InvPatch: Prefix-Based Conditional Generation for Inverse Dynamics

Nemanja Rašajski[1], Konstantinos Makantasis[2], Antonios Liapis[1], Georgios N. Yannakakis[1]

***Abstract*— Inverse Dynamics Models (IDMs) attempt to predict the actions that cause observable changes in a scene. Current methods for building accurate IDMs either rely on rule-based systems that exploit video metadata or require large-scale training over thousands of video hours. These approaches are inevitably limited to a single domain due to the difficulty of acquiring metadata or sufficient training data across different domains. In response to these challenges, this study draws inspiration from data-efficient video captioning methods, specifically prefix-based conditional generation. This approach maps visual features into prefix tokens that condition the action-prediction process. We introduce InvPatch, a framework that builds on prefix-based conditional generation and extends it by adding learned visual-representation compression. In InvPatch, attention-based patch selection and pooling are applied to features extracted from a ViT backbone, reducing the conditioning input from a set of frame-by-frame features to a single vector. We evaluate our framework across two diverse settings: 3D third-person real world (KIT Bimanual Actions) and 2D synthetic (MUGEN). Our method achieves 97.21% accuracy on MUGEN and surpasses state-of-the-art results on KIT Bimanual Actions. Our sensitivity analysis highlights the data efficiency of this approach, as InvPatch maintains comparable performance even when trained with 30% less data. Additionally, our runtime analysis demonstrates the computational efficiency of InvPatch, which requires fewer trainable parameters and performs fewer FLOPs compared to other methods.**

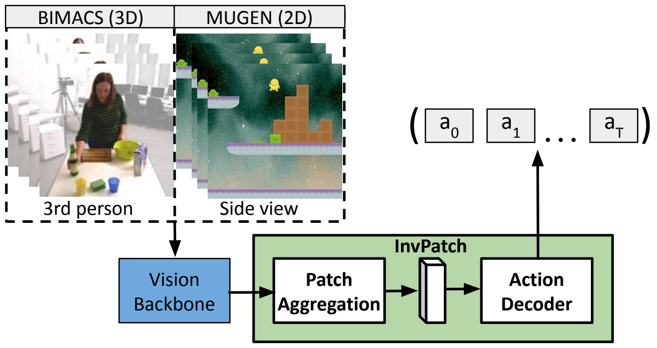


Fig. 1. High level overview of InvPatch: we learn a global video embedding and use it to condition the action decoder to produce viable sequences of actions $(a_0, a_1, ..., a_T)$ across T timesteps.

I. Introduction

An Inverse Dynamics Model (IDM) is a mathematical model that infers causal actions from observed outcomes. IDMs have been employed across multiple fields to solve tasks such as robotic control [1], musculoskeletal load estimation in biomechanics [2], and sequential action inference from gameplay footage in video games [3] to enable supervised learning from unlabeled demonstrations [4]. One approach to train IDMs is to use hand-crafted rules built from high-level features [3]. Alternatively, IDMs can be trained to infer actions directly from videos, but doing so requires large labeled datasets and substantial computational resources [5]. This study builds on prior work on learning from videos, aiming to reduce both the computational resources and the amount of data required.

With these goals in mind, we introduce InvPatch, a novel method for inverse dynamics modeling. InvPatch combines prefix-based conditional generation with a patch-level representation aggregation strategy for integrating spatiotemporal information from frozen pre-trained Vision Transformer (ViT) [6] backbones. Prefix-based conditional generation uses a vision model to map videos into prefix embeddings, that guide the decoder in autoregressively predicting the sequence of actions that generated the video. This method was initially introduced for data-efficient image and video captioning [7], [8], a task closely related to modeling inverse dynamics over videos. Our InvPatch framework builds on this line of work by introducing an aggregation layer (see Fig. 1) that processes patch-level representations. This layer identifies the most informative spatial regions over time via attention-based selection and aggregates them into a single global embedding. This embedding is used as the prefix to condition the prediction of action sequences. The single embedding strategy is informed by prior work demonstrating both the substantial redundancy present in videos, especially across consecutive frames [9], and the favorable performance and compute trade-offs achieved when multiple input tokens are represented by a single vector [10]. Additionally, representing the video as a single vector enables the use of video-native backbones—such as VideoMAE [9]—which produce strong representations but do not output frame-level features.

We evaluate the effectivenes of InvPatch across two action prediction benchmarks: MUGEN [11] and KIT Bimanual Actions [12]. These benchmarks offer both real-world and simulated environments viewed from 2D and 3D perspectives. Our IDMs achieve up to 97.21% accuracy on MUGEN and improve state-of-the-art performance on KIT Bimanual Actions by 3%, demonstrating the effectiveness of our approach. In addition, our proposed spatio-temporal aggregation method outperforms all baseline methods, in-

This work has been supported by Project GenV (REP-2025-032) funded by Xjenza Malta.

Code: https://github.com/nrasajski/InvPatch

[1]Nemanja Rašajski, Antonios Liapis and Georgios N Yannakakis are with the Institute of Digital Games, University of Malta. Contact: `nemanja.rasajski@um.edu.mt`

[2]Konstantinos Makantasis is with the Department of AI, University of Malta

cluding mean, max, and GeM pooling [13], attention-based pooling [14], NetVLAD [15], and LSTM [16]. We also show that InvPatch achieves comparable performance when trained with 30% less data, demonstrating its data efficiency, and confirm its computational efficiency through runtime analysis. Finally, we examine how many frames InvPatch can compress into a single representation; the results indicate promising scalability in both frame compression and long-horizon action prediction.

The key contributions of this paper are as follows. First, to the best of our knowledge, this is the first work to introduce prefix-based conditional generation for data-efficient inverse dynamics. Second, InvPatch provides an efficient method for aggregating information from pre-trained backbones; this design outperforms common baselines, achieves near-optimal accuracy on MUGEN, and improves the state of the art on KIT Bimanual Actions. Finally, we evaluate the sensitivity of InvPatch to data availability and the number of observed frames, and demonstrate robustness across both dimensions.

II. Related Work

Inverse Dynamics Models predict the action required to transition from one state to the next given two consecutive states [17]. IDMs are widely used in biomechanics and sequential decision-making. In biomechanics, they estimate joint forces and torques during human movement by combining motion capture data with anthropometric models, enabling detailed analysis of muscle and joint contributions [2]. IDMs have also been applied in imitation learning, such as transferring actions from simulation to real-world tasks [18] or conditioning agents on task goals in visual environments to predict action sequences [19]. They can reduce action mismatches between expert demonstrations and agent behaviour [20], and convert sequences of states into actions to replicate trajectories [21], [22]. IDMs are further used to expand training data: rule-based models have been built for games like *Counter-Strike: GO* (Valve, 2012) [3], while large-scale deep IDMs supported video-based pretraining in environments such as *Minecraft* (Mojang, 2011) [5]. While this study, like [5], [22], learns action sequences from video, it differs in two key ways: first, we prioritize data efficiency and compute-plausible training [5], and second, we introduce prefix-based conditional generation as a distinct approach to action-sequence prediction [22].

Image and Video Captioning models generate natural-language descriptions of visual inputs. Early systems paired CNN feature extractors with RNN caption generators [23], [24], later replaced by transformer-based models [25], [26]. Recent work uses large-scale vision–language pretraining on paired and unpaired data to improve accuracy and generalisation [27]–[30]. Multimodal models like GPT-4V [31] also enable automatic curation of high-quality video-caption datasets [32]. Alternatively, prefix-based conditional generation methods guide language models with visual features, enabling efficient captioning for images (ClipCap [7]) and for both images and videos (DeCap [8]). While ClipCap focuses solely on image captioning, DeCap employed image captioning to video using a single global vector from mean-pooled frame features. We extend prefix-based conditional generation models like ClipCap and DeCap with a learned patch-level feature pooling layer, producing richer representations that boost performance.

Global Video Representations aim to compress a sequence of local descriptors extracted at the frame or patch level into a single embedding that captures relevant information such as motion dynamics. Simple, lightweight approaches that do not require training include mean pooling, which averages features over time to produce stable summaries [33], and max pooling, which emphasizes the most discriminative features [34]. Generalized mean (GeM) pooling [13] generalizes these methods by learning an exponent that interpolates between mean-like and max-like behaviour, and has been used to aggregate frame-level features into a compact global embedding [35]. Beyond fixed pooling, structured encoding methods such as Vector of Locally Aggregated Descriptors (VLAD) [15], [36] can be extended to video by learning spatio-temporal codebooks. Local features are assigned to learned codewords, and residuals are aggregated across space and time into a compact video-level descriptor [37]. Recurrent architectures like LSTMs sequentially integrate temporal information to capture long-range dependencies [38]. Attention-based pooling assigns adaptive weights to frames or patches to highlight the most relevant content [14]. Our aggregation layer, which produces global embeddings for decoder conditioning, relies on attention-based pooling. To improve computational efficiency, we apply patch merging [39] and use a computationally efficient variant of spatio-temporal attention [40].

III. Method

Figure 2 provides a high-level overview of the InvPatch training procedure. We formalize this process below and describe our architectural and training choices.

A. Problem Statement

Inverse dynamics from pixels can be formulated as learning a mapping $f : X \to S$, where X is the space of observed image sequences or videos and S is the space of corresponding action sequences. An observation $x \in X$ is a sequence of frames $x = \langle I_1, \ldots, I_T \rangle$, and an action sequence $s \in S$ is given by $s = \langle a_1, \ldots, a_L \rangle$, where each action $a_t \in A$, and A is the actions space (i.e. a finite discrete set of possible actions). In the general case $T \neq L$, since multiple actions may occur within a single timestep. For notation simplicity, however, we assume that exactly one action is associated with each frame, so $T = L$, and we treat both X and S as spaces of sequences of length T. Given a dataset D of N paired videos and action sequences $\{x_i, s_i\}_{i=0}^{N}$, where $x_i = \langle I_{i,1}, \ldots, I_{i,T} \rangle$ and $s_i = \langle a_{i,1}, \ldots, a_{i,T} \rangle$ the goal is to learn a function f that maps an unseen video x_i to its corresponding action sequence $s_i = f(x_i)$. We train f by maximizing the conditional log-likelihood of the action sequence given the video. Formally, the training objective is:

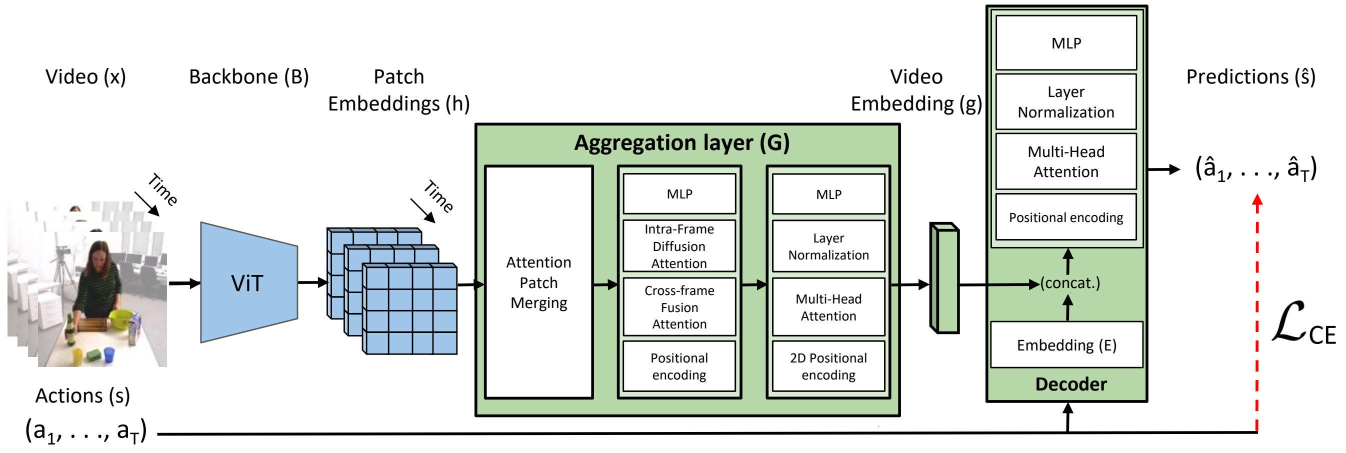


Fig. 2. Overview of the training pipeline including a high-level architectural description of InvPatch. Blue components indicate evaluation mode, while green components represent trainable InvPatch modules. We first extract patch-level representations from a ViT backbone, then aggregate them into a single vector. This aggregated vector is prepended to the action embeddings and passed to the transformer decoder. Finally, we compute a cross-entropy loss between the predicted and ground-truth actions.

$$f(x_i) = \max_{\theta} \sum_{i=1}^{N} \log p_\theta(a_{i,1}, ..., a_{i,T} | x_i) \tag{1}$$

where θ denotes the model's trainable parameters. Following ClipCap [7] and DeCap [8], we utilise the condition as a prefix to the action sequence s_i, and apply an autoregressive model to predict the next token. Thus Eq. (1) can be rewritten to describe our training objective as follows:

$$\begin{gathered} f(x_i) = \max_{\theta} \sum_{i=1}^{N} \sum_{j=1}^{T} \log p_\theta(a_{i,j} | z_{i,j}), \\ z_{i,j} = [x_i; a_{i,1} : a_{i,j-1}] \end{gathered} \tag{2}$$

B. Aggregation layer

Prior work [9] shows that videos contain substantial pixel-space redundancy across consecutive frames. We posit that this redundancy further extends to the intra-frame region level, where many areas provide minimal information for inverse dynamics prediction. To address this, we propose filtering redundancy in representation space to produce an informative single vector representation. The input to our aggregation layer consists of patch-level embeddings from a frozen ViT backbone. We first apply attention-based patch merging [39]. For each frame I_t, let $H_t = \langle h_1, \ldots, h_C \rangle$ denote the sequence of patch embeddings. We group image patches into contiguous 2×2 local windows and perform attention-based aggregation. This yields merged patch embeddings $\hat{H}_t$, reducing the sequence length by a factor of four. This hierarchical merging serves two key purposes: a) eliminating low-information regions, and b) lowering computational cost since self-attention scales quadratically with token count. This efficiency gain enables training our IDMs on a single RTX 3090 GPU. We then apply a Cross-frame Communication Transformer [40] that uses two components: Cross-frame Fusion Attention (CFA) and Intra-frame Diffusion Attention (IFA). Together, these provide a more computationally efficient variant of spatio-temporal attention. For each patch sequence $\hat{H}$, we prepend a learnable embedding fg_t, which serves as the representation of frame I_t. Information is exchanged across frames through an auxiliary message token m_t, generated online as a linear transformation of fg_t and later discarded. CFA operates on all message tokens $M = \langle m_1, \ldots, m_T \rangle$ to capture the global spatio-temporal dependencies of the input video as follows:

$$\hat{M} = M + \mathrm{CFA}(\mathrm{LN}(M)) \tag{3}$$

where $\hat{M} = \langle \hat{m}_1, \ldots, \hat{m}_T \rangle$, and LN denotes layer normalization. Next, we concatenate ($\frown$) each fg_t with its corresponding $\hat{m}_t$ and apply IFA. This allows fg_t to incorporate the temporal cues aggregated in $\hat{m}_t$ while retaining frame-specific semantics. Afterward, the message token is discarded, and only the frame token is kept. Formally:

$$\begin{gathered} \mathrm{fg}'_t \frown m'_t = \mathrm{fg}_t \frown \hat{m}_t + \mathrm{IFA}(\mathrm{LN}(\mathrm{fg}_t \frown \hat{m}_t)) \\ \mathrm{fg}_t = \mathrm{fg}'_t + \mathrm{FFN}(\mathrm{LN}(\mathrm{fg}'_t)) \end{gathered} \tag{4}$$

where FFN denotes a feed-forward network. Finally, a simplified ViT with 2D positional encodings processes the sequence of frame features $\langle \mathrm{fg}_1, \ldots, \mathrm{fg}_T \rangle$. This module treats the video as a spatial grid, interpreting motion as spatial variation [32], and outputs the final global embedding g.

C. Decoder

To predict action sequences conditioned on visual representations, we adopt a decoder-only transformer following [8]. The decoder is equipped with an embedding layer E, implemented as a lookup table of learned vector representations indexed by actions, which is standard in transformer architectures. This layer maps action sequences into the embedding space used by the model. The size of the lookup table matches the size of the action space A. The embedding vectors are trained jointly with the decoder and the visual information aggregator networks.

D. Training procedure

Visual patch features are extracted for each video x_i using a backbone network B operating in evaluation mode with frozen weights. The resulting patch embeddings are then processed by our trainable aggregation layer denoted as G, which maps them to a single vector representation $g_i = G(B(x_i))$. Subsequently, we map action sequence s_i to embedding space using the embedding layer E, $\langle e_1, ..., e_T \rangle = E(\langle a_{i,1}, \ldots, a_{i,T} \rangle)$. Finally, we concatenate the prefix embedding with action embeddings obtained from the decoder as follows $z_i = [g_i; e_1 : e_T]$. During training we feed the decoder network with the concatenated embeddings $\{z_i\}_{i=0}^{N}$. Our training objective is predicting the action sequence $\hat{s}$ conditioned on the prefix in an autoregressive fashion. For that purpose, we train the decoder and aggregation layer based on the training objective in Eq. (2) by minimising the cross entropy loss $\mathcal{L}_{CE}$ between the generated and ground truth actions.

IV. Experiments

We design our experiments to assess the action prediction performance of InvPatch, evaluate the impact of our proposed spatio-temporal aggregation approach, and validate our goals of strong data and computational efficiency.

A. Datasets

MUGEN [11] is a synthetic dataset of 375K video clips generated from the CoinRun platformer [41]. Each clip is 3.2s long at 30Hz and is accompanied by a JSON file containing frame-level metadata: position on the map (x, y) and positional change between consecutive frames $(\Delta x, \Delta y)$. Seven actions are possible: inaction, move left, move right, jump, descend, jump left, and jump right. We randomly sample a subset of 2,500 clips (2h20m) and extract action sequences from the metadata. We use this dataset as a controlled 2D setting that differs from natural images, challenging visual backbones trained on real-world data.

KIT Bimanual Actions Dataset (BIMACS) [12] contains 540 recordings from 6 subjects performing 9 tasks, each repeated 10 times. In the BIMACS dataset, 14 actions can be performed by the left and right hands: idle, approach, retreat, lift, place, hold, pour, cut, hammer, saw, stir, screw, drink, and wipe. Recordings were made in two settings—kitchen and workshop—and total 2h18m. Each clip, averaging around 15 seconds, shows a single person performing a complex everyday task. The dataset includes frame-wise action labels for all 14 actions. We use this dataset as a real-world benchmark with a third-person camera perspective.

B. Experimental Setup

Inverse Dynamics. We evaluate the overall performance of action prediction and assess the impact of the proposed aggregation layer. Following prior work [3], [9], we use input sequences of 16 frames. For MUGEN, sequences are grouped by video ID, while for BIMACS we group the data by subject, task, and repetition to ensure a strict separation between training and test sets. We perform 5-fold cross-validation (CV). Action sequences have a fixed length: 16 for MUGEN, with one action per frame, and 32 for BIMACS, with two actions per frame (left and right hand).

Sensitivity Analysis. We conduct two sensitivity analyses: (a) training set size, where we reduce the number of training samples to assess data efficiency; and (b) embedded frame count, to evaluate how the number of frames compressed into a global embedding—and the resulting action sequence length—affects performance. For (a), we use 16 frames and vary the training set size to 80%, 50%, 20%, and 10% of the data. For (b), we vary the number of frames (4, 8, 16, 32) while keeping the global embedding size fixed, using an 80/20 train–test split. Both experiments are run three times with randomly sampled train–test splits.

Evaluation Metrics. We adopt metrics commonly used in inverse dynamics modeling and sequence generation. Specifically, we use: (a) action classification accuracy and precision [42]; (b) BLEU@4 [43], which measures n-gram precision between generated and reference sequences; and (c) METEOR [44], which computes a recall-weighted harmonic mean of unigram precision and recall, following [8].

Baselines. For comparison with prior prefix-based conditional generation methods, we test InvPatch against a ClipCap-inspired model [7] conditioned on multiple frame-level embeddings (the backbone's pre-trained CLS token) and a DeCap-inspired model [8] conditioned on a single embedding obtained via simple pooling (mean or max). To contrast frame and patch level aggregation strategies, we apply several aggregation methods to frame-level (CLS token) features: GeM pooling [13], LSTMs, attention-based pooling [14], and NetVLAD [15] for quantization-based pooling. The same methods are employed to consider patch-level features, with LSTMs applied first across the temporal dimension of each patch, followed by mean pooling across the spatial dimension.

Implementation Details. Our method is implemented in PyTorch [45], and all experiments run on a single NVIDIA RTX 3090 GPU, demonstrating computational feasibility. Training uses the AdamW optimiser [46] for 90 epochs with a batch size of 64 and a learning rate of 1e-4, warmup over the first 10% of epochs, and cosine decay thereafter. We use two vision encoders, VideoMAE [9] and DINOv2 [47], both in the ViT-B variant to ensure computational feasibility, and widely adopted for video representation learning [48], [49]. The decoder is a 4-layer GPT-2 [50] model with embedding size 768.

V. Results

Inverse Dynamics. Table I reports results on the MUGEN dataset. As this is the first use of MUGEN for IDMs, we include two simple baselines—random (row 22) and majority action (row 23)—for context; all tested configurations outperform them. Aggregating features into a single global video embedding does not degrade performance; in fact, global embeddings outperform the per-frame configuration (row 1) by 10%, even when using frame-level features (row

TABLE I

MUGEN DATASET: INVERSE DYNAMICS RESULTS MEAN ± 95% CI OVER 5-FOLD CV; HIGHER IS BETTER. **BOLD** MARKS THE BEST PER METRIC AND BACKBONE, AND **BOLD AND UNDERLINED** THE OVERALL BEST. SIMPLE BASELINES INCLUDED FOR CONTEXT.

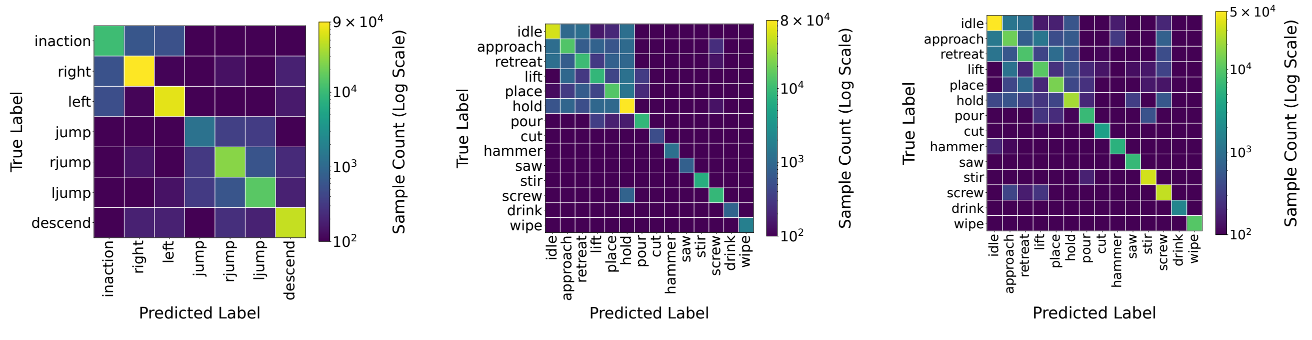
Embedding	Decoder Input	Aggregation Method	Backbone	Accuracy	Precision	B@4	METEOR
Frame	Frame Embeddings	CLS token	DINOv2	54.97±0.86	53.73±0.88	54.03±0.68	58.83±0.69
Frame	Global Embedding	Mean Pooling	DINOv2	41.65±0.28	40.13±0.41	42.71±0.27	47.77±0.36
		Max Pooling		40.69±0.49	39.39±0.55	41.94±0.40	47.09±0.48
		GeM Pooling		39.77±0.77	38.83±0.52	40.82±0.76	45.87±0.75
		NetVLAD		32.83±1.68	31.49±1.57	32.02±1.48	36.67±1.41
		LSTM		64.92±1.01	64.14±0.98	62.61±1.07	67.18±1.03
		Attention Pooling		63.30±0.82	62.82±0.89	62.80±0.88	67.73±0.87
Patch	Global Embedding	Mean Pooling	DINOv2	41.04±0.57	39.40±0.72	41.70±0.54	46.66±0.55
		Max Pooling		15.59±2.29	42.63±1.55	13.10±2.12	16.07±2.39
		GeM Pooling		40.76±0.54	38.87±0.68	38.39±0.68	42.44±0.72
		NetVLAD		60.56±1.43	58.92±1.36	62.68±1.24	68.41±1.11
		LSTM		87.92±0.21	87.76±0.25	88.38±0.27	91.74±0.23
		Attention Pooling		74.24±0.49	73.61±0.51	77.71±0.37	82.51±0.29
		InvPatch (ours)		**94.01±0.36**	**93.88±0.36**	**93.48±0.24**	**95.75±0.10**
		Mean Pooling	VideoMAE	76.19±0.57	75.18±0.62	78.21±0.58	82.57±0.49
		Max Pooling		74.83±0.48	73.85±0.45	79.43±0.38	79.43±0.38
		GeM Pooling		71.58±1.09	69.43±1.27	71.23±1.68	75.95±1.62
		NetVLAD		69.26±1.16	67.52±1.29	70.70±1.47	75.72±1.51
		LSTM		94.72±0.22	94.68±0.23	94.09±0.20	96.29±0.09
		Attention Pooling		92.89±0.22	92.79±0.22	92.40±0.20	94.91±0.18
		InvPatch (ours)		**97.21±0.18**	**97.20±0.18**	**96.19±0.16**	**97.66±0.13**
Random Action				13.40±0.18	20.23±0.38	4.41±0.05	18.12±0.15
Majority Action				31.57±0.82	9.97±0.51	28.47±0.74	33.09±0.85

TABLE II

BIMACS DATASET: INVERSE DYNAMICS RESULTS MEAN ± 95% CI OVER 5-FOLD CV; HIGHER IS BETTER. **BOLD** MARKS THE BEST PER METRIC AND BACKBONE, AND **BOLD AND UNDERLINED** THE OVERALL BEST. SIMPLE BASELINES INCLUDED FOR CONTEXT. BIGNN RESULTS REPORTED PER [42].

Embedding	Decoder Input	Aggregation Method	Backbone	Accuracy	Precision	B@4	METEOR
Frame	Frame Embeddings	CLS token	DINOv2	79.50±0.97	79.54±0.97	72.35±1.32	73.00±1.07
Frame	Global Embedding	Mean Pooling	DINOv2	77.03±1.37	77.52±0.94	69.20±1.78	70.62±1.39
		Max Pooling		78.23±0.86	78.03±0.70	70.82±1.20	72.10±0.99
		GeM Pooling		77.10±1.11	77.97±0.88	69.19±1.23	70.66±0.92
		NetVLAD		74.01±0.75	70.68±1.25	63.80±1.05	61.41±2.10
		LSTM		83.61±1.06	83.46±0.99	77.17±1.22	77.50±1.05
		Attention Pooling		83.57±0.90	83.28±0.87	77.38±1.19	77.12±0.87
Patch	Global Embedding	Mean Pooling	DINOv2	77.47±1.28	76.91±1.04	69.14±1.54	71.01±1.11
		Max Pooling		74.01±0.75	70.68±1.25	63.80±1.05	67.27±0.70
		GeM Pooling		73.87±0.51	70.47±0.73	64.18±0.78	67.34±0.58
		NetVLAD		78.88±1.23	78.70±1.21	70.47±1.57	71.44±1.48
		LSTM		84.34±0.61	84.28±0.65	78.06±0.60	78.12±0.59
		Attention Pooling		81.09±0.99	81.17±0.97	73.83±1.19	74.28±0.95
		InvPatch (ours)		**87.56±0.51**	**87.49±0.47**	**81.24±1.14**	**81.13±0.66**
		Mean Pooling	VideoMAE	80.02±0.88	79.60±0.89	71.60±1.28	73.29±0.86
		Max Pooling		69.69±0.88	66.37±1.00	57.73±1.32	62.48±0.85
		GeM Pooling		69.72±0.67	67.31±0.61	57.03±1.06	62.03±0.79
		NetVLAD		72.95±2.20	72.73±2.07	62.66±3.01	65.93±2.28
		LSTM		85.54±0.79	85.42±0.80	78.67±1.30	79.10±0.94
		Attention Pooling		85.66±0.90	85.56±0.85	78.73±0.98	79.10±0.74
		InvPatch (ours)		**88.99±0.69**	**88.89±0.70**	**83.07±0.88**	**82.80±0.62**
Random Action				7.51±0.21	14.25±0.30	1.87±0.02	8.76±0.10
Majority Action				23.63±0.28	5.59±0.13	5.37±8.93	14.25±4.87
BiGNN [42]				87.00±0.00	85.00±0.00	-	-

6). Global embeddings trained on patch-level representations provide larger gains—approximately 35% with the DINOv2 backbone. Comparing global embeddings trained on frame-level (rows 1-7) versus patch-level (rows 8-13) representations on the same backbone shows little difference for simple pooling methods (rows 2-4 vs. 8-10), except for



(a) MUGEN (b) BIMACS left hand (c) BIMACS right hand

Fig. 3. **InvPatch Confusion Matrix Analysis.** We present confusion matrices aggregated over 5 folds, with sample counts shown on a logarithmic scale due to the large number of samples, for the VideoMAE 16-frame configuration: (a) MUGEN dataset; (b) Left-hand action classification for the BIMACS dataset; (c) Right-hand action classification for the BIMACS dataset.

max pooling, which performs worse with patch features. In contrast, trainable aggregation methods perform better with patch-level (rows 11-13) than frame-level (rows 5-7) representations, by up to 23%. This suggests that patch-level training better handles distribution shifts, as MUGEN differs from the real-world data used to pretrain DINOv2 and its CLS token. With DINOv2, InvPatch improves all metrics by roughly 30% over the best frame-level baseline and yields an average 6% gain over the next-best LSTM patch-level baseline. With VideoMAE, InvPatch improves accuracy and precision by 2.5%, and BLEU@4 and METEOR by 2% and 1.4%, respectively. The best configuration (InvPatch with VideoMAE) achieves a near-optimal accuracy of 97.21%.

Results on the KIT Bimanual Actions Dataset (see Table II) show that, again, all tested configurations outperform the simple baselines (rows 22–23). Aggregating to a global embedding improves performance compared to passing multiple frame-level embeddings, by up to 9%. The performance gap between frame-level (rows 2–7) and patch-level (rows 8–14) embeddings on the same backbone is smaller than on MUGEN. The best baseline aggregation method (LSTM) improves by only about 0.8% when trained with patches (row 12) instead of frames (row 6). This likely reflects the distributional similarity between BIMACS and the pretraining data, which makes the DINOv2 CLS token better aligned with the dataset. State-of-the-art results on this dataset obtained with a similar evaluation setup—such as those reported in the BiGNN study [42] (row 24)—report accuracy and precision of 87% and 85%, respectively. Combined with DINOv2, our method achieves comparable accuracy and improves precision by about 2%. Combined with the VideoMAE backbone—the best overall configuration—it yields roughly a 1.5% gain in accuracy and a 3% gain in precision. Importantly, BiGNN relies on bounding boxes, whereas InvPatch uses only videos.

Confusion Matrix Analysis. We present confusion matrices of Inverse Dynamics results for our best-performing VideoMAE configuration. For the MUGEN dataset (see Fig. 3a), most predictions lie on the diagonal, indicating correct classification. Confusion mainly arises for jump (61% correct, 2,000 occurrences), inaction (89%, 11,000), and left jump (92%, 16,000), the three least represented actions. BIMACS results are split into left- and right-hand confusion matrices (see Figs. 3b-c). Both show strong diagonal intensity with minimal differences between hands. Ambiguity is highest for approach (left 74%, right 70%), retreat (left 70%, right 67%), lift (left 70%, right 76%), and place (left 80%, right 83%), reflecting challenges in temporal depth perception and vertical direction estimation. Notably, these actions are not among the least represented, all having more than 10,000 examples, whereas some actions such as drink with the left hand reach 92% accuracy despite fewer than 1,000 examples. Vertical errors may stem from pretraining augmentation, while depth perception issues could arise from the aggregation method, which may introduce errors by treating motion as spatial variation. In conclusion, InvPatch generally classifies actions correctly, but future improvements could include selecting backbones with augmentations that better preserve directional cues and refining aggregation to enhance depth perception.

Training Set Size: Sensitivity Analysis. Figure 4 shows results for varying training set sizes, evaluating accuracy degradation as data is reduced. All experiments employ the best-performing configuration: i.e. VideoMAE with InvPatch. For space considerations, we report only classification accuracy. On MUGEN, accuracy decreases with modest drops of roughly 2% for the first two reductions and about 3% for the last. Even with only 10% of the training data, the model still achieves nearly 90% accuracy. BIMACS shows a similar trend initially: using 50% of the data yields a mean accuracy of 86.82%. The decline becomes more pronounced at 20% and 10%, which can be explained by differences in the demonstrations across the 10 repetitions. One or two demonstrations do not capture enough of the different ways the same action can be performed.

Embedded Frame Count: Sensitivity Analysis. Figure 5 shows the impact of compressing different numbers of input frames into a single global embedding on model performance. As with the earlier experiments on training set size, experiments reported here use the best-performing configura-

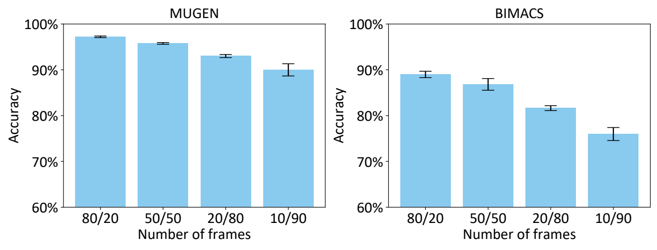


Fig. 4. **Sensitivity Analysis: Training Set Size.** Data efficiency is evaluated for the VideoMAE configuration. We report mean accuracy over 3 runs on MUGEN (left) and BIMACS (right), with 95% CI error bars.

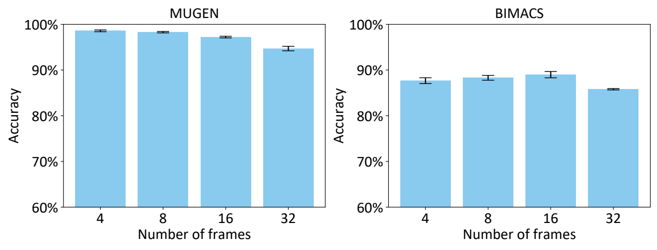


Fig. 5. **Sensitivity Analysis: Embedded Frame Count.** We evaluate the effect of frame count on performance for the VideoMAE configuration. We report mean accuracy over 3 runs on MUGEN (left) and BIMACS (right), with 95% CI error bars.

tion: VideoMAE with InvPatch. Due to space considerations we only report the classification accuracy obtained. Using 4 or 8 frames in MUGEN yields the highest accuracy, while increasing the number of frames to 16 and 32 leads to accuracy drops of 1% and 4%, respectively. For BIMACS, 8 and 16 frames perform the best, while 4 frames yield roughly 1% lower performance, and 32 frames, result in a 3% decrease. The drop in the 4-frame configuration for BIMACS is likely due to prediction on this dataset benefiting from additional temporal context. The decline in the 32-frame setting across both datasets likely reflects the fixed size of the final embedding vector, which becomes limiting as more frames require larger representations. Still, the gradual nature of the drop suggests that compressing multiple frames into a single embedding, and conditioning on it, remains feasible for longer sequences.

Runtime Analysis. Our best-performing configuration uses 16 frames with a frozen VideoMAE ViT-B backbone and 74M trainable parameters (45M in aggregation, 29M in the decoder). The frozen backbone requires 33.8×10^9 floating point operations (FLOPs) per sample and is computed once, then reused during training and inference. InvPatch adds 13×10^9 FLOPs during training, mostly from aggregation. At inference, aggregation is performed once and the decoder runs autoregressively, totaling 20.5×10^9 FLOPs for 16 frames, or 54×10^9 FLOPs per sample including the backbone, with an inference time of 41.8 ms for a 16-action sequence. For comparison, VPT [5] has 500M parameters and requires 4.4×10^{12} FLOPs with 128 frames; even with 16 frames it still requires 550×10^9 FLOPs. Recent vision–language–action models are substantially larger, with smaller variants such as OpenVLA [51] reaching 7B parameters and an estimated 3.3×10^{12} FLOPs [52], as it builds on LLaMA-2 [53]. Overall, InvPatch is a more cost-effective and efficient alternative to contemporary methods.

VI. Discussion

Limitations. This study focused on data and compute efficiency and was conducted on datasets with fixed-length action sequences. Although our sensitivity analysis suggests that InvPatch can scale to longer sequences, we did not evaluate variable-length sequences, which are common in real-world scenarios. Due to limited computational resources, we also did not study scalability with increasing data and compute, which prevented a full comparison with contemporary methods such as VPT [5] or vision-language-action models like OpenVLA [51], beyond runtime analysis.

Future Work. We plan to extend the evaluation to more complex domains, including egocentric video and autonomous driving, which involve variable-length action sequences from overlapping activities (e.g., navigation and tool use, or simultaneous control of steering, throttle, and braking). We will evaluate on both synthetic and real-world datasets: Minecraft [5] for simulation, Ego4D [54] for egocentric actions, and CARLA [55] with HDD [56] for autonomous driving. Another direction is to study generalisation and train a single model across environments, this requires explicit evaluation of the generalisation capabilities of pre-trained vision backbones for fine-grained action prediction, as well as the development of a multi-environment action tokenizer.

VII. Conclusion

This paper introduces InvPatch, a novel method for inverse dynamics modeling that combines prefix-based conditional generation with a patch-level representation aggregation strategy for integrating spatiotemporal information from pre-trained ViT backbones. We evaluated InvPatch: a 2D platformer game (MUGEN) and 3D third-person demonstrations from the KIT Bimanual Actions dataset. Our best configuration surpasses all baselines, achieving 97.21% accuracy on MUGEN and improving the state of the art on the KIT Bimanual Actions dataset by approximately 3% in both precision and accuracy. Combined with our sensitivity and runtime analyses, InvPatch is shown to be an efficient and robust method for inverse dynamics modeling, highlighting its potential to advance research in action understanding and control through videos.

References

[1] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. Wiley, 2008.
[2] D. A. Winter, *Biomechanics and Motor Control of Human Movement*. Wiley, 2009.
[3] T. Pearce and J. Zhu, “Counter-Strike deathmatch with large-scale behavioural cloning,” in *Proc. of IEEE Conf. on Games*, 2022.
[4] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC press, 2017.
[5] B. Baker *et al.*, “Video pre-training (VPT): Learning to act by watching unlabeled online videos,” *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[6] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. of International Conf. on Learning Representations*, 2021.

[7] R. Mokady, A. Hertz, and A. H. Bermano, "ClipCap: CLIP prefix for image captioning," *arXiv preprint arXiv:2111.09734*, 2021.

[8] W. Li, L. Zhu, L. Wen, and Y. Yang, "DeCap: Decoding CLIP latents for zero-shot captioning via text-only training," in *Proc. of International Conf. on Learning Representations*, 2023.

[9] Z. Tong, Y. Song, J. Wang, and L. Wang, "VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training," *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[10] C. Shao, D. Li, F. Meng, and J. Zhou, "Continuous autoregressive language models," *arXiv preprint arXiv:2510.27688*, 2025.

[11] T. Hayes *et al.*, "MUGEN: A playground for video-audio-text multimodal understanding and generation," in *Proc. of European Conf. on Computer Vision*. Springer, 2022.

[12] C. R. G. Dreher, M. Wächter, and T. Asfour, "Learning object-action relations from bimanual human demonstration using graph networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, 2020.

[13] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018.

[14] R. Girdhar and D. Ramanan, "Attentional pooling for action recognition," in *Advances in Neural Information Processing Systems*, 2017.

[15] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, 1997.

[17] E. De Vito, L. Rosasco, A. Caponnetto, U. De Giovannini, F. Odone, and P. Bartlett, "Learning from examples as an inverse problem," *Journal of Machine Learning Research*, vol. 6, no. 5, 2005.

[18] P. Christiano *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[19] K. Paster, S. A. McIlraith, and J. Ba, "Planning from pixels using inverse dynamics models," in *Proc. of International Conf. on Learning Representations*, 2021.

[20] C. Yang *et al.*, "Imitation learning from observations by minimizing inverse dynamics disagreement," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] B. S. Pavse, F. Torabi, J. Hanna, G. Warnell, and P. Stone, "RIDM: Reinforced inverse dynamics modeling for learning from a single observed demonstration," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, 2020.

[22] M. Tot *et al.*, "Adapting a world model for trajectory following in a 3D game," in *Proc. of ICLR Workshop on World Models: Understanding, Modelling and Scaling*, 2025.

[23] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2015.

[24] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based LSTM and semantic consistency," *IEEE Trans. on Multimedia*, vol. 19, no. 9, 2017.

[25] T. Brown *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[26] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "VideoBERT: A joint model for video and language representation learning," in *Proc. of the IEEE/CVF International Conf. on Computer Vision*, 2019.

[27] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proc. of International Conf. on Machine Learning*, 2022.

[28] P. Zhang *et al.*, "VinVL: Revisiting visual representations in vision-language models," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021.

[29] X. Li *et al.*, "Oscar: Object-semantics aligned pre-training for vision-language tasks," in *Proc. of European Conf. on Computer Vision*. Springer, 2020.

[30] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[31] J. Achiam *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[32] L. Chen *et al.*, "ShareGPT4video: Improving video understanding and generation with better captions," *Advances in Neural Information Processing Systems*, vol. 37, 2024.

[33] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. of European Conf. on Computer Vision*, 2016.

[34] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, 1999.

[35] J. Hernandez, R. Villegas, and V. Ordonez, "ViC-MAE: Self-supervised representation learning from images and video with contrastive masked autoencoders," in *Proc. of European Conf. on Computer Vision*. Springer, 2024.

[36] H. Jégou, M. Douze, C. Schmid, and Pablo. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. of IEEE/CVFConf. on Computer Vision and Pattern Recognition*, 2010.

[37] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for action classification," in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2017.

[38] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2015.

[39] Z. Liu *et al.*, "Swin Transformer: Hierarchical vision transformer using shifted windows," in *Proc. of the IEEE/CVF International Conf. on Computer Vision*, 2021.

[40] B. Ni *et al.*, "Expanding Language-Image pretrained models for general video recognition," in *Proc. of European Conf. on Computer Vision*. Springer, 2022.

[41] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proc. of International Conf. on Machine Learning*, 2019.

[42] F. Ziaeetabar, M. Tamosiunaite, and F. Worgotter, "A hierarchical graph-based approach for recognition and description generation of bimanual actions in videos," *IEEE Access*, 2024.

[43] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2002.

[44] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005.

[45] A. Paszke *et al.*, "Automatic differentiation in Pytorch," in *Proc. of NeurIPS Workshop Autodiff*, 2017.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[47] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[48] F. M. Thoker, L. Jiang, C. Zhao, and B. Ghanem, "SMILE: Infusing spatial and motion semantics in masked video learning," in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2025.

[49] F. Baldassarre *et al.*, "Back to the features: DINO as a foundation for video world models," *arXiv preprint arXiv:2507.19468*, 2025.

[50] A. Radford *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.

[51] M. J. Kim *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[52] E. Hsu and K. Roberts, "Leveraging large language models for knowledge-free weak supervision in clinical natural language processing," *Scientific Reports*, vol. 15, no. 1, 2025.

[53] H. Touvron *et al.*, "LLAMA 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[54] K. Grauman *et al.*, "Ego4D: Around the world in 3,000 hours of egocentric video," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2022.

[55] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. of Conf. on Robot Learning*, 2017.

[56] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko, "Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.