Simulator-Free Visual Domain Randomization via Video Games

Chintan Trivedi [*1] **Nemanja Rašajski** [*1] **Konstantinos Makantasis** [1] **Antonios Liapis** [1] **Georgios N. Yannakakis** [1]

Abstract

Domain randomization is an effective computer vision technique for improving transferability of vision models across visually distinct domains exhibiting similar content. Existing approaches, however, rely extensively on tweaking complex and specialized simulation engines that are difficult to construct, subsequently affecting their feasibility and scalability. This paper introduces BehAVE, a video understanding framework that uniquely leverages the plethora of existing commercial video games for domain randomization, without requiring access to their simulation engines. Under BehAVE (1) the inherent rich visual diversity of video games acts as the source of randomization and (2) player behavior—represented semantically via textual descriptions of actions—guides the *alignment* of videos with similar content. We test BehAVE on 25 games of the first-person shooter (FPS) genre across various video and text foundation models and we report its robustness for domain randomization. BehAVE successfully aligns player behavioral patterns and is able to zero-shot transfer them to multiple unseen FPS games when trained on just one FPS game. In a more challenging setting, BehAVE manages to improve the zero-shot transferability of foundation models to unseen FPS games (up to 22%) even when trained on a game of a different genre (Minecraft). Code and dataset can be found at https://github.com/nrasajski/BehAVE.

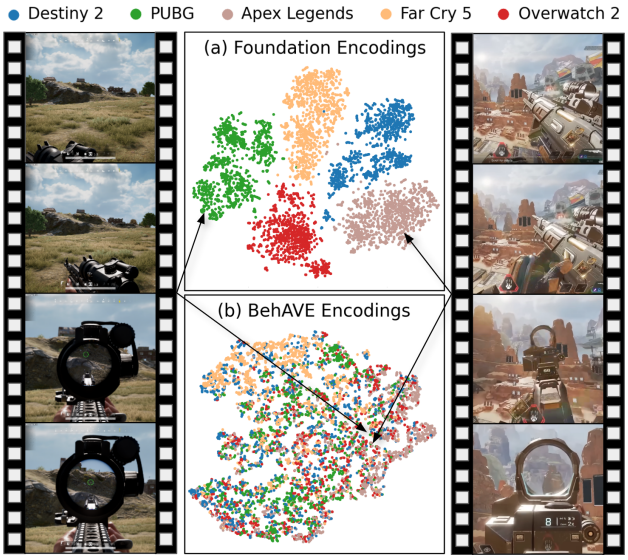


Figure 1. Illustrating the impact of behavior-alignment training within the **BehAVE** framework. The t-SNE plots display encodings of short video sequences from 5 distinct FPS games: (a) reveals the *domain gap* between encodings of different games obtained from a video foundation model, while (b) showcases encodings aligned by BehAVE. The framework closely positions encodings with similar player behavior (*e.g.*, aim gun) across visually diverse games such as *PUBG* (left) and *Apex Legends* (right) in this example.

1. Introduction

The construction of robust and generalizable computer vision (CV) models has emerged as a significant area of research in recent years (Kawaguchi et al., 2017; Yan et al., 2020; Li et al., 2022). As CV systems find applications across an increasing number of diverse scenarios ranging from virtual worlds (*e.g.*, *video games*) to real world settings (*e.g.*, *robotics*, *autonomous driving* and *sports analytics*), it becomes imperative to ensure their adaptability across visually distinct domains; that would, in turn, minimize the efforts needed for their seamless **transferability**. One highly promising avenue for enhancing the transferability of CV models is through **domain randomization** (Tobin et al., 2017), a simple technique that improves the robustness of a CV model by training it on visuals derived from randomizing rendering parameters of a simulator engine. Building accurate (world-realistic) and large-scale simulators, however, is a formidable challenge which requires significant time, expertise, and effort (Dosovitskiy et al., 2017). Identifying and randomizing relevant simulation parameters adds further to the complexity, making the entire process a resource-intensive endeavor.

[*]Equal contribution [1]Institute of Digital Games, University of Malta, Msida, Malta. Correspondence to: Chintan Trivedi <ctriv01@um.edu.mt>, Nemanja Rašajski <nemanja.rasajski@um.edu.mt>.

This project has received funding from the EU's Horizon 2020 programme unedr grant agreement No 951911.

To reduce the reliance of CV on costly simulations, in this paper we introduce a novel approach to domain randomization that instead leverages the rich visual diversity inherent in **video games**. Specifically we introduce a learning framework that distinguishes itself by not relying on game engine access for the randomization process. This unique feature unlocks the potential of CV to be trained and tested on videos from commercial-grade video game titles, a capability that has, until now, remained elusive. Our Behavior Alignment of Video Game Encodings (**BehAVE**) framework harnesses the unique characteristic of gameplay videos opposed to any other videos available. Gameplay video footage is generated by a latent generative factor: sequential player actions (*i.e.,* controller inputs) that control on-screen animations characterized as *player behavior*. Using player actions, BehAVE is able to align video encodings of similar player behaviors across visually diverse games (*i.e.*, different *domains*). Crucially, it employs *semantic action encoding*, a method that infuses semantic information about behavior through textual descriptions of actions, which are encoded using pre-trained text encoders (Patil et al., 2023). As a result, the text encodings of player actions, expressing a player's behavior, steer the behavioral alignment of video encodings.

We train BehAVE's alignment module, implemented on top of foundation video encoders (Tong et al., 2022), across a diverse array of games from the first person shooter (FPS) game genre, namely our introduced *SMG-25* dataset. BehAVE systematically restructures the video encoder's representation space and groups closely together the encodings of videos that showcase similar player behavior. Our findings suggest that BehAVE is able to uncover similar behavioral patterns—despite visual distinctions, such as variations in game style or aesthetics—across unseen games of the SMG-25 dataset (see Figure 1 for an illustrative example). The alignment training proves efficient even with small datasets comprising only a few games, and it demonstrates robustness across various tested video and text encoders.

To assess the *transferability* of BehAVE we examine the transfer capacity of the behavior-aligned video encodings in accurately classifying player behavior across various FPS games, while solely being trained on the FPS game *Counter Strike: GO* (Valve, 2012). Further, we test a more challenging scenario evaluating the transfer performance to the FPS genre from *Minecraft* (Mojang, 2011) a first person game from a different genre (non-FPS). Our findings indicate higher transferability when learning to classify behavior from our aligned representation space as compared to without, showcasing up to 22% higher classification accuracies across the different behavior categories tested. We view this as a potential avenue for annotating extensive datasets of online gameplay videos with behavior labels, thereby serving as a stepping stone towards learning generalized representations of behavior in videos.

The key contributions of this work are as follows. First, we introduce the **BehAVE** framework for domain randomization via commercial video games. It aligns video encodings of similar player behavior in video games using player actions as a supervision label, and notably, it operates without requiring access to the game engine. Second, we propose the innovative approach of **Semantic Action Encoding** for representing player actions as textual descriptions processed through a pretrained text encoder which serves the purpose of injecting semantic information into the alignment training. Third, we offer extensive experimental results of BehAVE on our newly introduced **SMG-25** dataset (containing synchronized frames and player actions of FPS gameplay from 25 different commercial games). Our findings indicate better transferability of behavior classification from aligned video encodings across stylistically diverse videos.

2. Background

Video Understanding in CV. Video understanding methods seek to interpret visual information embedded within temporal image-sequences. Recent strides in deep learning have led to attaining remarkable performance in diverse video understanding tasks, including but not limited to *video classification* (Bertasius et al., 2021), *video summarization* (Apostolidis et al., 2021), *short and long-form video understanding* (Wu & Krahenbuhl, 2021), and *object tracking* (Zhu et al., 2023). Current endeavors focus on training strategies that are independent of any specific downstream task. The resulting *video foundation models* (Wang et al., 2023a;b) yield powerful video representations, readily applicable across a diverse range of tasks. We use such foundation models in our study courtesy of their out-of-the-box performance and employ them as is (*i.e.* frozen) bounded by limited computational resources. This underscores the computational efficiency of our video understanding framework, ultimately enhancing its accessibility.

Transferable CV and Domain Randomization. Despite their impressive out-of-the-box performance, foundation models showcase limited capacities on transferring obtained knowledge from one domain to another *visually distinct* domain, primarily due to the "domain gap" challenge, as identified by Trivedi et al. (2021; 2023). Tobin et al. (2017) introduce the technique of *domain randomization* to train transferable vision models by introducing variability (*i.e.*, injecting domain gap) during learning, achieved through the randomization of rendering in a simulator generating training data. Leiprecht (2020) showcase the efficacy of domain randomization in CARLA (Dosovitskiy et al., 2017), a large-scale driving simulator. Mishra et al. (2022), however, bring to light the numerous complexities associated with identifying and tweaking relevant parameters of such

simulators. Furthermore, Kim et al. (2022) emphasize the *limited* variability that can be attained from a single simulator, impacting the transfer capacity (Yue et al., 2019). Hence, in this work, we adopt a simulator-free approach for visual domain randomization.

Video Games for CV. Inspired by insights from Risi & Togelius (2020) and Reed et al. (2022) suggesting that procedurally generated sets of diverse games enhance generality in machine learning, we explore the use of existing video games in CV. Several recent studies investigate the use of commercial-standard games as an alternative to dedicated in-lab simulators or procedural game level generation approaches, in an attempt to circumvent limitations related to inaccessible game engines. Notably, *Grand Theft Auto 5* (Rockstar, 2013) serves as a popular video game for collecting annotated data, achieved by intercepting rendering communication between graphics hardware and the screen buffer (Richter et al., 2016) or employing a game modification such as "infrared vision mod" (Gu et al., 2023). Alternatively, Pearce & Zhu (2022) gather internal game state information from *CS:GO* by probing the machine's memory. In contrast to such prior works involving the "reverse-engineering" of game engines, our approach instead captures high-level game information such as player actions using raw inputs from the machine's I/O devices, thereby simplifying the collection of annotated gameplay and boosting the scalability of our method across numerous commercial video games.

Multimodal Alignment of CV Models. Given that BehAVE considers different modalities of input such as videos of gameplay and corresponding player actions, we draw inspiration from contemporary work in *video action recognition* (Zhu et al., 2020). In their work with a paired video-text caption dataset, Song et al. (2016) extract *verbs* from captions and use them as *action labels*. Our framework builds on similar principles utilizing language models (Patil et al., 2023), but instead, encodes player actions; BehAVE then uses these action encodings for alignment with another modality, namely gameplay videos. To achieve this, we rely on *multimodal alignment frameworks* that operate with and align vision and language such as CLIP (Radford et al., 2021) and VideoCLIP (Xu et al., 2021; Ko et al., 2022). Drawing upon insights from the aforementioned studies, we propose a novel method for performing visual domain randomization with commercial games by aligning gameplay videos with semantically represented player actions.

3. The BehAVE Framework

As introduced earlier, we present *BehAVE* (Behavior Alignment of Video Game Encodings), a framework operating on paired visuals-and-actions datasets derived from commercial games, with the aim of aligning video encodings based on similar player behavior. The BehAVE method is illustrated in Algorithm 1 and visually depicted in Figure 2a. In Section 3.1, we explain the special structured dataset of games imperative for our framework, followed by Section 3.2 covering the encoder models used for both modalities. Finally, Section 3.3 details the training method employed.

Algorithm 1 Behavior-Alignment Training with BehAVE

Inputs: Games Dataset $\mathbb{D}$, semantic action mapper m, pre-trained text encoder h, pre-trained video encoder f and trainable alignment projector p.
for (video V, actions A) in $\mathbb{D}$ **do**
  Compute video encoding $z^{\text{video}} = f(V)$
  Project to aligned encoding $z^{\text{align}} = p(z^{\text{video}})$
  Convert to behavior text caption $A^{\text{caption}} = m(A)$
  Compute action encoding $z^{\text{caption}} = h(A^{\text{caption}})$
  Calculate loss $\mathcal{L}_{\text{cos}} = 1 - cosine(z^{\text{align}}, z^{\text{caption}})$
  Back-propagate $\mathcal{L}_{\text{cos}}$
  Update projector network parameters p_θ
end for
Output: Trained alignment projector p.

3.1. Games Dataset for Training BehAVE

A crucial component of our domain randomization framework involves the meticulous preparation of a dataset adhering to a specific *structure* that accommodates semantically similar visual content represented across diverse visual styles. We enforce this structure via the selection criteria of the various commercial games in the training dataset. Note that since BehAVE is trained upon player-game interaction data, we do not require access to the game engines, making it a viable strategy to use commercial games.

Game Selection for Domain Randomization. Let $\mathcal{G}$ represent a game, with a frame-renderer g, and $\mathcal{G} \in \mathcal{C}$, where $\mathcal{C}$ denotes the family of games of a certain game genre category. Given that domain randomization with customizable simulators involves the adjustment of simulator render parameters, we proceed to identify and formally define comparable parameters ξ within the context of video games. Each game's renderer encompasses certain game-specific parameters denoted as "game style parameters" ($\xi^{\mathcal{G}}$), associated with either the visual aesthetics of the game, such as *textures and color palette of game objects*, or the underlying rules governing the game, such as *game physics*. These parameters are considered invariant throughout the duration of the game, reflecting game design choices made during development phase, and are less likely to be shared across all games of this genre category. In the context of our analysis involving multiple commercial games, we observe $\xi^{\mathcal{G}} \in \Xi$ where Ξ represents the diverse *global game-design space*, introducing implicit "randomization" into our framework. This unique characteristic of games makes them ideal for

the purpose of visual domain randomization.

Additionally, we also characterize all game-state-specific parameters of the renderer, including the *player's spatial coordinates*, *health or ammunition status*, and *camera perspective*, as "game content parameters" ($\xi_t^{\mathcal{C}}$), which dynamically evolve at each timestep t in response to player interactions with the game environment. Note that these parameters remain largely consistent across different games that are categorized under the same genre.

Synchronized Gameplay Recording. At each timestep t, we obtain two synchronized information streams—visuals and actions. Player inputs or actions are selected from the shared action space of the game genre $\mathcal{C}$ and are represented by a set of $N^{\mathcal{C}}$ unique keypresses as $A_t = \{a_n\}_{n=1}^{N^{\mathcal{C}}}$, where $a_n \in \{0, 1\}$. Visuals are recorded in the form of RGB frames $F_t \subset \mathbb{R}^{h \times w \times 3}$ where h is height and w is width. The visuals of a game can be regarded as dynamic sequences of frames that arise from the interactions between the player and the game, as follows: $F_{t+1} = g(F_t, A_t, \xi_t^{\mathcal{C}} \mid \xi^{\mathcal{G}})$. Thus, the video frames are generated sequentially by the game renderer g processing the game content and player action information at every timestep for the given predefined game style. This inherent characteristic of gameplay visuals, derived from player interactions, enables us to employ actions for effectively discerning visual content.

Data Pre-processing. Although we collect data at the timestep level, our framework operates on videos for identifying behavior. To this end, we aggregate data over consecutive timesteps, forming a window of length T to obtain video sequences $V = (F_1, F_2, ..., F_T)$ and action sequences $A = (A_1, A_2, ..., A_T)$. Consequently, for each game, the synchronized gameplay-actions dataset is denoted as $D^{\mathcal{G}} = \{(V_i, A_i)\}_{i=1}^{I}$ where $|D^{\mathcal{G}}| = I$. We construct the *overall* dataset $\mathbb{D} = \bigcup_{\mathcal{G} \in \mathcal{C}} D^{\mathcal{G}}$ comprising of $k = |\mathbb{D}|$ "distinct" games from the game genre $\mathcal{C}$, adhering to the previously outlined selection strategy. This dataset forms the basis for behavior alignment training.

3.2. Encoding the Modalities

Video Encoding and Alignment. As previously stated, we harness the capabilities of a pre-trained video foundation model in our study for video understanding. Let f denote the backbone model of a *video encoder*; thus, the latent representation of the backbone's video encoding can be given by $z^{\text{video}} = f(V)$ (refer to line 3 of Algorithm 1). Note that we employ f within our training in a frozen state, a decision influenced by our evaluation of foundational models as well as other computational constraints. To facilitate the alignment of representation spaces of different modalities within our framework, we employ a trainable MLP model p that acts as an *alignment projector* and operates on top of the video encoder, yielding the aligned projection encoding $z^{\text{align}} = p(z^{\text{video}})$ (refer to line 4 of Algorithm 1).

Semantic Action Encoding. As previously explained, each video is associated with a sequence of binary actions A, indicating the presence or absence of each key-press action at every timestep. However, binary labels for actions offer limited insights into the inter-relationships among various sub-actions. For instance, in the context of an FPS game, the binary encodings of the four actions—*move left*, *move right*, *shoot gun*, and *aim gun*—are equidistant from one another. This encoding type, however, fails to capture the underlying semantic similarity between the first two actions (*i.e.*, related to movement), the last two actions (*i.e.*, related to weapon use) and also the semantic difference between these two behavioral categories.

To address the above limitation, we propose to equip BehAVE with a hand-crafted *semantic action mapper* function m which injects semantic information into the action encodings via text. It maps the binary sequence of keypresses to a behavior text caption $A^{\text{caption}} = m(A)$ that describes the semantic behavior associated with that action, within the context of the game genre (refer to line 5 of Algorithm 1). Then, we use a pre-trained text foundation model in the form of a *text encoder* h that gives the caption's text encoding $z^{\text{caption}} = h(A^{\text{caption}})$ (refer to line 6 of Algorithm 1). We argue that such pre-trained encoders will be able to better capture the inter-relationships among the joint distribution of actions that are otherwise difficult to represent with binary action encodings.

3.3. Alignment Training

Upon obtaining the encodings of videos and actions, we initiate the training phase of the framework using the specified dataset of games. We observe different video sequences exhibiting similar behavior across different games. Consequently, to align the representation space of the video encoder to match that of the text encoding of behavior, we choose to train our projector head as attached to the video encoder. Subsequently, we use a loss $\mathcal{L}_{\cos}$ (refer to line 7 of Algorithm 1) based on the cosine similarity between the video projector encoding and the behavior text encoding so that the former aligns with the latter on the same (shared) representation space. The loss is defined as follows:

$$\mathcal{L}_{\cos}(z^{\text{align}}, z^{\text{caption}}) = 1 - \frac{z^{\text{align}} \cdot z^{\text{caption}}}{\|z^{\text{align}}\|_2 \, \|z^{\text{caption}}\|_2} \quad (1)$$

Upon completion of the alignment training, the video encoder equipped with the trained alignment projector can be utilized on any other visual content for video understanding, without requiring access to any other modalities—such as player actions—that only pertain to games.

In summary, the introduced BehAVE framework operates

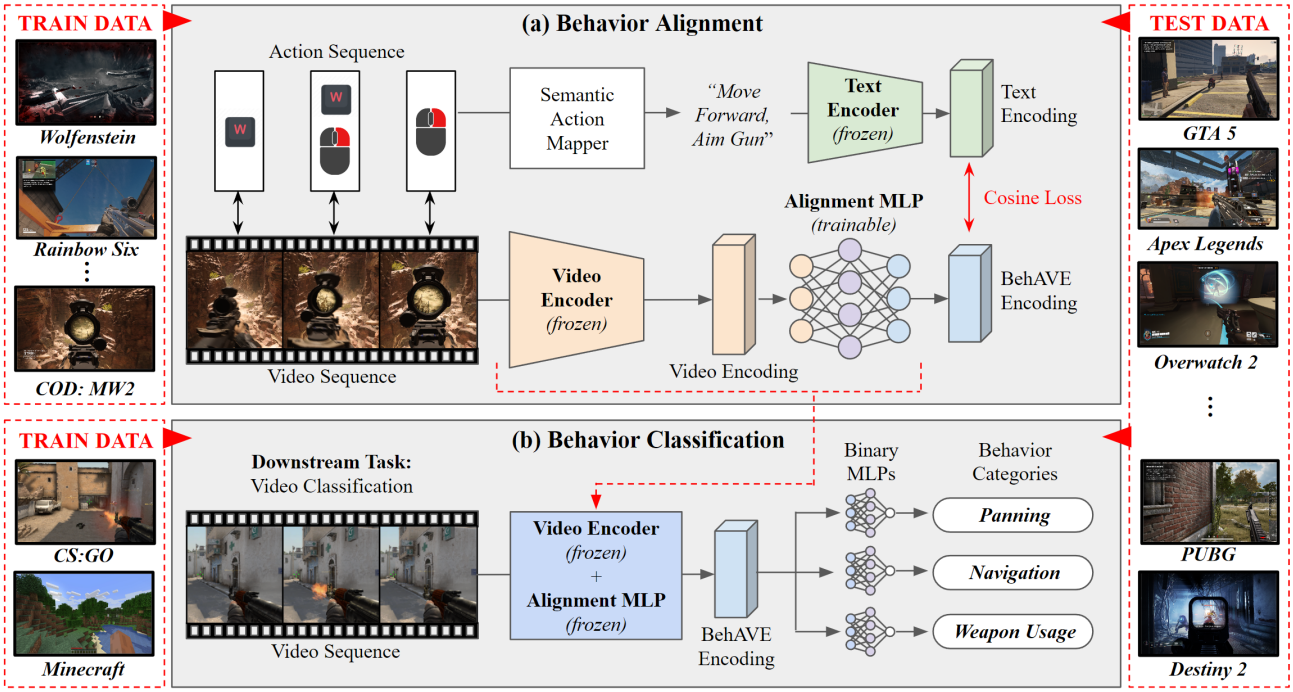


Figure 2. The general framework of the experiments presented and corresponding datasets employed in our study: **(a) Behavior Alignment:** BehAVE is trained on synchronized pairs of gameplay video and player actions of the SMG-25 train dataset and evaluated on unseen (test) games of the SMG-25 dataset. **(b) Behavior Classification:** We test the transferability of a *video classification* downstream task with behavior category as the class label. BehAVE is trained independently on CS:GO and Minecraft, and transferred to SMG-25 test dataset. Note the disjoint train and test sets of games.

as follows. The structurally enriched dataset of the framework facilitates domain randomization, the pre-trained video encoder enables video understanding, the semantic action encoding introduces the semantic notion of behavior, and the alignment training module ensures enhanced transferability in video understanding.

4. Experiments

Figure 2 outlines the two primary experiments conducted in our study: (a) *Behavior Alignment*, where we perform alignment training (Section 4.2), and (b) *Behavior Classification*, where we assess transferability of the aligned models in a downstream classification task (Section 4.3). Before delving into the experiments, in Section 4.1 we introduce the datasets and evaluation metrics employed. Note that all experiments involving model training and inference have been carried out on a single GTX 1070 (8GB) GPU, highlighting the cost-effective nature of our method.

4.1. Dataset and Metrics

We test BehAVE on three datasets, namely *SMG-25*, *CS:GO*, and *Minecraft*, across the two experiments of our study. The **SMG-25** (Synchronized Multi-Game FPS Dataset) is our newly introduced dataset that encompasses synchronized gameplay visuals and player action data from multiple commercial First Person Shooter (FPS) games, gathered following the structure outlined in Section 3.1. It comprises over ∼250K data points spanning 25 visually diverse FPS games, encompassing actions related to player behavior categories such as *panning* (player looking around), *navigation* (player moving in the environment) and *weapon usage* (player engaging the equipped weapon). We partition it into an SMG-25 *train set* for use in Experiment (a) and an SMG-25 *test set* used for evaluations in both Experiments (a) and (b). The train-test splits consist of disjoint sets of games, enabling evaluation of zero-shot performance on unseen games. Further details about this dataset are available in Appendix A. Additionally, we source similar gameplay-actions data from other commercial games, namely **CS:GO** (180K data points of *cs-dust* level from Pearce & Zhu (2022)) and **Minecraft** (50K data points of *contractor demonstrations* from Baker et al. (2022)), which serve as training datasets for Experiment (b).

Evaluation Metrics. To comprehensively assess the alignment quality in Experiment (a) and the transferability of aligned models in Experiment (b), we employ several metrics. The *Silhouette Score* (Rousseeuw, 1987), ranging from

Table 1. The video and text foundation models used in our experiments. (†) For video and text models, respectively, input size indicates timesteps of RGB frames and maximum token length. The number of parameters and the encoding size of the models are also listed.

INPUT	PRETRAIN METHOD	MODEL	INPUT SIZE†	#PARAMS	ENCODING SIZE
VIDEO	I3D (CARREIRA & ZISSERMAN, 2017)	3D-CONVNET	$16 \times 3 \times 224 \times 224$	79M	512
	VIDEOMAEV2 (WANG ET AL., 2023A)	VIT-BASE	$16 \times 3 \times 224 \times 224$	87M	768
	MVD (WANG ET AL., 2023B)				
TEXT	GPT-2 (RADFORD ET AL., 2019)	TRANSFORMER	512 TOKENS	110M	768
	CLIP (RADFORD ET AL., 2021)		77 TOKENS	63M	512
	BERT (REIMERS & GUREVYCH, 2019)		256 TOKENS	33M	384

-1 to 1, quantifies the cluster quality of embeddings through intra-cluster compactness and inter-cluster separation. In Experiment (a), it is used to gauge the effectiveness of the alignment projection based on behavior categories as cluster labels, with a higher score indicating better-defined clusters. Additionally, the *Transferability Score* evaluates results in Experiment (b) by considering the percentage difference in classification test accuracy between models trained on BehAVE encodings and those trained on foundation video encodings. A positive difference signifies better transferability for the alignment method relative to the corresponding foundation method, and vice versa.

4.2. Behavior Alignment

In this experiment, depicted in Figure 2a, we test the BehAVE framework on the SMG-25 dataset. Table 1 provides details on various pre-trained video and text encoders analyzed, while a comprehensive report on the configurations tested is available in the Appendix (see Table 6). For the alignment projector, we opt for a 4-layer MLP with ReLU activations and 50% dropout. The size of the final layer of the MLP is adjusted to match the encoding size of the selected text encoder. As elaborated next, we perform a thorough analysis of various design choices incorporated into our framework.

Action Encoding Schemes. We conduct a comparative study on our semantic action encodings, focusing on actions alone, without videos, in our dataset. We compare the cluster quality of actions encoded traditionally in binary form (*i.e., keypress labels*) to text encodings (*i.e., behavior captions*), with the aim of highlighting benefits of infusing semantics into our BehAVE framework. Results using only the unique set of actions from SMG-25 are presented in Section 5.1.

Impact of Alignment Training. Subsequently, we train BehAVE with the previously mentioned pre-trained video and text encoders and analyze the benefits of aligning the representation space of the video encoder with that of the text encoder. The alignment projector is trained for 10 epochs using the *adam* optimizer with a learning rate of $1e^{-3}$. The training data comprises a set of 15 games from SMG-25, processed in batch sizes of 128. The resulting representation space post-alignment is evaluated using the *silhouette score* metric on 10 games from the SMG-25 test set, allowing us to assess the "zero-shot" performance of our methods on unseen games. Results from 5 independent runs, with randomized train-test splits for each run, are presented in Section 5.1.

Sensitivity Analysis of k: An insightful study from the perspective of domain randomization involves analyzing how the number of diverse games (k) in BehAVE's training process affects alignment performance. For that purpose, we explore the impact of varying the number of games included in the training set, ranging from 1 to 15 and compare them on a fixed test set of 10 games. We randomize the selection of games in the train and test sets across the 5 independent runs reported in Section 5.1.

4.3. Behavior Classification

Following alignment training, we emphasize the benefits of BehAVE encodings for transferring a downstream task across visually distinct domains. For this purpose, we select video classification as a representative task in video understanding, with behavior categories serving as the class labels. As depicted in Figure 2b, for each behavior category, we train a classifier (3-layer MLP with binary output) on video encodings of a single game (source domain) not included in our SMG-25 dataset, and evaluate the performance of this classifier on multiple FPS games from the SMG-25 test set (target domains). We first use foundation video encodings as input to this classifier, and then BehAVE encodings are employed in the same fashion. We ultimately compare the classifiers' performance on FPS games between the two methods using the *transferability score* metric. We first perform this experiment with our source domain being CS:GO, a game of the *same* FPS genre, followed by a more challenging experiment with our source domain being Minecraft, a *similar* first-person perspective game but from non-shooter genre. In both cases, we evaluate classification performance on the SMG-25 test set of games not encountered in alignment training to gauge the zero-shot transfer capacity of our classifiers. Transferability results from both CS:GO and

Table 2. **Impact of Alignment Training.** Silhouette scores (average and standard deviation over 5 runs) for behavioral and game labels on SMG-25 test set. Higher is better for the behavior categories and lower is better for the game labels, highlighted in bold.

Alignment (Approach)	Encoder Methods (Video - Action)	Alignment Dimension	Behavior Labels Panning ↑	Navigation ↑	Weapon ↑	Game Label ↓
Foundation (Baseline)	I3D - None	512	$0.03_{\pm 0.04}$	$0.04_{\pm 0.04}$	$0.00_{\pm 0.01}$	$0.07_{\pm 0.07}$
	VideoMAEv2 - None	768	$0.08_{\pm 0.00}$	$0.13_{\pm 0.00}$	$0.03_{\pm 0.00}$	$0.10_{\pm 0.00}$
	MVD - None	768	$0.14_{\pm 0.02}$	$0.09_{\pm 0.04}$	$0.01_{\pm 0.01}$	$-0.05_{\pm 0.02}$
Keypress (Naive)	I3D - Binary	16	$0.36_{\pm 0.01}$	$0.45_{\pm 0.01}$	$0.30_{\pm 0.02}$	$-0.21_{\pm 0.01}$
	VideoMAEv2 - Binary	16	$0.35_{\pm 0.00}$	$0.48_{\pm 0.01}$	$0.32_{\pm 0.01}$	$-0.16_{\pm 0.00}$
	MVD - Binary	16	$0.43_{\pm 0.02}$	$0.44_{\pm 0.03}$	$0.09_{\pm 0.01}$	$-0.24_{\pm 0.02}$
BehAVE (Ours)	VideoMAEv2 - GPT-2	768	$\mathbf{0.51_{\pm 0.02}}$	$\mathbf{0.58_{\pm 0.05}}$	$0.20_{\pm 0.05}$	$\mathbf{-0.29_{\pm 0.06}}$
	VideoMAEv2 - CLIP	512	$0.40_{\pm 0.01}$	$0.49_{\pm 0.01}$	$\mathbf{0.35_{\pm 0.00}}$	$-0.20_{\pm 0.01}$
	VideoMAEv2 - BERT	384	$0.41_{\pm 0.03}$	$0.48_{\pm 0.04}$	$\mathbf{0.35_{\pm 0.02}}$	$-0.17_{\pm 0.03}$

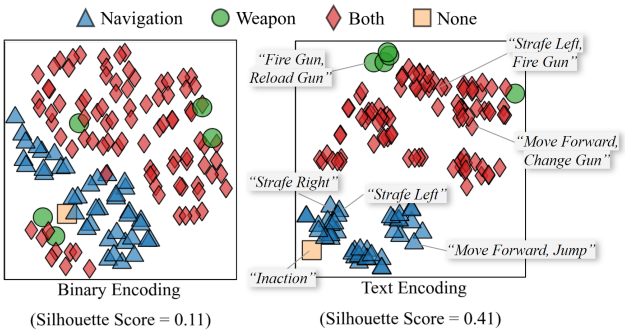


Figure 3. **Comparing Action Encoding Schemes.** t-SNE embeddings and corresponding silhouette scores of actions encoded as binary labels (left) compared to pretrained text encoders (right).

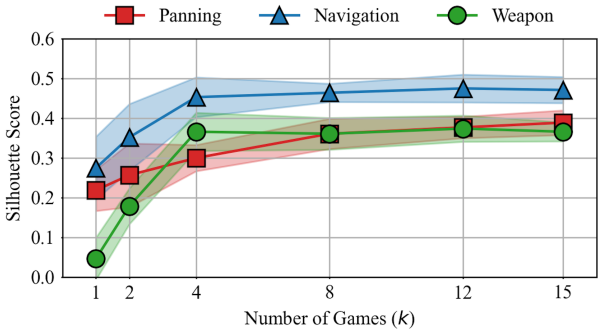


Figure 4. **Sensitivity Analysis of *k*.** Varying the number of games included in alignment training and observing the outcome on clustering of behavior categories across 10 test (unseen) games.

Minecraft experiments are reported in Section 5.2.

5. Results

5.1. Behavior Alignment

Comparing Action Encoding Schemes. Figure 3 presents the comparison between binary action encoding and semantic text encoding from the pre-trained CLIP text encoder (Radford et al., 2021). The silhouette score for binary actions (0.11) is significantly lower than that of text encodings (0.41), indicating superior cluster quality for the latter. This advantage is also evident when encodings are projected onto two dimensions using t-SNE which shows that for actions belonging to the same behavior categories, we observe more compact and well-separated clusters. This brings forth the benefit of encoding actions not merely as binary keypresses but rather as text encodings based on semantics.

Impact of Alignment Training. Table 2 presents the analysis of the representation spaces before and after running BehAVE's alignment training. First, we observe that for all three behavior categories, even alignment based on a naive approach of binary encoding of actions (see middle block of table) improves clustering quality across all video foundation models tested (upper block). Second, more interestingly, we observe even bigger improvements when using BehAVE's text encoding scheme (lower block). As a consequence of this alignment, we also observe that the domain gap, indicated by the clustering results based on game labels, reduces for all alignment approaches compared to the foundation approach. Evidently, the benefits of BehAVE are apparent across all tested configurations; a comprehensive list of all video and text encoders are provided in Table 6 of the Appendix.

Sensitivity Analysis of *k*. Figure 4 illustrates the sensitivity of the domain randomization process to the number of games (k) used in training. Notably, the performance varies across different behavior categories. For *weapon usage* and *navigation*, we observe convergence between 4 to 6 games, indicating that a relatively small number of games is sufficient for identifying these categories. Surprisingly, for *panning*, alignment continues to improve (even moderately) beyond 10 games. We argue that this phenomenon is attributed to the substantial variability in panning actions across dif-

ferent games, particularly due to different mouse sensitivity presets in SMG-25 that were approximated through visual inspection rather than precise extraction from the game engine (refer to Appendix A.2 for more details). As a result, in our experiments with SMG-25, we chose 15 games for training and 10 for testing to balance the effectiveness of domain randomization while maintaining a sufficiently large test set for zero-shot performance evaluation.

5.2. Behavior Classification

Transfer from CS:GO (*Same* Genre). Figure 5a showcases transferability scores we observe across all three behavior categories, when the source domain is CS:GO and target domains are multiple FPS games of the SMG-25 test set. Notably, we notice the poor transfer capacity of the foundation encodings based classifiers due to the aforementioned domain gap problem. In the case of BehAVE, while the absolute performance of the classifiers on source domains decline (highlighted in red above the respective bars), we observe between 3% to 22% higher performance on transfer to target domains (highlighted in green). These findings highlight the advantage in transferability we obtain with BehAVE, particularly in the context of zero-shot performance on unseen games of the same genre.

Transfer from Minecraft (*Similar* Genre). Figure 5b showcases similar benefits when using Minecraft as the source domain, despite the non-shooter nature of its genre. While *panning* category obtains minimal improvement, we observe between 6% to 22% improvements in classifying *navigation* and *weapon* categories, respectively. This finding is surprising since the weapon equipped by the Minecraft player is a pick-axe, but the classifier is still able to transfer this knowledge to the gun-related behaviors of SMG-25 games. This indicates that our model, trained on a diverse dataset encompassing multiple game genres with aligned semantic behavior patterns, can generalize its knowledge to new game genres with overlapping behavioral characteristics. It is crucial to acknowledge, however, that while the obtained results already showcase the transferability capacity of BehAVE even across genres, our exploration in this direction is preliminary; further investigation on across-genre transferability is deferred to future work.

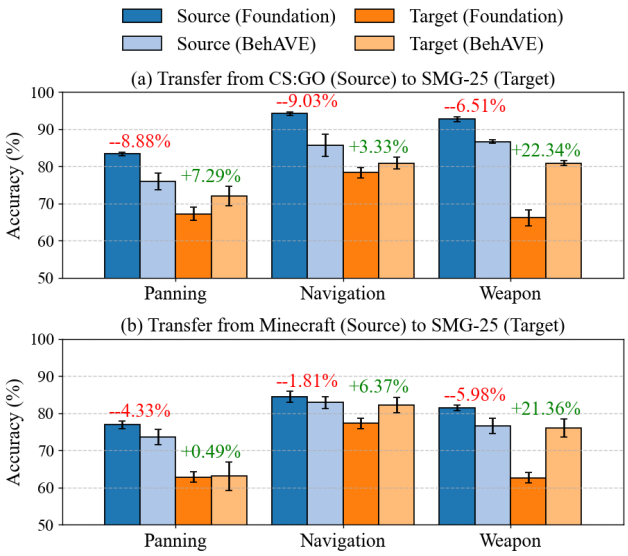


Figure 5. **Behavior classification** accuracy across the 3 behavior categories when transferring from (a) CS:GO (FPS game) and (b) Minecraft (non-FPS game) to unseen games of the FPS genre. While BehAVE (aligned) encodings perform slightly worse on the test sets of source domains over foundation (unaligned) encodings, we notice high improvements in generalization to the different target domains, highlighting BehAVE's enhanced transfer capacities.

6. Discussion

Downstream Applications. BehAVE's zero-shot performance presents promising applications in imitation learning, demonstrated across both first-person shooter and non-shooter game genres. Future experiments could explore training on multiple genres and measuring transferability to entirely different, unseen genres. While our focus has been on transferable video understanding, a complementary study could delve into representation learning optimized for specific downstream tasks. Fine-tuning foundation modules might balance performance and transferability, tailoring BehAVE for specific applications. Section D.2 of the Appendix briefly discusses future research directions delving into BehAVE's potential in learning inverse dynamics models.

Scalability. As outlined in Section 4, we operate within a limited compute budget, hindering data collection expansion and comparisons with end-to-end trained models. BehAVE could potentially perform better with increased data and compute resources, but our focus is on introducing an accessible method that proves effective even at a smaller scale.

Ethical Considerations. Recognizing inherent risks in FPS game data and ethical considerations in deploying such data for real-world applications is crucial. A comprehensive analysis of these implications is imperative for responsible use and future research in this area (Melhart et al., 2023).

7. Conclusion

In this paper, we introduced BehAVE, a video understanding framework that employs a simulator-free domain randomization technique, exploiting the inherent variations in graphical and animation styling found in commercial video games. Tested extensively within games of the first-person shooter genre, BehAVE is able to successfully align video encodings (displaying similar player behavior) across visually distinct games. Compared to foundation models, BehAVE demon-

strates significantly higher levels of zero-shot transferability to unseen FPS games in a downstream behavior classification task when trained independently on a game from the same genre (CS:GO) and even a game from a different genre (Minecraft). The introduced BehAVE framework showcases superior zero-shot transferability capacities offering a robust and efficient method towards generalizing perception across visually diverse environments.

Acknowledgments

We would like to express our gratitude to Tim Pearce for insightful discussions that contributed to the development of this work. Special thanks also go to Roberto Gallotta and Marvin Zammit for their invaluable assistance in collecting the necessary data for our research.

References

Apostolidis, E., Balaouras, G., Mezaris, V., and Patras, I. Combining global and local attention with positional encoding for video summarization. In *2021 IEEE international symposium on multimedia (ISM)*, pp. 226–234. IEEE, 2021.

Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., and Auli, M. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, pp. 1298–1312. PMLR, 2022.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In *ICML*, volume 2, pp. 4, 2021.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715. If you use this software, please cite it using these metadata.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Gu, X., Liu, G., Zhang, X., Tang, L., Zhou, X., and Qiu, W. Infrared-visible synthetic data from game engine for image fusion improvement. *IEEE Transactions on Games*, 2023.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pp. 4904–4916. PMLR, 2021.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 1(8), 2017.

Kim, Y.-w., Mishra, S., Jin, S., Panda, R., Kuehne, H., Karlinsky, L., Saligrama, V., Saenko, K., Oliva, A., and Feris, R. How transferable are video representations based on synthetic data? *Advances in Neural Information Processing Systems*, 35:35710–35723, 2022.

Ko, D., Choi, J., Ko, J., Noh, S., On, K.-W., Kim, E.-S., and Kim, H. J. Video-text representation learning via differentiable weak temporal alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5016–5025, 2022.

Leiprecht, S. Using simulations and domain randomization for autonomous driving. *Technical Reports in Computing Science*, (July 2020):1–4, 2020.

Li, J., Chen, G., Tang, Y., Bao, J., Zhang, K., Zhou, J., and Lu, J. Gain: On the generalization of instructional action understanding. In *The Eleventh International Conference on Learning Representations*, 2022.

Li, W., Zhu, L., Wen, L., and Yang, Y. Decap: Decoding clip latents for zero-shot captioning via text-only training. *arXiv preprint arXiv:2303.03032*, 2023.

Melhart, D., Togelius, J., Mikkelsen, B., Holmgård, C., and Yannakakis, G. N. The ethics of ai in games. *IEEE Transactions on Affective Computing*, 2023.

Mishra, S., Panda, R., Phoo, C. P., Chen, C.-F. R., Karlinsky, L., Saenko, K., Saligrama, V., and Feris, R. S. Task2sim: Towards effective pre-training and transfer from synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9194–9204, 2022.

Patil, R., Boit, S., Gudivada, V., and Nandigam, J. A survey of text representation and embedding techniques in nlp. *IEEE Access*, 2023.

Pearce, T. and Zhu, J. Counter-strike deathmatch with large-scale behavioural cloning. In *2022 IEEE Conference on Games (CoG)*, pp. 104–111. IEEE, 2022.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Richter, S. R., Vineet, V., Roth, S., and Koltun, V. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 102–118. Springer, 2016.

Risi, S. and Togelius, J. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, 2020.

Rousseeuw, P. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

Singh, A., Hu, R., Goswami, V., Couairon, G., Galuba, W., Rohrbach, M., and Kiela, D. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15638–15650, 2022.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

Song, Y. C., Naim, I., Al Mamun, A., Kulkarni, K., Singla, P., Luo, J., Gildea, D., and Kautz, H. A. Unsupervised alignment of actions in video with text descriptions. In *IJCAI*, pp. 2025–2031, 2016.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Tong, Z., Song, Y., Wang, J., and Wang, L. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Trivedi, C., Liapis, A., and Yannakakis, G. N. Contrastive learning of generalized game representations. In *2021 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2021.

Trivedi, C., Makantasis, K., Liapis, A., and Yannakakis, G. N. Towards general game representations: Decomposing games pixels into content and style. *arXiv preprint arXiv:2307.11141*, 2023.

Wang, L., Huang, B., Zhao, Z., Tong, Z., He, Y., Wang, Y., Wang, Y., and Qiao, Y. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14549–14560, June 2023a.

Wang, R., Chen, D., Wu, Z., Chen, Y., Dai, X., Liu, M., Yuan, L., and Jiang, Y.-G. Masked video distillation: Rethinking masked feature modeling for self-supervised video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6312–6322, 2023b.

Wu, C.-Y. and Krahenbuhl, P. Towards long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1884–1894, 2021.

Xu, H., Ghosh, G., Huang, P.-Y., Okhonko, D., Aghajanyan, A., Metze, F., Zettlemoyer, L., and Feichtenhofer, C. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv:2109.14084*, 2021.

Yan, X., Misra, I., Gupta, A., Ghadiyaram, D., and Mahajan, D. Clusterfit: Improving generalization of visual representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6509–6518, 2020.

Yue, X., Zhang, Y., Zhao, S., Sangiovanni-Vincentelli, A., Keutzer, K., and Gong, B. Domain randomization and pyramid consistency: Simulation-to-real generalization

without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2100–2110, 2019.

Zhu, Y., Li, X., Liu, C., Zolfaghari, M., Xiong, Y., Wu, C., Zhang, Z., Tighe, J., Manmatha, R., and Li, M. A comprehensive study of deep video action recognition. *arXiv preprint arXiv:2012.06567*, 2020.

Zhu, Z., Hou, J., and Wu, D. O. Cross-modal orthogonal high-rank augmentation for rgb-event transformer-trackers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22045–22055, 2023.

Appendix

A. The *SMG-25* Dataset

In this section we provide additional information detailing the entire pipeline of collecting, pre-processing and curating the SMG-25 dataset introduced in Section 4.1. Note that we intend to make this dataset publicly available upon the acceptance of the paper.

A.1. Data Collection

Hardware and software specifications: Our data collection was carried out on a handheld gaming machine running Windows 11 operating system, equipped with an AMD 780M iGPU with 4GB of VRAM. As illustrated in Figure 6, 25 different commercial games were sourced from various game distribution platforms such as Steam (*Valve, 2003*), Xbox Game Pass (*Microsoft, 2017*) and Ubisoft Connect (*Massive, 2020*). We run all games in full-screen mode on an external FullHD monitor connected to the handheld machine, along with keyboard and mouse attachments for player inputs. While running the game on the monitor screen, we visualize the collected data in real-time on the handheld's screen to ensure good quality and correctness of the data being recorded.

Game selection criteria: We ensure that the games selected for inclusion in the dataset are listed under the first person shooter category on their respective distribution platforms, and that we do not select games whose content policy might

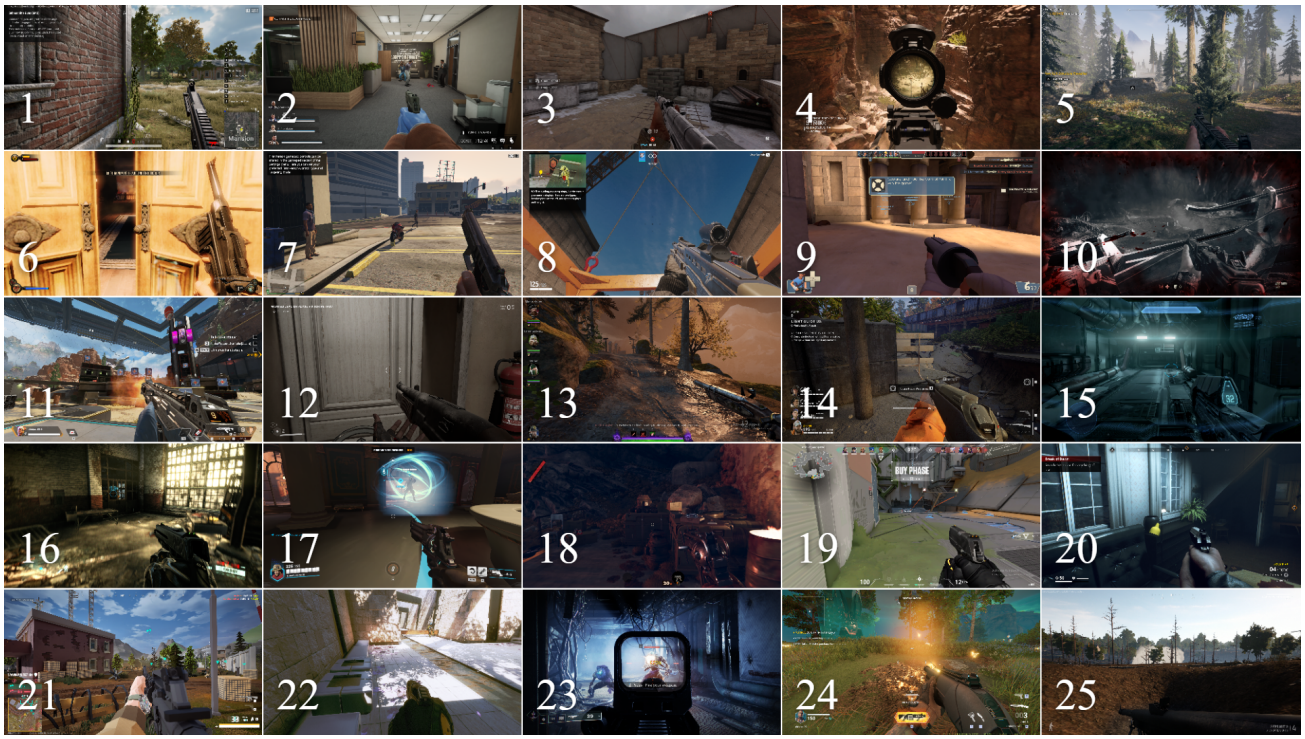


Figure 6. Screenshots from all 25 FPS games in the SMG-25 dataset, highlighting diverse graphical styles of the games considered: **[1]** PUBG: Battlegrounds (*PUBG Studios, 2017*); **[2]** Payday 3 (*Starbreeze Studios, 2023*); **[3]** Insurgency: Sandstorm (*New World Interactive, 2021*); **[4]** Call of Duty: MW2 (*Infinity Ward, 2022*); **[5]** Far Cry 5 (*Ubisoft, 2018*); **[6]** Bioshock Infinite (*Irrational Games, 2013*); **[7]** Grand Theft Auto 5 (*Rockstar, 2013*); **[8]** Rainbow Six: Siege (*Ubisoft, 2015*); **[9]** Team Fortress 2 (*Valve, 2007*); **[10]** Wolfenstein (*Machine Games, 2014*); **[11]** Apex Legends (*Respawn Entertainment, 2019*); **[12]** Atomic Heart (*Mundfish, 2023*); **[13]** Warhammer: Vermintide 2 (*Fatshark, 2018*); **[14]** Back 4 Blood (*Turtle Rock Studios, 2021*); **[15]** Halo 4 (*343 Industries, 2012*); **[16]** Crysis 2 (*Crytek, 2011*); **[17]** Overwatch 2 (*Blizzard Entertainment, 2022*); **[18]** Deathloop (*Arkane Lyon, 2021*); **[19]** Valorant (*Riot Games, 2020*); **[20]** Generation Zero (*Systemic Reaction, 2019*); **[21]** Polygon (*Readaster Studio, 2020*); **[22]** Titanfall 2 (*Respawn Entertainment, 2016*); **[23]** Destiny 2 (*Bungie, 2017*); **[24]** Shatterline (*Frag Lab, 2022*); **[25]** Operation Harsh Doorstep (*Drakeling Labs, 2023*).

Table 3. Additional metadata and statistics of all games present in the study, including (1) 25 games from the newly introduced SMG-25 dataset, (2) Counter Strike: Global Offensive (*Valve, 2012*) data sourced from Pearce & Zhu (2022) and (3) Minecraft (*Mojang, 2011*) data sourced from Baker et al. (2022). Note that the impact of various game modes on data frequency statistics is detailed in Section A.1, whereas the mouse-related metadata is detailed in Section A.2.

Data Collection			Data Frequency (%) by Behavior			Mouse Recording			
Game Name	Game Mode	#Frames	Panning	Navigation	Weapon	Type	$\overline{\delta_x}$	$\overline{\delta_y}$	$t(\delta)$
PUBG	Battle Royale	9768	82	96	24	Free-form	119	28	20
Payday 3	Bank Heist	9056	96	90	43	Free-form	127	32	10
Insurgency	Death Match	10055	91	92	25	Free-form	75	13	20
Call of Duty	Campaign	10309	73	84	35	Auto-center	13	4	2
Far Cry 5	Free Roam	10039	93	80	43	Auto-center	22	4	2
Bioshock	Campaign	10150	90	94	35	Free-form	174	40	20
GTA 5	Free Roam	10046	83	95	19	Free-form	67	13	20
Rainbow Six	Tutorial	10153	64	78	21	Free-form	34	12	20
Team Fortress 2	Death Match	11388	71	90	41	Auto-center	7	5	2
Wolfenstein	Campaign	10400	89	85	34	Free-form	75	18	20
Apex Legends	Battle Royale	11005	82	95	16	Auto-center	9	3	1
Atomic Heart	Campaign	10063	91	90	9	Free-form	72	18	20
Warhammer	Survival	11030	93	98	42	Auto-center	71	9	2
Back 4 Blood	Survival	10054	95	84	55	Free-form	102	31	20
Halo 4	Campaign	10383	91	87	36	Free-form	125	29	20
Crysis 2	Campaign	10430	84	90	32	Free-form	58	13	20
Overwatch 2	Death Match	10888	89	93	45	Auto-center	43	8	2
Deathloop	Campaign	10669	96	90	21	Free-form	228	35	20
Valorant	Death Match	11011	89	90	26	Free-form	131	39	20
Generation Zero	Campaign	10511	86	91	22	Free-form	54	15	20
Polygon	Death Match	10288	81	92	23	Free-form	49	9	20
Titanfall 2	Tutorial	10542	80	87	34	Auto-center	8	3	2
Destiny 2	Campaign	10467	93	90	32	Free-form	63	14	10
Shatterline	Survival	10250	95	91	38	Free-form	128	27	20
Harsh Doorstep	Death Match	10831	76	75	48	Free-form	48	9	20
CS:GO	Death Match	180000	88	83	36	Auto-center	72	18	20
Minecraft	Free Roam	50445	66	69	50	Free-form	98	19	40

be violated by collecting gameplay information. Games that run anti-cheat software in the background block the reading of mouse or keyboard information and were subsequently not included. Special attention was paid to ensure coverage of diverse yet representative games of the FPS genre, not only from a graphical styling or level of photo-realism standpoint, but also from the gameplay mode perspective (refer "Game Mode" column in Table 3). Notably, within the FPS genre, there are certain gameplay modes which yield significant differences in the distribution of actions between games, influencing different player behaviors. For example, games played in the "Battle Royale" game mode usually involve significantly more navigation and environment exploration, and relatively less combat compared to games from the "Death Match" game mode. These differences are highlighted in the "Data Frequency by Behavior" column in Table 3– for example in case of PUBG played in "Battle Royale" mode, only 24% of the recorded data involves video sequences where a weapon was engaged, in contrast to 55% in case of Back 4 Blood played in "Survival" mode.

Collection script: We execute a python script in parallel to playing these games by self (*i.e.*, by a human player and not a bot). In order for our script to interface with these close-sourced commercial games, we bypass the need for access to their respective game engines by instead, interfacing directly with the machine's input devices (mouse and keyboard; no gamepads were used) using python's *Win32API* package as well as the machine's output device (screen) using *OpenCV*. Since our study primarily focuses on analyzing gameplay footage, we implement real-time pause/resume feature in our script to ensure that only gameplay sequences are recorded and non-gameplay sequences such as *loading screens*, *cut-scenes* or *menu overlays* are omitted to the best of our ability.

This script records screen data (RGB frames) and player actions (*i.e.*, mouse movements, mouse clicks and keyboard keypresses) synchronized by timestamps, sampled at roughly 16Hz. Inspired by Pearce & Zhu (2022), our aim was to set the sampling frequency exactly at 16Hz; the actual capture frequency, however, was contingent upon hardware capability at

Table 4. Detailed list with description and statistics of all actions recorded by our script, as well as various animation-related hyper-parameters chosen for pre-processing these actions (detailed explanations in Sections A.2 and B.2).

Script Recording		FPS Genre Behavior		SMG-25	Animation (#frames)		
Device	Action/Key	Description	Category	Freq. (%)	Delay	Length	Cutoff
Mouse	Move Left	Look Left	Panning	62	1	2	2
	Move Right	Look Right	Panning	62	1	2	2
	Move Up	Look Up	Panning	36	1	2	2
	Move Down	Look Down	Panning	40	1	2	2
	Left click	Shoot	Weapon	19	1	2	2
	Right click	Aim	Weapon	15	1	2	2
Keyboard	W	Move Forward	Navigation	77	1	2	2
	A	Strafe Left	Navigation	39	1	2	2
	S	Move Backward	Navigation	11	1	2	2
	D	Strafe Right	Navigation	40	1	2	2
	R	Reload	Weapon	7	3	16	6
	Space	Jump	Navigation	8	1	2	2
	L. Shift (Hold)	Sprint	Navigation	20	1	2	6
	L. Ctrl (Toggle)	Crouch	Navigation	2	1	2	2
	C (Toggle)	Crouch	Navigation	0.1	1	2	2
	1	Switch Item	Weapon	0.4	3	8	6
	2	Switch Item	Weapon	0.4	3	8	6
	3	Switch Item	Weapon	0.1	3	8	6
	F	Interact	-	8	3	5	3

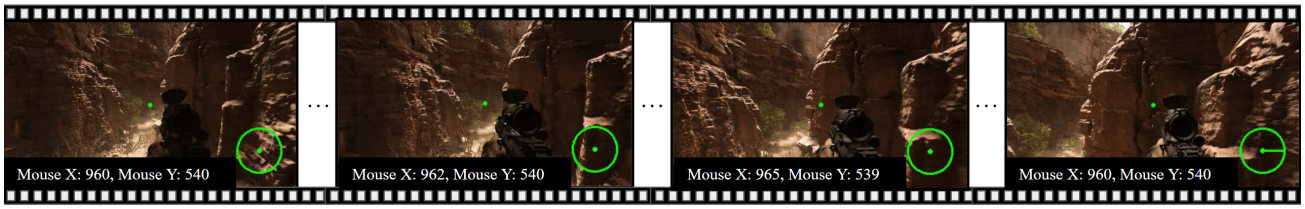
our disposal, adjusting to accommodate the computational resources required for running graphically demanding games. As a result, data of some compute-intensive games were sampled approximately in the range between 12Hz and 16Hz instead. Screen captures of the game are taken at a resolution of 1920×1080, subsequently downsized to 398×224 before saving to disk due to storage limitations. Simultaneously, keyboard and mouse clicks are recorded as binary data (*i.e.*, 1 for key-pressed state and 0 otherwise) and mouse movements are recorded in the form of positional x and y screen coordinates.

The SMG-25 dataset collected using this script contains $\sim$250K samples of synchronized frames and actions across the 25 FPS games, amounting to $\sim$6hrs of gameplay footage. For each game included in the dataset Table 3 provides a number of statistics of the data collected. The detailed list of all actions recorded and the frequency with which they appear in our dataset can be found in Table 4. The specific behavioral actions and their corresponding behavioral category are selected in this study as they act as primitive and general action patterns prevalent in most, if not all, games of the FPS genre. Note that while we use the aforementioned script to collect data for FPS games in this paper, the script can potentially be employed without any modification to collect similar gameplay information for any other game genre. Future enhancements to our data collection pipeline may include sampling of data at higher frequency and integrating additional modalities such as sound.

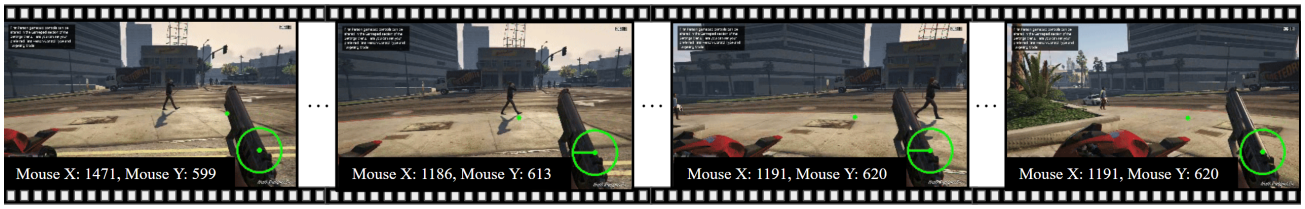
A.2. Data Pre-Processing

In this section we highlight the necessary steps taken to convert raw action keypress values (obtained from our script) to animation labels. Note that these pre-processing steps are pertaining specifically to the FPS genre of games and may not be directly applicable to data collected for other game genres.

Mouse movements: Mouse movements in FPS games are tied to the panning behavior used by a player to look around the game environment. In this step we aim to convert our recorded mouse coordinates to mouse movement labels. Upon examining the raw mouse coordinate values, we identify two predominant methods employed by various FPS games in handling mouse movements— (1) *Auto-Center*: This involves the automatic reset of the pointer position to the center of the screen once the user ceases mouse movement. (2) *Free-Form*: In this mode, the mouse pointer can freely occupy any position within the bounds of the screen resolution. The reader is referred to Figure 7b for a visual representation of each scenario. The "Mouse Recording Type" column of Table 3 identifies the relevant category for each game. For Auto-Center, we introduce a simple check that only player-initiated mouse movements are processed while the game-initiated pointer resets are ignored. No such checks are necessary for the Free-Form since all movements are player-initiated. Next, we use



(a) **Auto-Center** type of mouse movement in *Call of Duty: MW2*, highlighting the player-initiated mouse movement in the first three depicted frames and the system-initiated mouse coordinate reset in the final depicted frame.



(b) **Free-Form** type of mouse movement in *Grand Theft Auto 5*, where all mouse movements are player-initiated and the mouse pointer can take any value within the boundaries of the screen.

Figure 7. Examples of the two types of mouse movements observed predominantly in FPS games. "Mouse X" and "Mouse Y" values displayed at the bottom left of each frame are the raw observations captured by the data collection script, also visualized on the frames by a green dot. The circular radar at the bottom right of each frame indicates the inferred direction of mouse movement. Notice the delay between the player moving the mouse and the associated panning animation occurring on the screen. For instance, in the case of *Call of Duty: MW2*, the Mouse X value exhibits an initial increase over the first three depicted frames, yet the predominant portion of the *pan-right* animation takes effect not until the final depicted frame.

these raw coordinate values to infer a direction for mouse movement, categorized into one of the 8 possible discretized directions: *up, down, left, right, up-left, up-right, down-left* and *down-right*. For this purpose, we perform the following pre-processing operation.

Delta Thresholding: In order to correctly interpret mouse movement, we are required to consider that different games use different mouse sensitivity presets, resulting in significantly different deltas between consecutive timesteps (*i.e.*, difference between successive mouse coordinate values) for different games. For each game, the average values of these deltas in horizontal direction ($\overline{\delta_x}$) and vertical direction ($\overline{\delta_y}$) are presented in the "Mouse Recording" column of Table 3. Because of the nature of FPS games, players of this game genre tend to yield more drastic horizontal mouse movements compared to vertical movements (Pearce & Zhu, 2022). Furthermore, based on these deltas as well as fine-tuned adjustments based on human estimation of mouse sensitivity, we choose a minimum-threshold hyper-parameter ($t(\delta)$) to define significant panning animation visible on screen. These threshold values chosen for each game are provided in Table 3 and are applied to both horizontal and vertical axes of mouse movements for defining the overall panning direction.

Animation Labels from Raw Actions: A significant challenge we are faced with when we attempt to associate every single frame of the game with an appropriate animation label stems from the fact that player behavior is not available directly from the game engine per se but rather it has to be inferred from raw recordings of the machine's input devices. Every keypress action taken by a player using an input device is sent to the game engine, which subsequently decides what animation to trigger on-screen, if any at all. One might claim that some of our labels are approximate "proxy-label" of on-screen animations in the form of keypresses. We are unable to directly address the issue of refining these "proxy-labels" as that would require either access to the game engines for accurate labeling, or significant human-resources for the manual labeling of all frames in our dataset. We consider both of these options as logistically infeasible and prohibiting in terms of cost and effort. We instead attempt to address such challenges through the framework introduced in this paper. We wish our method to be zero-shot transferable and accessible for domain randomization and we design it with such limitations in mind: no access to the games themselves and no resources for manual action labeling.

Not having access to game engines gives rise to another challenge. Each animation associated with different keypresses can exhibit different properties such as delayed onset of the said animation, as well as varying length/duration until the animation is completed on screen. This can lead to poor quality of synchronization between animation frames and their respective action labels inferred via just raw keypresses. For instance, within the genre of FPS games, upon performing a

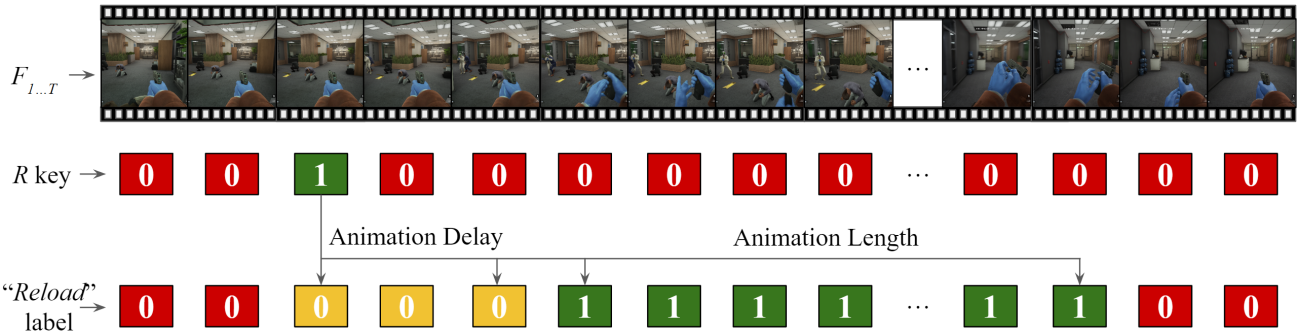


Figure 8. **Label Propagation** example for *reload gun* animation activated by the player pressing the 'R' key. The yellow timesteps emphasize the delay in the onset of the animation visible on screen following the keypress, whereas the green timesteps indicate the complete duration during which the animation persists on screen. The animation *delay* and *length* hyper-parameters selected for each keypress are provided in Table 4.

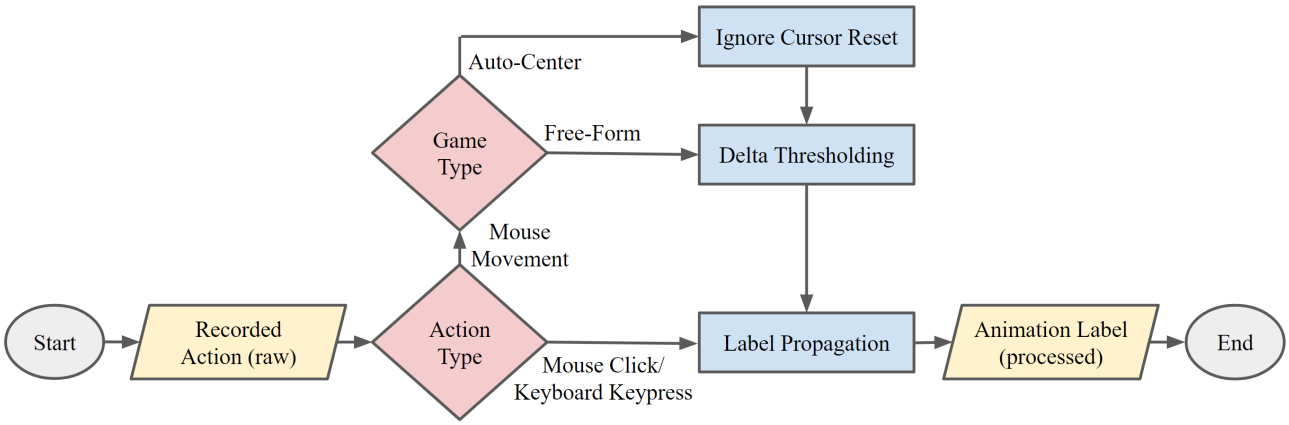


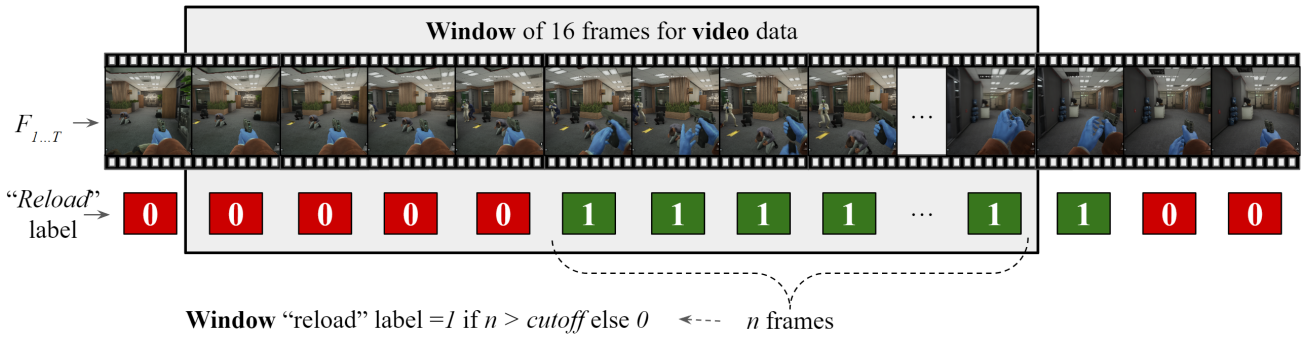
Figure 9. **Data Pre-Processing Pipeline** employed to transform keypress actions into animation labels in the SMG-25 dataset. Note that the additional datasets of CS:GO and Minecraft used in this study have been converted to a data format akin to SMG-25, and, subsequently, have been processed using this pipeline to ensure the uniformity and consistency of all data employed in our experiments.

keypress for actions such as "Move Forward" or "Shoot", the associated animations are relatively more instantaneous and have a shorter duration compared to actions such as "Reload" or "Switch Item" which may last up to several seconds. Refer to the example in Figure 8 for a visual representation of this challenge.

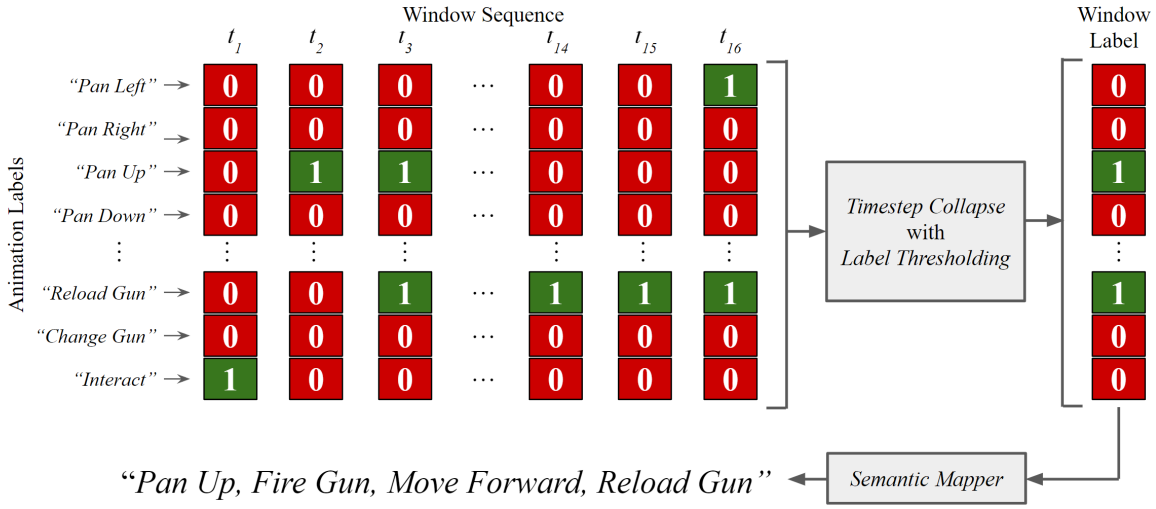
Label Propagation: As a part of our data pre-processing step, we try to address the animation-related challenges with a simple technique we call *label propagation*. This technique applies *shift* and *span* transformations to a keypress label propagating it forward along the temporal dimension. *Shift* looks to account for animation *delay* while *span* tackles challenges related to animation *length*. Figure 8 provides a visual example for this technique. This operation ensures that more frames within our dataset are associated with the correct animation label compared to using raw keypresses as a label. The "Animation" column of Table 4 provides the chosen hyper-parameters that indicate estimates of these animation-related properties (in terms of average number of frames), based on our observations across various FPS genre games. Figure 9 summarizes all the steps of our data pre-processing pipeline.

B. From Time-Steps to Time-Windows

For the purpose of our research on player behavior, as explained in Section 1, we opt to utilize videos instead of images as the input to our vision encoders. Consequently, we outline the necessary steps undertaken to transform the SMG-25 dataset, which includes synchronized data at the timestep level, into a format defined by time interval windows.



(a) **Label Thresholding** example to ensure that sufficient length of animation is present within a window.



(b) **Timestep Collapse** example for converting frame-wise animation labels to a window label in the form of a text caption.

Figure 10. Example showcasing how we combine frames and animation labels from consecutive timesteps to create associated text description of the actions performed in the time window of a video.

B.1. Frames-to-Video Conversion

The approach chosen involves stacking consecutive frames into a video sequence (as shown in Figure 10a), with each sequence representing a window size of 16 frames corresponding to approximately one second of gameplay. This selection is informed by pertinent studies on video backbone models (Wang et al., 2023a;b) and methodologies specific to game analysis (Pearce & Zhu, 2022; Baker et al., 2022). A standard frame transformation involving resizing to dimensions 224×224 is applied, aligning with the specifications suitable for video representation models used in this study. Furthermore, to augment the dataset size during the frames-to-video conversion, a sliding window protocol with a stride of 8 (*i.e.*, half the window size) is employed, resulting in twice the number of unique training samples as compared to using a non-overlapping sliding window protocol. Finally, we take measures to ensure that consecutive frames showcasing temporal discontinuity (*e.g.*, due to any pause or resume of the gameplay; refer to Section A.1), are not consolidated into a unified video sequence.

B.2. Labels-to-Text Conversion

Similar to combining consecutive frames into a video, we combine animation labels from each time-step to a single text description of the overall animation occurring in that respective time window. For this purpose we perform two sequential operations explained here.

Timestep Collapse: Illustrated in Figure 10, the timestep collapse operation is applied to sequences of animation labels within a given window, yielding a singular binary value for each animation label. This binary output indicates the presence or absence of the corresponding animations within the considered time window. Notably, we incorporate an intermediate

Table 5. Lists of alternative phrase descriptions tested for each possible mouse action and keypress. The phrase description options are based on common terminology used within the genre of FPS games and are separated by "|" in the rightmost column of the table. All reported experimental results of this study rely on the phrases appearing in bold.

DEVICE	ACTION/KEY	ALTERNATE PHRASE DESCRIPTIONS
MOUSE	MOVE LEFT	**"Pan Left"** | "Rotate Left" | "Look Left" | "Turn Left"
	MOVE RIGHT	**"Pan Right"** | "Rotate Right" | "Look Right" | "Turn Right"
	MOVE UP	**"Pan Up"** | "Rotate Up" | "Look Up" | "Turn Up"
	MOVE DOWN	**"Pan Down"** | "Rotate Down" | "Look Down" | "Turn Down"
	LEFT CLICK	**"Fire Gun"** | "Discharge Gun" | "Attack" | "Shoot Gun"
	RIGHT CLICK	**"Aim Gun"** | "Gun Scope"
KEYBOARD	W	**"Move Forward"** | "Advance" | "Go Forward"
	A	**"Strafe Left"** | "Move Left" | "Go Left"
	S	**"Move Backward"** | "Retreat" | "Go Back"
	D	**"Strafe Right"** | "Move Right" | "Go Right"
	R	**"Reload Gun"** | "Replace Clip"
	SPACE	**"Jump"** | "Leap"
	L. SHIFT	**"Sprint"** | "Run"
	L. CTRL, C	**"Crouch"** | "Squat"
	1, 2, 3	**"Change Gun"** | "Switch Gun"
	F	**"Interact"**

Label Thresholding operation, as depicted in Figure 10a. This additional step ensures that a sufficient amount of animation is visually perceptible within the window. A *cutoff* hyper-parameter is introduced to establish a minimum threshold, determining whether a time window contains sufficient length of animation. The action-specific *cutoff* values corresponding to each animation label are given in the "Animation" column of Table 4.

Semantic Action Mapper: This operation, illustrated in Figure 10, converts the singular binary values obtained from timestep collapse to a textual description of the animation occurring in the window. Each animation label is mapped to its relevant phrase describing the visual outcome of applying that animation within the game. We combine all obtained phrases for all observed animations with a *comma* delimiter to form our overall text sequence for the window under consideration. This text sequence acts as our label for the associated video for the selected time window. Table 5 lists a number of phrasing alternatives that we experimented with in this study; we did not, however, observe any significant differences in the outcomes of any of our experiments. This suggests that the proposed behavior alignment framework is robust to alternative phrasing as long as the phrasing is tight to the phrased action and the domain under investigation (*i.e.*, FPS games in our study)

Collectively, these conversion steps contribute towards the creation of a dataset that contains synchronized pairs of videos and text description of all actions taken by the player within that video. As covered in Section 3.3 we use this dataset to align similar player behavior across a diverse set of FPS games.

C. Behavior Alignment

Algorithm 2 provides a PyTorch-like pseudocode implementation of the BehAVE framework depicted in Algorithm 1. Table 6 presents extended results from a comprehensive list of all configurations tested in the experiments reported in Section 5.1. Notably, in addition to varying the foundational video and text encoders, we also vary the alignment model architecture (see Section C.1) as well as the loss function employed in the behavior alignment training (see Section C.2.

C.1. Varying the Size of the Alignment Model

To thoroughly evaluate the alignment module of BehAVE we empirically alter the depth of the Multi-Layer Perceptron (MLP) head employed as our projection model for behavior alignment training. Our experimental results, depicted in Table 6, indicate performance enhancements when transitioning from a 1-layered network to a 4-layered network. Extending the depth beyond four layers, however, does not yield notable performance improvement, especially without a significant

Algorithm 2 BehAVE PyTorch-like pseudocode.

```
# semantic_mapper(.):  convert action keypress to behavior text caption
# video_encoder(.):  frozen video foundation model
# text_encoder(.):  frozen text foundation model
# alignment_projector(.):  trainable alignment model

for v, a in dataloader:  # video sequence and corresponding actions
    # convert keypresses to text sequence
    a_c = semantic_mapper(a)

    # obtain foundation encodings and L2 normalize
    z_video = torch.normalize(video_encoder(v), p=2)
    z_caption = torch.normalize(text_encoder(a_c), p=2)

    # train alignment model
    z_align = torch.normalize(alignment_projector(z_video), p=2)
    loss = torch.cosine_embedding_loss(z_align, z_caption)
    loss.backward()
    update(alignment_projector.params) # adam optimizer
```

increase of the number of trainable parameters. The results corresponding to these experiments are reported in the "Proj." column of Table 6 as #-MLP, where # is the number of hidden layers. To enhance the generalization capacity of BehAVE, dropout is applied between the linear layers at a rate of 40%, and Rectified Linear Unit (ReLU) activations are employed. All networks are trained using the *adam* optimizer with a learning rate of $1e^{-4}$ and batches of size 128. The 4-layered MLP emerges as the optimal alignment projection model (with diminishing returns on increasing layers) for subsequent experiments, providing superior results while maintaining a reasonable number of training parameters.

C.2. Varying the Loss Function

In addition to the cosine loss detailed in Section 3.3, we explore alternative loss functions, such as the mean-squared error (MSE) loss defined as follows:

$$\mathcal{L}_{\text{mse}}(z^{\text{align}}, z^{\text{caption}}) = \frac{1}{|z|}\sum_{n=1}^{|z|}(z_n^{\text{align}} - z_n^{\text{caption}})^2 \tag{2}$$

The results reported on Table 6 indicate that alignment with MSE loss fails to achieve performance on par with the cosine loss. Consequently, we exclude this loss function from the remainder of our experiments.

In contrast to the single-data-point focus of Cosine and MSE losses, we explore a pairwise comparison approach using the preference learning paradigm. This approach is motivated by the challenges discussed in Section A.2, where not every keypress corresponds to a visible animation, and vice versa. For example, a mouse *left click* action may not lead to a gun fire animation within the game if the weapon is empty, hence keypresses are only considered as "proxy-labels" for annotating on-screen animations. To accommodate this nuance in our recorded dataset, we adopt a preference learning method that avoids strict assumptions about keypress labels. Thus, the *conditional preference loss* in Equation 3 (with a margin hyper-parameter λ) can be conceptualized as a term that specifically considers the relative proximity of a video embedding to its corresponding text action embedding, when compared to the embedding of a different video. This formulation introduces a more lenient constraint on the alignment process, reducing susceptibility to training noise stemming from the "proxy-labels".

$$\mathcal{L}_{\text{pref}}(z_i^{\text{align}}, z_j^{\text{align}} \mid z_i^{\text{caption}}) = \max\left(0, \mathcal{L}_{\cos}(z_i^{\text{align}}, z_i^{\text{caption}}) - \mathcal{L}_{\cos}(z_j^{\text{align}}, z_i^{\text{caption}}) + \lambda\right) \tag{3}$$

This loss focuses on the relative closeness between specific video-text pairs, providing a more robust training mechanism

Table 6. Comparison of various alignment training configurations, including differences in pretrained video encoders, the number of layers in the projection model, action/text encoders, and the loss function. The resulting dimension of the shared space is denoted by the "Dim" column. Silhouette scores on the SMG-25 test set are presented, with average and standard deviation values calculated over 5 runs, each utilizing a distinct, randomly selected set of games. Higher scores are preferable for behavior-based cluster labels (indicated by ↑), while lower scores are optimal for game labels (indicated by ↓). The best-performing configuration in each category is highlighted in bold.

Training Configuration					Behavior Labels			Game
Loss	Video Enc.	Proj.	Action Encoding	Dim.	Panning ↑	Navigation ↑	Weapon ↑	Label ↓
-	VideoMAE	-	-	768	$0.08_{\pm 0.00}$	$0.13_{\pm 0.00}$	$0.03_{\pm 0.00}$	$0.10_{\pm 0.00}$
	MVD			768	$0.14_{\pm 0.00}$	$0.09_{\pm 0.00}$	$0.01_{\pm 0.00}$	$-0.05_{\pm 0.00}$
	I3D			512	$0.03_{\pm 0.00}$	$0.04_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.07_{\pm 0.00}$
Cosine	VideoMAE	1-MLP	CLIP	512	$0.27_{\pm 0.00}$	$0.35_{\pm 0.00}$	$0.15_{\pm 0.00}$	$-0.12_{\pm 0.00}$
		2-MLP			$0.36_{\pm 0.01}$	$0.46_{\pm 0.00}$	$0.33_{\pm 0.00}$	$-0.21_{\pm 0.01}$
		3-MLP			$0.37_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.36_{\pm 0.00}$	$-0.22_{\pm 0.01}$
		4-MLP			$0.40_{\pm 0.01}$	$0.49_{\pm 0.01}$	$0.35_{\pm 0.00}$	$-0.20_{\pm 0.01}$
Pref. MSE	VideoMAE	4-MLP	CLIP	512	$0.19_{\pm 0.02}$	$0.26_{\pm 0.04}$	$0.23_{\pm 0.02}$	$-0.14_{\pm 0.01}$
					$0.38_{\pm 0.03}$	$0.48_{\pm 0.03}$	$\mathbf{0.38_{\pm 0.02}}$	$-0.17_{\pm 0.03}$
Cosine	VideoMAE	4-MLP	Binary	16	$0.35_{\pm 0.00}$	$0.48_{\pm 0.01}$	$0.32_{\pm 0.01}$	$-0.16_{\pm 0.00}$
	MVD				$0.43_{\pm 0.02}$	$0.44_{\pm 0.03}$	$0.09_{\pm 0.01}$	$-0.24_{\pm 0.02}$
	I3D				$0.38_{\pm 0.01}$	$0.45_{\pm 0.02}$	$0.22_{\pm 0.01}$	$-0.21_{\pm 0.01}$
Cosine	VideoMAE	4-MLP	AllMiniLM-L12-V2 (Reimers & Gurevych, 2019)	384	$0.41_{\pm 0.03}$	$0.48_{\pm 0.04}$	$0.35_{\pm 0.02}$	$-0.17_{\pm 0.03}$
			AllMiniLM-L6-V2 (Reimers & Gurevych, 2019)	384	$0.42_{\pm 0.02}$	$0.49_{\pm 0.05}$	$0.31_{\pm 0.01}$	$-0.17_{\pm 0.03}$
			Mobile BERT (Sun et al., 2020)	512	$0.27_{\pm 0.06}$	$0.49_{\pm 0.05}$	$0.31_{\pm 0.01}$	$-0.17_{\pm 0.03}$
			BERT (Devlin et al., 2018)	768	$0.48_{\pm 0.06}$	$\mathbf{0.60_{\pm 0.04}}$	$0.23_{\pm 0.05}$	$-0.24_{\pm 0.07}$
			Flava (Singh et al., 2022)	768	$0.47_{\pm 0.02}$	$0.51_{\pm 0.02}$	$0.22_{\pm 0.03}$	$-0.24_{\pm 0.05}$
			Data2Vec (Baevski et al., 2022)	768	$0.47_{\pm 0.05}$	$0.47_{\pm 0.05}$	$0.20_{\pm 0.03}$	$-0.23_{\pm 0.04}$
			GPT (Radford et al., 2018)	768	$0.46_{\pm 0.04}$	$0.48_{\pm 0.03}$	$0.30_{\pm 0.04}$	$-0.18_{\pm 0.05}$
			GPT-2 (Radford et al., 2019)	768	$0.51_{\pm 0.02}$	$0.58_{\pm 0.05}$	$0.20_{\pm 0.05}$	$\mathbf{-0.29_{\pm 0.06}}$
			All-MPNet-Base-V2 (Reimers & Gurevych, 2019)	768	$0.45_{\pm 0.01}$	$0.51_{\pm 0.03}$	$0.26_{\pm 0.02}$	$-0.22_{\pm 0.03}$
			MPNet (Song et al., 2020)	768	$0.48_{\pm 0.04}$	$0.52_{\pm 0.06}$	$0.18_{\pm 0.04}$	$-0.25_{\pm 0.06}$
			ALIGN (Jia et al., 2021)	768	$\mathbf{0.52_{\pm 0.03}}$	$0.56_{\pm 0.03}$	$0.19_{\pm 0.03}$	$-0.26_{\pm 0.04}$
			GPT Neo (Black et al., 2021)	2048	$0.50_{\pm 0.02}$	$0.52_{\pm 0.04}$	$0.25_{\pm 0.03}$	$-0.21_{\pm 0.03}$

less sensitive to inaccuracies in the labeling process. However, as shown in Table 6, alignment with such a preference loss did not yield significant benefits compared to the Cosine loss, leading us to discard this research paradigm for the current study.

D. Behavior Classification

Section D.1 provides results from additional video encoder configurations tested on the downstream task presented in Section 4.3, while Section D.2 presents our preliminary efforts towards crafting a more fine-trained action prediction system instead of higher-level behavior categories.

D.1. Varying the Video Encoder

Table 5 provides supplementary results pertaining to the behavior classification transferability from CS:GO to unseen games from the SMG-25 test set. Notably, we illustrate the advantages of zero-shot transferability when employing alternative video encoders to VideoMAE, including the I3D or the MVD. Across all behavior categories, we notice up to 22% improvement on average for the I3D encoder and up to 17.5% improvement on average for the MVD encoder. Neither of these alternate video encoders attain absolute performance levels comparable to VideoMAE. However, the comparative analysis between the representations of pretrained models and our proposed aligned embeddings underscores the robustness and effectiveness of our alignment framework regardless of the choice of the video encoder.

D.2. From Behavior Classification to Inverse Dynamics

In the context of video games, the actions executed by a player play a pivotal role in generating on-screen visuals, determined by the transition dynamics within the game world as defined by its underlying game engine. This section presents our preliminary efforts to extend the generalized FPS behavior classification task towards the development of a more intricate generalized FPS Inverse Dynamics Model (IDM). In contrast to predicting overall player behavior, our focus shifts towards predicting a specific set of actions enacted by the player, manifesting as raw keypresses.

Table 7. Transferability of behavior classification analyzed via classification accuracy for behavior categories. The classifiers are trained on source game CS:GO and transferred to target domains of FPS games in SMG-25 test set. The transferability score is measured by percentage difference in accuracy between using aligned versus foundation encodings, averaged over 5 independent runs.

Encoder Methods (Video - Action)	Behavior Category	CS:GO (Source) Unaligned (%)	CS:GO (Source) Aligned (%)	CS:GO (Source) %-Diff.	SMG-25 (Target) Unaligned (%)	SMG-25 (Target) Aligned (%)	SMG-25 (Target) %-Diff.
VideoMAEv2 - CLIP	Panning	$83.32_{\pm 0.49}$	$75.93_{\pm 2.24}$	-8.88	$67.22_{\pm 1.74}$	$72.04_{\pm 2.57}$	+7.29
	Navigation	$94.15_{\pm 0.51}$	$85.65_{\pm 2.96}$	-9.03	$78.28_{\pm 1.40}$	$80.86_{\pm 1.64}$	+3.33
	Weapon	$92.68_{\pm 0.66}$	$86.65_{\pm 0.41}$	-6.51	$66.14_{\pm 2.11}$	$80.86_{\pm 0.69}$	+22.34
MVD - CLIP	Panning	$68.95_{\pm 3.15}$	$69.89_{\pm 2.94}$	+1.68	$68.79_{\pm 3.14}$	$70.34_{\pm 2.23}$	+2.42
	Navigation	$63.86_{\pm 6.87}$	$62.36_{\pm 2.64}$	-1.32	$53.73_{\pm 2.92}$	$62.91_{\pm 5.89}$	+17.55
	Weapon	$71.99_{\pm 1.86}$	$61.15_{\pm 1.20}$	-15.00	$57.96_{\pm 2.24}$	$58.77_{\pm 1.56}$	+1.54
I3D - CLIP	Panning	$79.94_{\pm 1.36}$	$72.11_{\pm 1.75}$	-9.53	$60.30_{\pm 3.90}$	$68.61_{\pm 1.29}$	+14.25
	Navigation	$83.21_{\pm 3.54}$	$73.64_{\pm 1.23}$	-11.36	$59.45_{\pm 3.54}$	$72.57_{\pm 2.89}$	+22.09
	Weapon	$87.79_{\pm 0.90}$	$72.98_{\pm 2.26}$	-16.87	$62.78_{\pm 2.66}$	$68.26_{\pm 2.06}$	+8.80

Table 8. Transferability of behavior classification from CS;GO (source) to SMG-25 test set (target). Actions having less than 30% data points present in the training dataset are highlighted in gray.

Action/Key	CS:GO (IDM Source) Freq. (%)	CS:GO (IDM Source) Unaligned (%)	CS:GO (IDM Source) Aligned (%)	CS:GO (IDM Source) %-Diff.	SMG-25 (IDM Target) Unaligned (%)	SMG-25 (IDM Target) Aligned (%)	SMG-25 (IDM Target) %-Diff.
Move Forward	69.83	90.26	85.01	-5.82	81.91	84.27	+2.88
Pan Right	44.85	72.98	65.3	-10.52	62.03	65.07	+4.90
Strafe Right	43.67	71.15	65.08	-8.53	54.52	61.11	+12.09
Pan Left	43.07	73.55	64.24	-12.66	58.23	65.83	+13.05
Strafe Left	41.37	75.09	67.11	-10.63	55.32	60.76	+9.83
Fire Gun	30.19	94.36	89.51	-5.14	77.53	88.09	+13.62
Pan Down	14.10	76.8	72.94	-5.03	58.66	56.12	-4.33
Pan Up	12.95	73.26	68.98	-5.84	59.5	52.72	-11.39
Move Backward	10.82	69.16	62.32	-9.89	55.47	66.79	+20.41
Reload Gun	4.77	92.86	58.83	-36.65	51.91	53.75	+3.54
Jump	2.20	86.69	71.6	-17.41	66.57	60.38	-9.30
Change Gun	1.58	100	100	0	60.32	55.56	-7.89
Crouch	0.78	81.08	71.62	-11.67	51.8	46.4	-10.42
Aim Gun	0.00	-	-	-	-	-	-
Sprint	0.00	-	-	-	-	-	-
Interact	0.00	-	-	-	-	-	-

It is noteworthy that the pretrained video backbone models employed herein are not explicitly designed to discern low-level patterns within video data. Their primary function, instead, is to identify high-level patterns, such as behavior classes. Nonetheless, we endeavor to assess the extent of our capabilities by utilizing frozen pretrained models without additional fine-tuning specific to our dataset or problem domain, aiming to recover precise actions through keypress predictions. This presents a more formidable challenge, as the objective involves identifying finer-grained actions as opposed to broader behavior categories. Consequently, this section delineates two distinct approaches employed for a per-key IDM.

Marginal Distribution of Actions: The initial approach involves predicting each keypress through an independent classification head. In other words, the IDM outputs a marginal distribution for each action. Our model processes a single window and utilizes video input to predict the fine-grained label, denoting the presence or absence of each keypress. As this model forecasts primitive actions based on preceding and succeeding frames, it can be conceptualized as an inverse dynamics model. Table 8 underscores the improved transferability capacity of BehAVE achieved in 8 out of 13 keypress actions. The outcomes, however, remain inconclusive for infrequently occurring keypresses, where data is limited (refer to "Freq." column in Table 8). If we focus on actions characterized by an occurrence frequency over 30%, we observe improved transferability across all six keypress actions (see the white background rows of Table 8). The lack of adequate data for learning to discern certain keypresses and actions, coupled with resource constraints hindering fine-tuning of large video backbone models are the limiting factors for higher performance in less frequent actions. We can only expect that larger datasets containing representative samples for all actions (as in Baker et al. (2022)) and appropriate fine-tuning of models to the domain under consideration would yield IDMs of high accuracy and transferability capacity via BehAVE.

Joint Distribution of Actions: The second approach we employ involves the prediction of all possible actions within a given temporal window using an autoregressive transformer model. In essence, the IDM outputs a joint distribution of all potential actions. To implement this approach, we draw inspiration from the work of (Li et al., 2023) and leverage their methodology to decode our aligned video embeddings into textual captions that serve as a semantic representation of the on-screen animation resulting from all actions executed within the specified window. The objective is to predict these captions to match the window animation labels (as shown in Figure 10). To achieve this, we train a GPT 2 (Radford et al., 2019) transformer to reconstruct the target window label, conditioned on the aligned latent. A notable difference in the text encoding strategy of our transformer involves treating each action label as a single, distinct token to ensure that a label for a single action remains atomic. This type of encoding ensures that our model perceives each individual keypress as a binary label, thereby eliminating the need to learn the vocabulary associated with our action text encoding. While we believe this approach has potential, we observe weak performance for this model likely due to insufficient training data for every possible action, and further investigation and improvements are an ongoing effort.

Our preliminary findings demonstrate the potential of BehAVE to generate a transferable zero-shot IDM for FPS games; however, further data and computational effort is likely required to enhance its performance. Future research endeavors could focus on fine-tuning video backbone models rather than limiting the training to frozen models. Additionally, a prospective avenue for subsequent investigation could involve treating temporal windows as recurrent entities, as opposed to the independent and identically distributed predictions examined in the current study.