6BjBgku7Ro.

Brahma S. Pavse, Faraz Torabi, Josiah P. Hanna, Garrett Warnell, and Peter Stone. Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration. *IEEE Robotics and Automation Letters*, 5:6262–6269, 2019. URL https://api.semanticscholar.org/CorpusID:189999150.

Tim Pearce and Jun Zhu. Counter-strike deathmatch with large-scale behavioural cloning. In *2022 IEEE Conference on Games (CoG)*, pp. 104–111, 2022. doi: 10.1109/CoG51982.2022.9893617.

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. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Pv1GPQzRrC8.

Tim Pearce, Tabish Rashid, Dave Bignell, Raluca Georgescu, Sam Devlin, and Katja Hofmann. Scaling laws for pre-training agents and world models, 2024. URL https://arxiv.org/abs/2411.04434.

Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88.

Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL http://arxiv.org/abs/1011.0686.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *CoRR*, abs/1807.06158, 2018a. URL http://arxiv.org/abs/1807.06158.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018b. URL https://api.semanticscholar.org/CorpusID:23206414.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines, 2024. URL https://arxiv.org/abs/2408.14837.

Oriol Vinyals, I. Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, J. Chung, David H. Choi, Richard Powell, Timo Ewalds, P. Georgiev, Junhyuk Oh, Dan Horgan, M. Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, J. Agapiou, Max Jaderberg, A. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, D. Budden, Yury Sulsky, James Molloy, T. Paine, Caglar Gulcehre, Ziyun Wang, T. Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pp. 1–5, 2019a.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019b. URL https://api.semanticscholar.org/CorpusID:204972004.

Peter Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael Thomure, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602:223–228, 02 2022. doi: 10.1038/s41586-021-04357-7.

Yingchen Xu, Jack Parker-Holder, Aldo Pacchiano, Philip J. Ball, Oleh Rybkin, Stephen J. Roberts, Tim Rocktäschel, and Edward Grefenstette. Learning general world models in a handful of reward-free deployments. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=RuNhbvX9o9S.

Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement, 2019.

A APPENDIX

A.1 DATA MODALITY

The dataset consists of 8,788 recorded matches spanning from September 2, 2020, to October 19, 2022. In total, these trajectories represent approximately 1.12 years of cumulative human gameplay. The original 60Hz data has been downsampled to 10Hz. Each trajectory includes both visual and telemetry data, along with player actions taken during gameplay.

The resolution of the visual modality in the dataset is 600×360 which is then down-sampled to 128×128 before passing it to the encoder.

For the action space, we used the standard Xbox controller scheme. Xbox controller has continuous input through the movement sticks, and discrete inputs through buttons, as seen in Figure A.1.[3] We normalise the continuous action space into a [-1, 1] range, discretise it into 11 bins, and treat all buttons as discrete values.

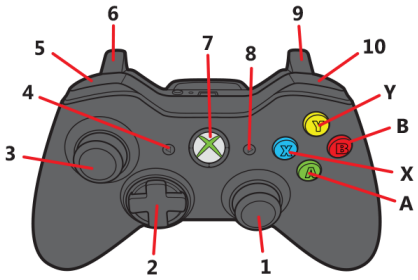


Figure A.1: Xbox controller input. Labels 1 and 3 represent the continuous stick inputs - each stick has two axes it can move in, while other labels represent the discrete button inputs.

A.2 TRAINING

A.2.1 HYPERPARAMETERS

To make the various MLP and GPT models comparable, all latent dimensions were set to 1024 for the ConvNeXt (Liu et al., 2022), DINOv2 (Oquab et al., 2024) and WHAM (Kanervisto et al., 2025) encoders. 4 layers with hidden dimensions of 1024 each were used for the MLP IDMs, whereas 4 GPT blocks with hidden dimensions of 1024 each were used for the GPT IDMs. In addition, 4 MLP layers with hidden dimensions of 1024 each were used as learnable state encoder layers for the pre-trained DINOv2 encoder. We trained on a single NVIDIA H100 80GB GPU machine. The learning rate used was 0.0001.

A.2.2 TRAINING CURVES

Figures A.2 to A.7 present the training curves of the model with separated continuous and discrete modalities, to further investigate the capability of the model for each action space.

[3]Controller image taken from: https://support.xbox.com/en-US/help/xbox-360/accessories/controllers

Figure A.2: Training loss curve.

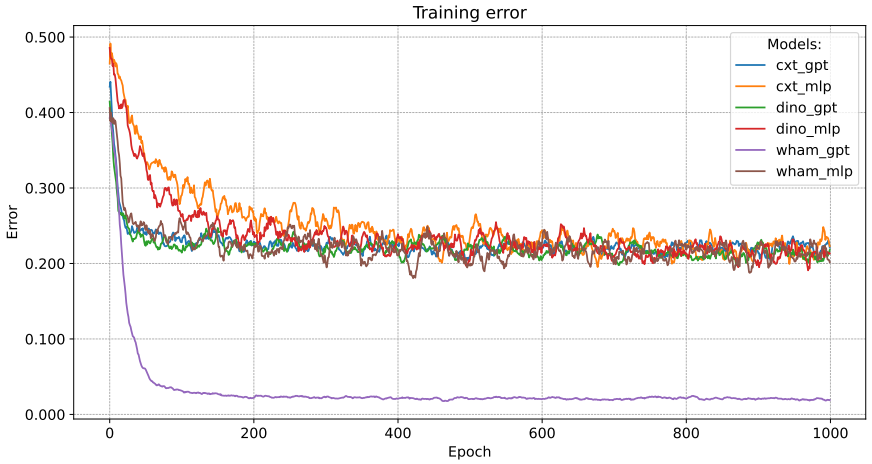


Figure A.3: Training error curve.

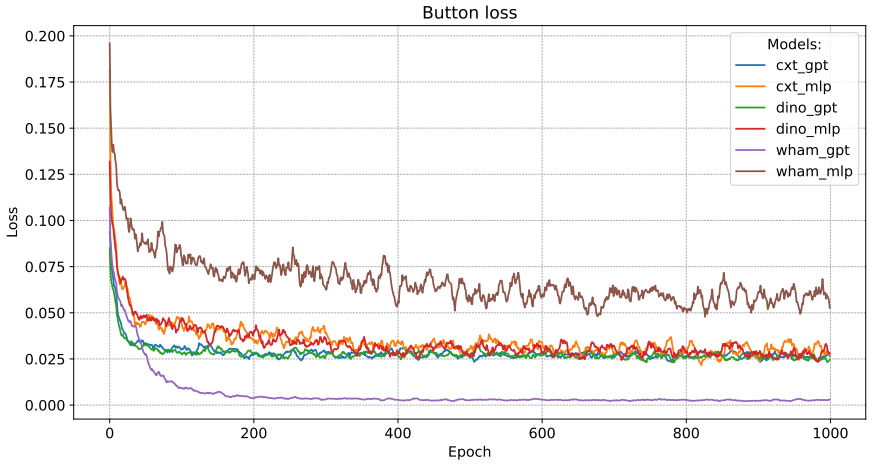


Figure A.4: Button loss curve for the six model variants, trained in the general setting.

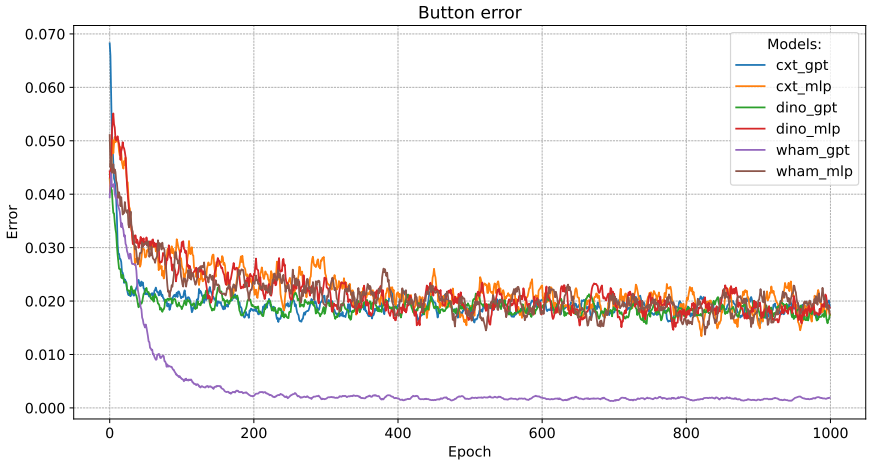


Figure A.5: Button error curve for the six model variants, trained in the general setting.

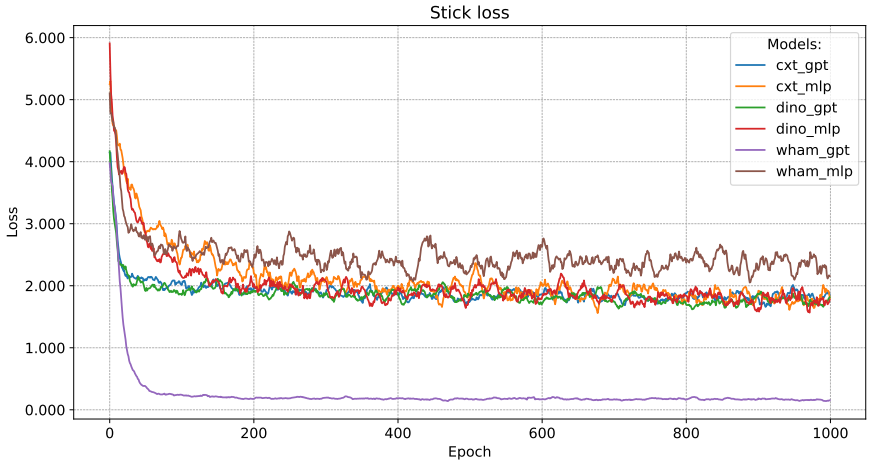


Figure A.6: Sticks loss curve for the six model variants, trained in the general setting.

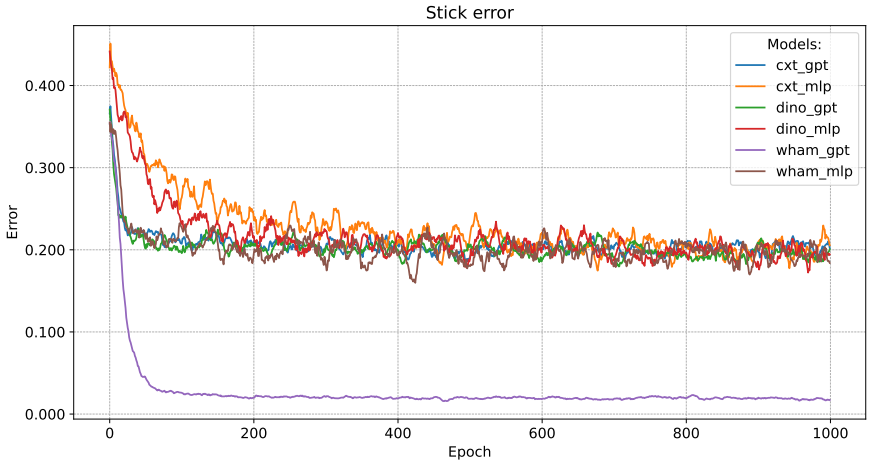


Figure A.7: Sticks error curve for the six model variants, trained in the general setting.

A.3 FUTURE SELECTION STRATEGIES

This section gives an extended view of the four future condition strategies from Section 3.2.

Static future conditioning Static future conditioning selects the first start timestep in the future based on the current timestep. Equation 1 states the formula for selecting the starting timestep of the future trajectory. This type of future conditioning is not conditioned on the spatial information.

Closest future conditioning The closest future conditioning uses the agent's current position and finds the timestep of the closest point in the ground truth trajectory measured by Euclidean distance. The formula can be seen in Equation 2.

Closest future conditioning takes into account the spatial distance between the agent and the conditioned point. By finding the closest point to the agent, we are minimising the spatial distance, keeping the agent in the training distribution for as long as possible.

The downside of this method is the potential to get stuck in an infinite loop. By ignoring the temporal aspect of the trajectory following, the agent is unable to differentiate between two frames in the same spatial position.

This can manifest in trajectories where the agent stands still for N frames and then starts moving afterwards. If N is larger than then the length of the trajectory used as an input to the agent, the spatial position of the goal state will be in the same spatial position as the starting state for which a trained IDM will do a no-op, since the goal is reached, causing an infinite loop, where the goal doesn't change, and the agent doesn't move, causing the goal not to change again.

Another example would be a trajectory containing loops, where the future selection strategy is not able to determine whether we are at the start or the end of the loop. Figure A.8 demonstrates this situation.

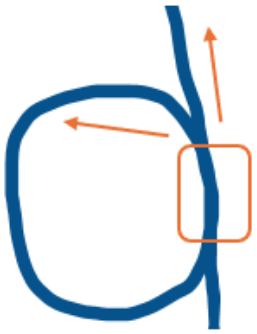


Figure A.8: Loop problem with closest future conditioning. Due to no temporal structure, if the trajectory contains two or more frames with the same location, the closest future conditioning strategy at the position outlined by the orange square doesn't know if the agent has already made the loop, and randomly chooses between the two possible future paths depicted by orange arrows.

Radius future conditioning Radius future conditioning moves the future start timestep by 1 if the agent is sufficiently close, i.e. inside of the specified radius, to the current future start timestep, as defined in Equation 3. Experimental results for different values for the radius are presented in Table A.1 in the Appendix.

Radius future conditioning takes into account both temporal and spatial distance. This approach successfully tackles the infinite loop examples in Figure A.8 due to the constraint that the future start timestep can only be increased.

Inner-Outer future conditioning The Inner-Outer future conditioning extends the Radius future conditioning by allowing it to progress the future for more than one frame. This method uses two radiuses, the inner radius and the outer radius. Experimental results for different values for the radius are presented in Table A.1.

There are three cases in which the agent could be:

1. Inside of the inner radius - We move the future until it leaves the inner radius. This causes the future conditioning to fix the issues where the agent can be ahead of the future conditioning due to incorrect actions, or FPS variation.
2. Outside of the inner radius, but inside of the outer radius - We move the future by 1 frame, same as the Radius selection strategy.
3. Outside of the outer radius - We do not move the future, same as the Radius selection strategy.

Table A.1: AUC results for different radius values

	Radius		Inner-Outer		
Trajectory	R=800	R=1600	I=100,O=200	I=200,O=400	I=800,O=3200
Jumppad left	**1.00**	**1.00**	0.70	0.98	**1.00**
Jumppad right	**0.97**	0.94	0.38	0.98	**1.00**
Jumppad mid	**1.00**	0.97	0.72	0.98	**1.00**
Benchmark 0	0.28	**0.48**	0.05	0.10	**0.79**
Benchmark 1	0.99	**1.00**	0.19	0.08	**1.00**
Benchmark 2	0.69	**0.94**	0.13	0.16	**0.79**
Dojo Ramp	0.33	**0.63**	0.17	0.21	**0.68**
Dojo Gong	0.25	**0.47**	0.06	0.18	**0.56**
Mean	0.69	**0.81**	0.30	0.55	**0.85**

A.4 Evaluation Metrics

This section defines the AUC metric used in the paper. For a reference trajectory τ and an agent rollout $\hat{\tau}$ represented by the state sequences $\tau = \{\boldsymbol{x}_t\}_{t=1}^{T}$ and $\hat{\tau} = \{\hat{\boldsymbol{x}}_t\}_{t=1}^{T}$ respectively, we define the coverage rate

$$f(\hat{\tau}, \tau, r) = \frac{1}{T} \sum_{t=1}^{T} \Big(\mathbb{I}(\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\| < r) \Big), \tag{4}$$

as the percentage of timesteps where the agent rollout is closer to the reference trajectory than a specified radius r. The indicator function $\mathbb{I}(\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\| < r)$ is 1 if $\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\| < r$ and 0 otherwise.

The AUC metric is defined as the average coverage rate for radius values $r \in [0, R]$ below a maximum radius R. This average is proportional to the area under the curve of f,

$$AUC(\hat{\tau}, \tau) = \frac{1}{R} \int_{0}^{R} f(\hat{\tau}, \tau, r) dr. \tag{5}$$

where the maximum radius R is defined as

$$R = max_{t \in [1,T]} \|\boldsymbol{x}_0 - \boldsymbol{x}_t\|. \tag{6}$$

On one hand, if the agent rollout is close to the reference trajectory, f grows rapidly as r increases resulting in a larger AUC value. On the other hand, if the agent rollout is not similar to the reference trajectory, for most values of r the coverage rate remains low resulting in a lower AUC value.

Additionally, by rescaling using the maximum radius R, the AUC metric is less dependent on the spread of the reference trajectory.

A.5 Full results

This section shows the full results, per trajectory per model.

Table A.2: FI results for the general setting evaluation on the 8 heldout trajectories. A higher FI is better.

	MLP			GPT		
Trajectory	ConvNeXt	DINOv2	WHAM	ConvNeXt	DINOv2	WHAM
Jumppad left	**1.00**	0.89	0.94	0.96	0.98	0.88
Jumppad right	**1.00**	**1.00**	0.87	0.97	**1.00**	0.88
Jumppad mid	**1.00**	0.98	0.69	**1.00**	0.85	0.92
Benchmark 0	0.25	0.21	0.18	**0.48**	0.30	0.25
Benchmark 1	0.61	0.63	0.14	**1.00**	0.54	0.43
Benchmark 2	0.62	0.21	0.09	**0.94**	0.92	0.33
Dojo Ramp	0.22	0.21	0.22	**0.29**	0.24	0.22
Dojo Gong	0.09	0.11	0.12	**0.18**	0.12	0.09
Mean	0.55	0.53	0.40	**0.73**	0.62	0.50

Table A.3: Ablation showing the performance of the ConvNeXt-GPT agent using different inputs to the model.

	Observations Only		Actions Only		Full Model	
Trajectory	AUC	FI	AUC	FI	AUC	FI
Jumppad left	**0.99**	**1.00**	0.93	**1.00**	0.99	0.96
Jumppad right	**0.99**	**1.00**	0.91	0.82	0.95	0.97
Jumppad mid	**0.99**	0.95	0.88	0.80	0.93	**1.00**
Benchmark 0	0.46	0.29	**0.53**	0.17	0.50	**0.48**
Benchmark 1	**0.98**	0.92	0.65	0.41	0.81	**1.00**
Benchmark 2	**0.93**	0.88	0.45	0.25	0.69	**0.94**
Dojo Ramp	**0.71**	0.27	0.55	0.21	0.63	**0.29**
Dojo Gong	**0.68**	**0.32**	0.67	0.08	0.67	0.18
Mean	0.84	0.70	0.69	0.47	**0.86**	**0.73**

Table A.4: Ablation showing the performance of the ConvNeXt-GPT agent with varying future and past lengths.

	BC		1P-1F		10P-1F		1P-10F		Full model	
Trajectory	AUC	FI	AUC	FI	AUC	FI	AUC	FI	AUC	FI
Jumppad left	0.90	0.78	0.99	**1.00**	0.99	0.96	0.99	**1.00**	0.99	0.96
Jumppad right	0.90	0.64	0.99	**1.00**	0.99	0.90	0.99	**1.00**	0.99	0.97
Jumppad mid	0.92	**1.00**	0.99	0.98	0.99	0.91	0.99	0.98	0.99	**1.00**
Benchmark 0	0.34	0.18	0.47	0.28	0.47	0.39	0.55	0.21	0.64	**0.48**
Benchmark 1	0.54	0.37	0.98	0.98	0.98	**1.00**	0.97	**1.00**	0.98	**1.00**
Benchmark 2	0.44	0.27	0.89	0.68	0.94	0.78	0.87	0.64	0.95	**0.94**
Dojo Ramp	0.55	0.21	0.72	**0.45**	0.73	0.40	0.54	0.23	0.65	0.29
Dojo Gong	0.51	0.11	0.69	0.18	0.72	**0.21**	0.44	0.04	0.70	0.18
Mean	0.64	0.45	0.84	0.69	0.85	0.69	0.79	0.64	0.86	0.73