Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Multi-Human Behavior Prediction Using Vision Language Models

Utsav Panchal

Course of Study:	Information Technology
Examiner:	Prof. Dr. Marco Aiello Dr. Ilche Georgievski
Supervisor:	Yuchen Liu[1] Dr. Luigi Palmieri[1] 1. Robert Bosch GmbH
Commenced:	Oct 15, 2024
Completed:	May 15, 2025

Abstract

The ability to accurately predict multiple human motions and behaviors is crucial for mobile robots operating in human-populated environments. It is essential to incorporate the context of the scene and the states of objects within the environment because human behaviors are inherently influenced by their surroundings. Although prior research focuses primarily on predicting actions in single-human scenarios from an egocentric view, robotic applications require understanding multiple human behaviors from a third-person perspective. In contrast, there are fewer pre-existing datasets that captures multi-human behavior, especially from a third person perspective. The gap in data availability further complicates the development of accurate and efficient prediction methods for real world applications. This thesis addresses the problem of forecasting actions of multiple humans within a scene from a third person's point of view. By leveraging Vision Language Models (VLMs) and Scene graphs, this thesis proposes a framework that is capable to predict multiple-human behavior in an indoor environment. Due to a lack of suitable dataset for multiple human behavior prediction, this thesis also fine-tunes open source VLMs with synthetic human behavior data and evaluates the resulting models on both synthetic sequences and real-world video recordings to assess their generalization capabilities. Additionally, this thesis also outlines the process of generating synthetic data generated by using a photo-realistic simulator. This thesis presents VISTA, which stands for Vision And Scene Aware Temporal Action Anticipation, a fine tuned VLM which is capable to predict human behavior up-to 5 seconds in a single-shot manner. This work also details a fine-tuning pipeline for VISTA utilizing methods such as Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) resulting in an 13% improvement over existing methods. In the end, this thesis also presents several ablation studies to examine different components of the framework and to understand the factors influencing the behavior prediction.

Contents

List of Figures

List of Tables

1 Introduction

Human action prediction is an important research area that lies at the intersection of computer vision, machine learning, and robotics. Being able to predict human behavior in shared space provides the robot with valuable information to take appropriate actions. An intra-logistics robot that has to navigate a warehouse with human workers present on its own is an example of a practical use case in the industrial sector. Predicted motion in this instance offers crucial information to allow for human-aware robot navigation that stays clear of obstacles and collisions without being unduly cautious [48]. Anticipating future human behaviors can also be advantageous for robots in a domestic setting. They can reduce undesired human-robot interactions that annoy people by taking them into account when planning tasks. Similarly, in autonomous driving, predicting the intention of a pedestrian is a key factor [5]. Human motion and behavior prediction can also be applied where anticipating the activities of other road users is also essential. This includes pedestrians, especially to ascertain as soon as possible if they plan to cross the street so that the autonomous car can react with a non-disruptive and safe action [58]. Furthermore, human action prediction has significant implications in healthcare, where it can be used to monitor patient behavior, assist in rehabilitation, and even detect early signs of neurological disorders by identifying abnormal movement patterns [4].

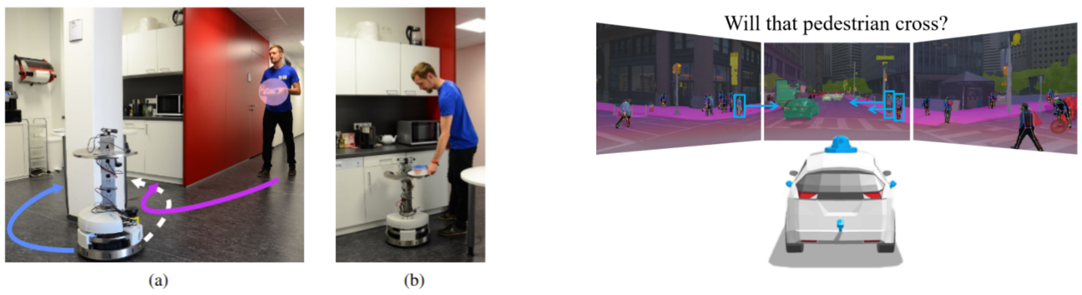


Figure 1.1: Use case of Human Action Prediction. Left: Human Aware Robot Navigation [7]. Right: Predicting Pedestrian Crossing [5]

Human behavior is generally complex, which involves a wide range of motions and gestures that differ in terms of intent, speed, and intensity [49]. Human Action Recognition (HAR) is the task to determine human activities from various kinds of sensor inputs. The required data is usually obtained from either ambient sensors at fixed locations or wearable sensors contained in smart devices, such as phones or watches [23]. Early representations of human actions relied heavily on low-level features such as silhouette shapes, optical flow, and motion trajectories extracted from video frames [61]. Although these representations work well in controlled settings, they are constrained by their incapacity to convey context and higher-level semantic information [61]. There are techniques that include geometric and semantic representations [19]. Statistical techniques such as Dynamic Time Wrapping (DTW) and Hidden Markov Models (HMMs) gave a better accuracy for temporal variability [63]. Although these approaches had good results in gesture recognition and activity analysis, their reliance on handwritten features ultimately limited their scalability and generalization in complex, real-world scenarios [49].

The field of human behavior prediction is still unexplored, despite the fact that human action recognition has been thoroughly studied. Predictive models face many challenges because of the dynamic nature of real-world environments and the inherent variability and complexity of human behavior [9] [49]. In addition to having trouble processing noisy or incomplete data, many traditional methods frequently lack the ability to generalize across a variety of scenarios [64] [17]. Furthermore, accurate and computationally efficient models are required for real-time prediction in high-stakes applications like autonomous driving and robotics [58].

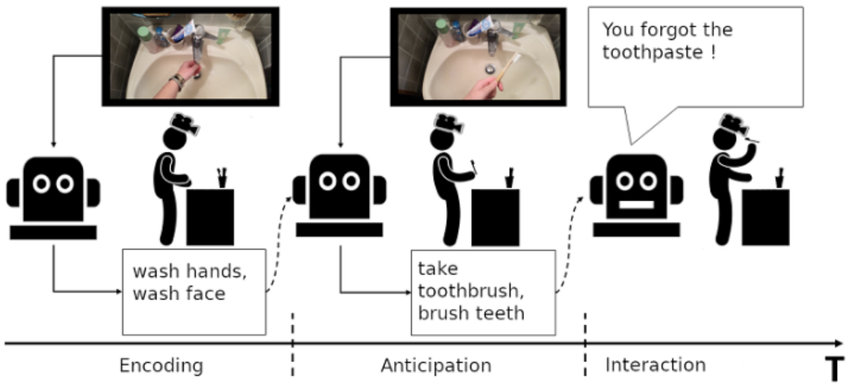


Figure 1.2: Egocentric Action Anticipation [48]

The task of action anticipation involves predicting future human actions based on observed video sequences [34]. Recent research strongly focus on first-person action anticipation [80] [18] using large-scale egocentric datasets [21]. Egocentric data is particularly relevant for applications such as augmented reality, as it captures an agent's viewpoint while interacting with the environment, as shown in Figure 1.2. First-person data is mainly limited to the visual field of a single agent. Existing datasets such as Ego4D [21] captures human actions in various outdoor scenes from an egocentric point of view and EPIC-Kitchens-55 [13] captures human actions in kitchen scenario. EGTEA GAZE+ [40] focuses on what a person is doing and where they are looking based on the analysis of video captured by a head worn camera.

As predicting the future comes with a high degree of uncertainty, it can be helpful to integrate information from different modalities. Visual data from an egocentric point of view and text have cues, so the combination provides a more detailed view on predicting future actions. This limitation poses challenges in scenarios where a robotic system needs to understand interactions between multiple agents, predict actions in dynamic environments, or make real-time decisions based on external observations [48]. In robotics applications, a shift towards action anticipation in third-person view is necessary for practical deployment. Third-person action anticipation, in contrast, allows for a more detailed analysis of spatial relations and contextual cues, making it more suitable for robots operating in industrial or collaborative environments [19].

The thesis is specifically focused on multiple human behavior prediction, as human movements are not isolated but are also dependent on the surrounding environment. While the spatial arrangement of obstacles is important for the trajectory, the behavior depends more on the type, states and properties of objects available nearby, i.e., the environment semantics [9]. Humans tend to act in a goal-directed manner, with their movements often driven by the intention to interact with objects in their surroundings [80]. Such interactions are commonly represented as action-object pairs, such as (sit, chair) or (open, dishwasher) which includes the relationship between human action and the surrounding environment. Therefore an accurate prediction of human action requires the incorporation of scene in which the human is located.

A scene can be represented using 3D Scene Graphs which provides richer semantic and topological information, enabling a more detailed understanding of object affordances and potential interactions. A 3D Scene graph (3DSG) is a data structure to organize and represent logical and spatial relationships within a three-dimensional scene [3]. It is typically structured as a graph or a tree, where each node represents an entity such as object and edges represents relationships between these entities. A 3DSG often include semantic information by organizing scenes in hierarchical layers such as buildings, rooms, objects, and cameras [3]. It often includes not just spatial but also functional relationships, modelling how objects interact for e.g., a handle operates a door [79].

Recent advancements in Large Language models (LLMs) have excelled in Natural Language processing task including text classification and machine translation [69]. LLMs have a broad understanding of the world because of their extensive and varied pretraining on textual data. [6].Vision Language Models (VLMs) expand this capacity by adding visual modalities and can now complete tasks that call for both textual and image-based reasoning [16]. In the context of human behavior prediction, VLMs provides the benefit of jointly processing human actions from video data and representing the surrounding environment using structured scene information, like a 3D scene graph.

The thesis addresses the potential applications for Human Behavior Prediction based on the above introduction, outlining the limitations of current models in terms of spatial reasoning. The main focus of the thesis will be on predicting human behavior in natural language which has proven to be significantly more promising when working with VLMs [69] [16].

This thesis makes the following key contributions:

- To address the aforementioned vacancies in the research field, this thesis proposes a novel VLM-based framework to enable a more efficient multiple human behavior prediction from a third-person view.
- To further improve the prediction accuracy of the competitive open-source VLMs, both Supervised Fine Tuning (SFT) and Direct Preference Optimization (DPO) are utilized to fine-tune and align the VLMs, which helps in capturing and anticipating multiple human actions in a scene.
- Due to lack of suitable datasets, this work introduces a new synthetic dataset, generated by a photo-realistic simulator, comprising video sequences of multiple human actions in indoor environments, designed to facilitate research on multi-human action prediction.
- Evaluated with both synthetic and real-world recorded data, the fine-tuned model achieves 13% improvement over existing baselines, indicating more accurate predictions.

The remaining chapters of the thesis are structured as follows. Chapter 2 will introduce the necessary background on VLMs and the prompting techniques such as In-context learning (ICL) and fine tuning techniques such as Supervised Fine Tuning (SFT) and Preference Optimization (DPO) Methods. The related work for human motion and behavior prediction will be discussed in Chapter 3. Next, Chapter 4 will present the VLM-based system developed during this thesis for human action prediction with scene context and formulate the specific prediction task focused on interactions. The evaluation of the system on human behavior prediction is presented in Chapter 5, including a detailed analysis of the different system components. Finally, Chapter 6 will conclude the findings of the thesis and give some insights to the future works.

2 Background

In this chapter, Section 2.1 covers an overview of Language Models including Large Language models and Vision Language models including its architecture and pretraining methods. Section 2.2 covers details about prompting techniques such as In-context learning. Section 2.3 covers detail about Fine Tuning methods used in this thesis and Section 2.4 provides a description on the simulator that is used for dataset generation.

2.1 Language Models

2.1.1 Large Language Models

In Natural Language Processing (NLP), large language models (LLMs) have become transforming tools that fundamentally change how text is produced, understood, and applied over many domains. Transformer architecture shown in 2.1 [69] which uses self-attention mechanisms to imitate complex, long-range dependencies in textual data at least partially drives this revolution. Transformers process data in parallel, unlike conventional recurrent neural networks (RNNs) and convolutional neural networks (CNNs), so greatly lowering training times and increasing scalability [69].

Built on an encoder–decoder architecture, the transformer architecture consists of several layers including feed-forward neural networks and self-attention [16]. Self-attention helps the model to dynamically balance the relevance of every word in a sentence, independent of its location, so enabling the capture of complex contextual relationships. From language translation and summarizing to sentiment analysis and question answering, this design innovation has been essential in attaining state-of- the-art performance on a wide spectrum of NLP tasks.

Model scale has fast increased in the field over the last few years [32]. LLM iterations one after another have expanded exponentially in terms of the number of parameters; models such as GPT-3 with 175 billion parameters [6]. Larger models are able to capture more varied patterns and dependencies in language, thus this dramatic increase in scale has been closely related to improved performance. Scaling rules help to explain the phenomena since they show that the predictive performance of language models increases with model size given enough computational resources and training data [32]. Hoffmann et al. [26] contend, however, that rather than the largest possible models, smaller models with more training data produce the best performance when considering a given training compute budget. Aiming for models that are less expensive at inference and can be served more readily, the openly published LLaMA models [68] change the focus from training compute budget to inference budget. They show that small models with 7B parameters outperform past models with far more parameters since they keep improving even after more than 1T training tokens.

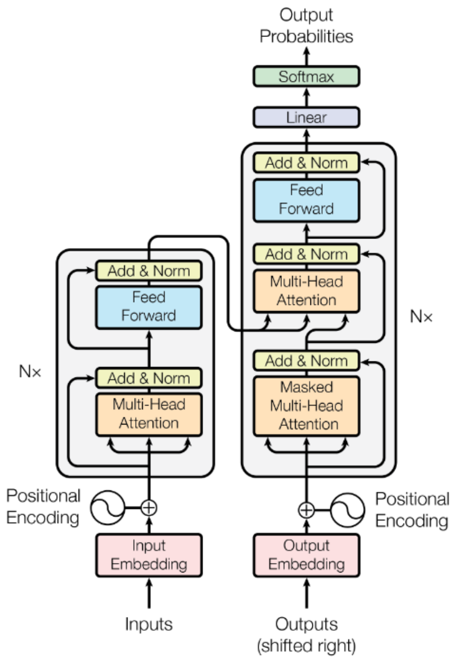


Figure 2.1: Transformer Architecture [69]

2.1.2 Vision Language Models (VLMs)

Vision language models (VLMs) bridges the distance between language and vision by integrating of textual and visual modalities inside a single framework [16]. Tokenization and positional encoding is the process by which conventional transformer architecture converts input sequences into a set of contextualized embeddings. Like tokens in a sentence, images in visual data are first divided into fixed-sized patches as seen in Figure 2.2 [16]. Usually with an additional spatial arrangement capture by means of a positional embedding, each image patch is then flattened and projected into a high-dimensional vector space. Popularized by the Vision Transformer (ViT), this method lets images be handled in a way akin to text, so enabling the transformer to attend over visual elements with the same mechanism it applies for words [16]. Combining both modalities, CLIP (Contrastive Language–Image Pre-training) learns visual concepts from natural language supervision [54]. Encoders for both modalities are trained jointly using a large set of image-caption pairs to guarantee that matching image and text pairs lie near in the shared embedding space [54].

VLM Architecture

Vision Language Models typically consists of three main components as shown in Figure 2.3 [71]. A Vision Encoder to extract information from input images, a Language Model backbone and a Projection Layer ensuring the connection from visual input to the language space.

A **Vision Encoder** backbone is responsible for extracting high-level visual representations from raw image or video inputs. This backbone processes pixel-based data through convolutional layers or transformer-based operations which encodes the spatial and semantic information into a compact

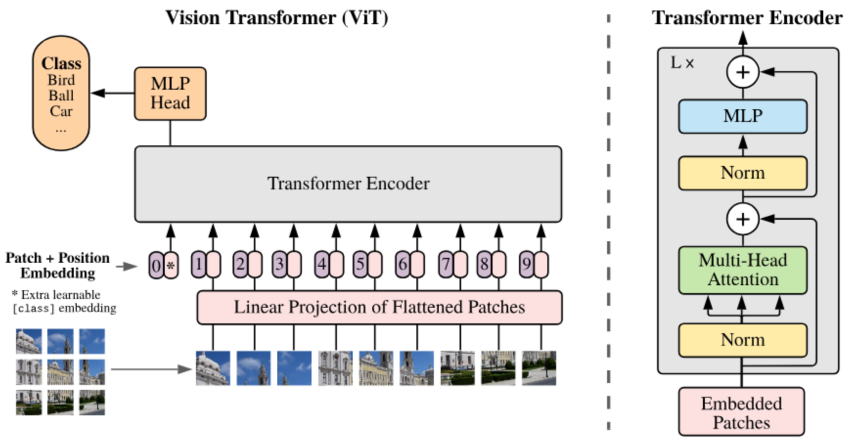


Figure 2.2: Vision Transformer [16]

feature space. The encoded visual representations are then aligned with textual embeddings from the language model backbone which facilitates multimodal understanding. Most VLMs rely on language-supervised CLIP models [54], as these come with the benefit of massive pretraining on text-image data. This way, the visual space is already pre-aligned with language, allowing an easier adaption of the visual tokens into the LLM token space. VLMs are not limited to one single vision encoder, but can use multiple at the same time. However, as different architectures come with different strengths, the performance can be further improved by combining multiple encoders. While CLIP models generally uses Vision Transformer (ViT) as their backbone, different kinds of backbones are available depending on the requirements. For example, OpenCLIP ConvNeXt [45] is majorly used in tasks that involves high resolution image processing and CoAtNet [11] which is hybrid model combining convolutional layers with attention mechanisms to capture local features while effectively modeling long range dependencies.

The **Language model** is the architecture which is responsible for processing and generating text. It serves as the fundamental structure upon which additional modalities, such as vision or audio, are integrated. The backbone typically consists of transformer-based architectures. Commonly used backbones in VLMs include GPT [55], Meta's LLaMa family [68], Google's Gemma models or Alibaba's Qwen [71].

The **Projection Layer** is responsible for guaranteeing interoperability between the visual and language domains. This adapter module can be implemented using a variety of architectures, including linear layers, Transformer-based modules, or cross-attention layers that are incorporated into the LLM [8]. The conceptually simplest approach is represented by linear layers, which project the visual features into the textual embedding space using a linear mapping. This can be executed in either a single-layer or multi-layer perceptron (MLP). The Q-Former, which is featured in BLIP-2, is the source of transformer-based adapters [39]. The Q-Former facilitates the exchange of information between the two modalities by utilizing two Transformer blocks with shared self-attention layers to bridge the vision and language space. The interaction between the image features and learnable queries vectors enables the extraction of specific relevant information.

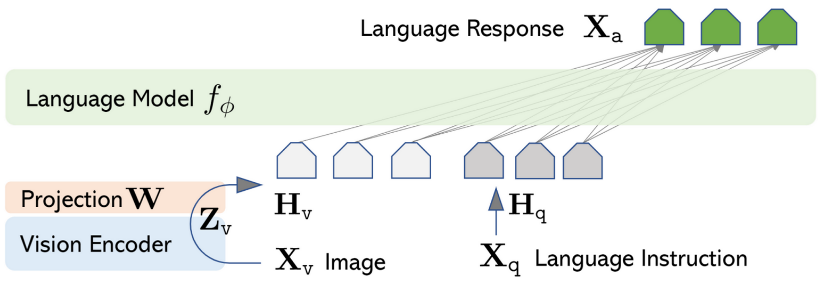


Figure 2.3: Llava-VL architecture [42]

Multimodal Pretraining

Large-scale video-text datasets are used in video-language pretraining to either perform self-supervised or unsupervised learning, to obtain generalized multimodal representations [50]. The pretraining tasks meant to improve various aspects of video-language comprehension. Important tasks include Masked Frame Modeling (MFM), which focuses on predicting mask-based visual frames to improve temporal representation learning, and Masked Language Modeling (MLM), which masks and recovers textual tokens to enable language prediction [14]. While Video-Language Matching (VLM) guarantees cross-modally alignment by discriminating related video-text pairs, Language Reconstruction (LR) improves sentence generation by reconstructing textual sequences from partial inputs [14]. By training the model to predict the proper order of textual descriptions and video frames respectively, Sentence Ordering Modeling (SOM) and Frame Ordering Modeling (FOM) also enforce temporal coherence [14]. These activities taken together help the model to learn sequential dependencies, multimodal alignment, and video caption generation, so enhancing its capacity for strong video-language reasoning.

This training can be either be done in one single stage or in a two-stage process [8]. Both approaches use an auto-regressive objective, which is standard for LLMs by optimizing the cross-entropy loss for next token prediction. A significant advancement in vision-language pretraining is Contrastive Language-Image Pre-Training (CLIP), which employs contrastive loss to acquire multimodal representations from weakly supervised datasets. Trained on a large-scale corpus of 400 million image-text pairs [54], CLIP has demonstrated remarkable performance in zero-shot visual recognition tasks, including image classification.

VLM Benchmarks

There are various benchmarks evaluating VLMs on different types of multimodal tasks. Tong et al. [67] examine the most often used benchmarks to evaluate VLM across several multimodal tasks in order to evaluate how accurately they test multimodal ability. They contrast VLM performance on these tests both with and without visual input. When visual input is included, benchmarks such as MMMU [78] and MathVista [46] show only a small variation, implying that their main emphasis is on the language component rather than the visual aspect. They further divide the benchmarks into four groups: General, Knowledge, Chart & OCR, and Vision-Centric. The few benchmarks that are really vision-oriented, such as MMVP [58] and Real World QA [33], have a rather limited sample count.

To address these limitations, Tong et al. [67] present the Cambrian Vision-Centric Benchmark (CVBench) to help overcome these constraints. As shown in Figure 2.4, CVBench highlights 2D understanding by evaluating spatial relationships between objects and object counting as well as 3D understanding by ordering objects by depth or relative distance to an anchor. The objects to be placed are textual input along with their matching bounding boxes in the image; hence, the VLM is needed to show referring ability, as already mentioned. Given its emphasis on real-world visual understanding in both 2D and 3D, CVbench is especially pertinent for VLM applied in embodied artificial intelligence, such robotics and autonomous vehicles, which must interpret and negotiate their physical environments.

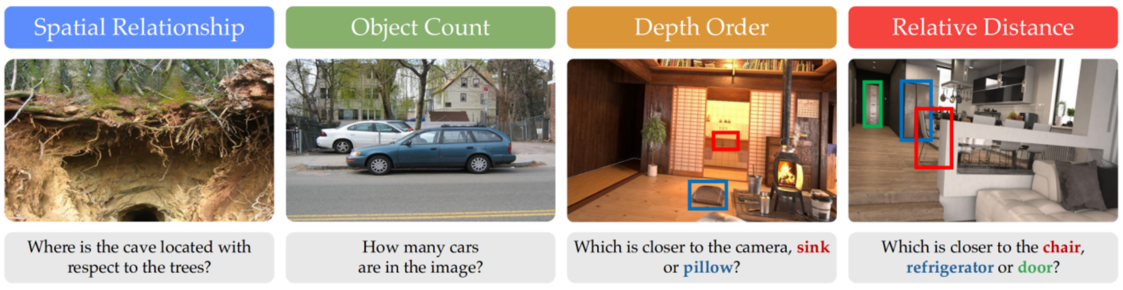

Spatial Relationship
Object Count
Depth Order
Relative Distance
Where is the cave located with respect to the trees?
How many cars are in the image?
Which is closer to the camera, sink or pillow?
Which is closer to the chair, refrigerator or door?

Figure 2.4: Cambrian Vision-Centric Benchmark (CV-Bench) [48] assesses VLMs on core 2D and 3D visual understanding skills, which are crucial for embodied AI systems operating in real-world environments. [67]

Designed to assess environmental understanding through natural language questions, OpenEQA [47] is introduced as an open-vocabulary benchmark dataset for Embodied Question Answered (EQA). It mostly addresses two important uses. The first is episodic memory, in which responses must be derived from a fixed history of observations captured by a wearable device, say smart glasses. A common query might be, "Q: I can't find my keys; where did I leave them? A: In the island kitchen.The second use is active exploration, in which a mobile robot independently gathers the required data to address a question by navigating its surroundings. For instance: "Q: Do we have home canned tomatoes? A: I did discover some canned tomatoes in the pantry. Since this benchmark is meant for foundation models including VLM integrating perception with language reasoning and general world knowledge calls for both The questions cover many spheres, including object localization, attribute recognition, state identification, and spatial or functional reasoning.

2.2 In Context Learning

Large language models (LLMs) have a major quality in their great pre-training on enormous amounts of internet-sourced data [14]. Strong zero-shot capabilities are made possible by the great range of texts included in the training data. Originally proposed in computer vision, the idea of zero-shot learning describes the capacity to identify or classify objects not included in training data [76]. In the framework of LLMs, a zero-shot environment denotes that the model gets only a textual description of the task within the prompt. By contrast, one-shot or few-shot learning gives the model task-specific examples in its environment [6]. These examples show the intended behavior by means of input-output pairs. Following these examples, the real input is provided, which forces the model to create an output by following the shown pattern. Using the example of sentiment prediction from food reviews, an example is shown in Figure 2.5.

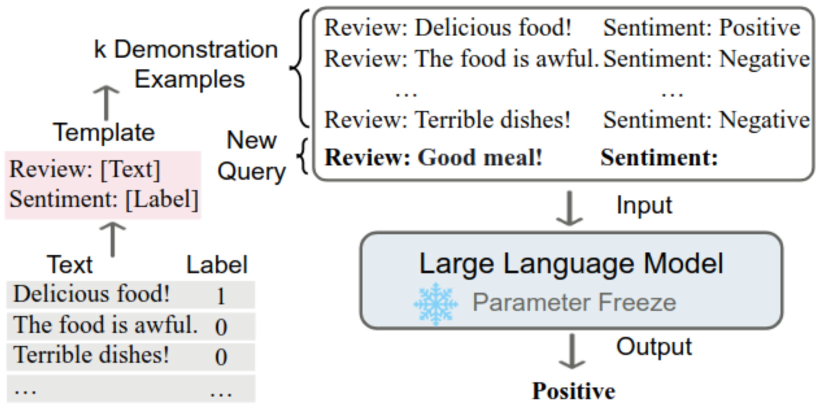


Figure 2.5: In Context Learning: ICL requires prompt context containing few examples written in natural language.LLMs are responsible for making predictions using this prompt and query. [15]

The term in-context learning comes from the reality that learning happens through input-output example pairs limited to the current context of the model. Unlike classic training examples, these do not change the weights of the model [6]. Two main benefits of this method surpass those of task-specific fine-tuning. First, in-context learning can be done during inference, so saving the need for extra training since it does not call for changing model weights [6]. Second, it greatly reduces the required quantity of training examples. Few-shot learning can achieve comparable results with just 10 to 100 examples, depending on the context window size of the model, even if fine-tuning usually calls for thousands of labeled examples [76]. Although fine-tuning offers great performance on optimized tasks, the growing scale of large language models both in size and pre-training data has led to improved few-shot generalizing [6]. LLMs can thus be used for a widening spectrum of tasks where in-context learning is sufficiently successful, so lessening the need for intensive fine-tuning.

Extending in-context learning from text-only domains to multimodal inputs makes it much more challenging. Flamingo models [1] are flexible architecture able of processing text interleaved with images or videos in arbitrary sequences to enable visual in-context learning. These models can manage several visual inputs since they are trained on vast web data including images and text. Flamingo can process several image-text pairs as few-shot learning examples during inference, not just one image [1] . Every extra image, however, has to be converted into a lot of visual tokens, which greatly increases inference costs. Scaling up the few-shot example count rapidly becomes resource intensive and usually reaches the maximum context length the VLM can support. Given these restrictions, downstream performance depends much on the relevance of particular examples. The limitations will be discussed in the Evaluation section. Flamingo uses the Retrieval-based In-

Context Example Selection (Rices) method [77] to maximize example choosing. Using visual features derived by the vision encoder, the system searches a support set for most similar images for a given query image. These retrieved examples are also arranged such that the most pertinent ones show toward the end of the input sequence, so mitigating recency bias [81] in LLMs. This method guarantees inclusion of the most important visual examples while maintaining a reasonable total input token length.

2.3 Fine Tuning Methods

2.3.1 Supervised Fine Tuning

Supervised fine-tuning (SFT) is a crucial method in machine learning that develops after pre-trained models to adapt them for particular tasks. When pre-training on large, general datasets is conducted and fine-tuning improves the model's capacity for more specialized uses, this process is particularly valuable. After a model has been pre-trained on a large, generic dataset [14], fine-tuning means changing its parameters to maximize its performance on a given task [51]. Usually beginning with pre-training parameters learned, the model uses a supervised learning technique to update all parameters during fine-tuning, so leveraging labeled data particular to the current task at hand. Usually, this process is generally known as Instruction Fine Tuning [73].

One common method is to fine-tune a pre-trained model such as a CNN for image classification or a transformer-based model for NLP by means of a target task. Using a smaller task-specific labeled dataset [14], fine-tuning adapts the model for particular NLP tasks, such sentiment analysis or question answering, from pre-training using a massive text corpus [14]. Using pre-trained MiniGPT-v2 weights from generally broad domains, PefoMed [25] further fine-tune the models using multimodal medical datasets in two stages. Google Gemini models [65] perform supervised fine tuning on downstream task by using the Gemma Pretrained models as shown in Figure 2.9. [66]

Parameter Efficient Fine Tuning

The computational requirements associated with the fine-tuning of Pretrained Language Models (PLMs) have grown to be a major difficulty as their scale keeps increasing. Models such BERT, with 110 million parameters [14] and T5, with 770 million parameters, show this trend and culminate in large-scale architectures such Falcon-180B, which may need minimum of 5120 GB of computational resources for Instruction fine-tuning [2]. Parameter-efficient fine-tuning (PEFT) has become a feasible substitute for these computational limits [27]. PEFT uses several deep learning methods to drastically lower the number of trainable parameters while preserving performance on par with full fine-tuning. Usually involving either adding extra trainable parameters or updating only a small subset of the pretrained parameters, this method preserves the knowledge embedded in the PLM while customizing it to particular downstream activities. Moreover, since the size of the fine-tuned dataset is usually far smaller than the pretrained dataset, full fine-tuning to update all the pretrained parameters could cause overfitting, which is avoided by the PEFT either selectively or not-updating the pretrained parameters.

PEFT introduces **Adapter** based training method [27]. Adapters are usually small neural network modules inserted between layers of a pre-trained transformer as shown in Figure 2.6. The original model remains mostly unchanged, and only the adapter layers are updated. Many adapters use low-rank transformations such as LORA (which is briefly discussed in next subsection) to efficiently capture task-specific knowledge. This approach significantly reduces the number of parameters that need to be updated, making fine-tuning more efficient in terms of memory and computation. The architecture of the adapter module is determined by the developer's design choices. The authors conducted experiments with multiple complex architectures and identified the depicted design (as shown in the Figure 2.6) as the most suitable configuration based on empirical evaluations [27].

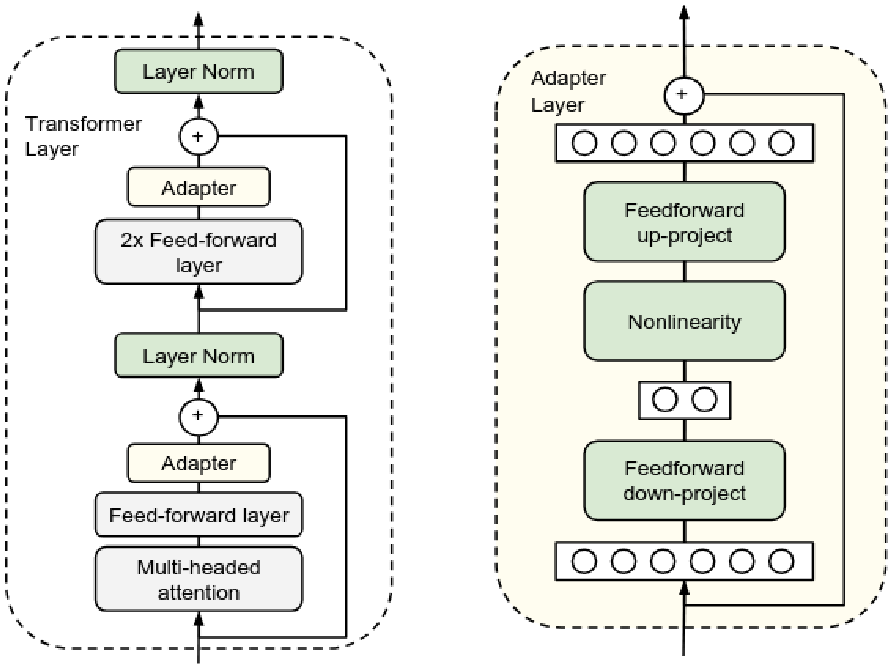


Figure 2.6: Architecture of Adapter Module and its integration with transformer. **Left:** Adapter Module is added twice to each transformer layer. **Right:** Internal Layers of a Single Adapter Module [25]

Low Rank Adaptation Algorithm

Low-Rank Adaptation (LoRA) [28] is a parameter-efficient fine-tuning technique designed to adapt large-scale pre-trained models to new tasks with minimal computational overhead. While the original authors uses GPT-3 as the test case and focuses on language models and NLP tasks, this technique is quite generalizable. It can be applied to various models in multiple contexts [28]. This thesis uses LoRA method to fine tune the VLM on downstream tasks, which is covered in subsequent chapters.

Previous studies have demonstrated that over-parameterized large models tend to lie on a low-dimensional intrinsic manifold [38]. LoRA builds upon this insight by saying that the change in weights during model adaptation also exhibits a low intrinsic rank or dimensionality [28]. Specifically, if $W_{n,k}$ represents the weight matrix of a given layer, and $\Delta W_{n,k}$ denote the change in these weights during the adaptation process. The LoRA framework proposes that $\Delta W_{n,k}$ is as a low-rank matrix, hence the adaptation takes place in a much smaller-dimensional subspace than the original weight space [28]. By constraining the complexity of the weight changes, this assumption enables more scalable and effective model fine-tuning.

LoRA creates low-rank matrices from the weight updates in transformer models. Instead of directly fine-tuning the full-rank weight matrix W, LoRA represents weight updates as:

$$\Delta W = AB$$

where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$ are learnable matrices with rank $r \ll d$. These matrices capture task-specific knowledge while keeping the pre-trained model weights frozen as shown in Figure 2.7. A random Gaussian initialization is used for A and B is initially to 0, so BA=0 at the start of training. The update BA is additionally scaled with a factor α/r, where r is the rank of the matrix and α is scaling factor [28]. LoRA is primarily applied to attention layers, improving efficiency without sacrificing performance.

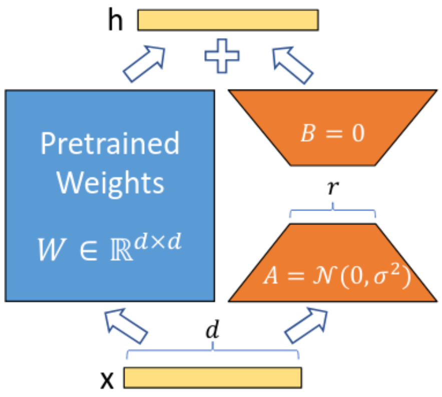


Figure 2.7: Reparameterization of LoRA. Only Matrices A&B are trained [28]

LoRA primarily helps with the decrease in training time [28]. Using the decomposition $\Delta W = A \cdot B$ helps one to adjust the just $r(n + k)$ parameters. This number of parameters is much less than the total number of parameters (nk) in the full-rank weight matrix, so lowering the memory and computational expenses related with fine-tuning. From this comes a more efficient adaption mechanism. LoRA also ensures that during model deployment no more inference time is involved. Computation of $W' = W + BA$ and storage of the results preserves the inference process unchanged, so ensuring no extra latency during producing use [28]. Moreover, LoRA simplifies task switching since only the LoRA weights rather than the whole set of parameters have to be swapped between tasks, so enabling faster and more economical model adaptation to different tasks. Once the LoRA weight matrices are aggregated into the whole weight matrix to cut further inference time, task switching loses flexibility. This process makes it more difficult for one to rapidly change jobs. Moreover, using LoRA in a situation involving several tasks involving different A and B matrix becomes challenging when batch inputs from several tasks in a single forward pass. Separate low-rank updates for every job lead to this issue, so complicating the batching process and maybe reducing computational efficiency.

Once the LoRA weight matrices are merged into the full weight matrix to eliminate additional inference time, the flexibility of task switching is compromised [28]. This process reduces the ease with which one can quickly swap between tasks. Additionally, when using LoRA in a scenario where multiple tasks with different A and B matrices are involved, it becomes challenging to batch inputs from different tasks in a single forward pass. This issue arises due to the need for separate low-rank updates for each task, complicating the batching process and potentially reducing computational efficiency.

2.3.2 Alignment Techniques

Reinforcement Learning from Human Feedback

Reinforcement Learning from Human Feedback (RLHF) is a method to align responses of LLMs with human preferences and values [10]. This process is important to make sure that the responses from LLMs aligns with human expectations, values and following critical safety requirements [52]. RLHF uses human feedback to define the responses as desirable or undesirable to guide the models behavior.

The RLHF framework involves three components[52].

Reward Model Training: Human Annotators provide feedback on LLM outputs, generally by ranking or rating criteria. This feedback is used to train a separate reward model that predicts how well the given output aligns with human preferences.

Policy Optimization: In the next step, the trained reward model is used in place of human. The LLM is then fine-tuned using reinforcement learning algorithms such as Proximal Policy Optimization [52] as shown in figure 2.9.

Iterative improvement: The process is iterative, as the LLM improves, new outputs can be evaluated by humans, further refining reward model and the LLMs behavior.

Direct Preference Optimization

VLMs and LLMs have great wide knowledge but their unsupervised training makes it difficult to produce organized output according to human preferences [14]. Traditionally, these models have been taught by methods such as RLHF as described in the above section, which alters the model's outputs to fit human expectations [10]. Still, this process can be complicated and unpredictable. A new approach, Direct Preference Optimization (DPO) [56] simplifies this from the beginning. Often referred to DPO, this preference tuning method avoids actual reinforcement learning and RLHF [10]. DPO combines direct human feedback to raise the accuracy and efficiency of the AI decision-making process. DPO changes the reward model in RLHF, thus simplifying the fine-tuning process as shown in Figure 2.9 and ensuring stability. It eliminates the need for ongoing training constant adjustments and repeated sampling.

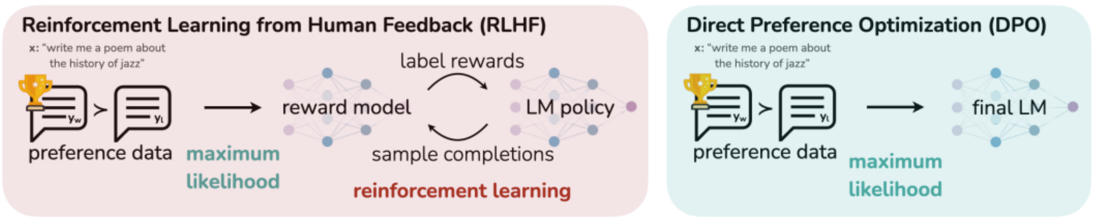

Reinforcement Learning from Human Feedback (RLHF)
x: "write me a poem about the history of jazz"
preference data
maximum likelihood
reward model
label rewards
LM policy
sample completions
reinforcement learning
Direct Preference Optimization (DPO)
x: "write me a poem about the history of jazz"
preference data
maximum likelihood
final LM

Figure 2.8: Difference between RLHF and DPO training methods: Left figure shown training process of RLHF and Right figure shows training process of DPO [56]

LLMs/VLMs like ChatGPT are built in three stages:

Pre-training: First, a base model such as GPT-3 [50] or Llama 3 [68] is trained on internet-scale data to predict the next word which gives the model general internet knowledge and language skills [14].

Supervised Fine-Tuning (SFT): Uses particular examples like to fine-tune the base model. For e.g., dialogues or question-answering to direct it towards a specific use case, so enhancing the outputs of it appropriate for tasks.

Preference Tuning (RLHF/DPO): After SFT, models are fine-tuned based on human preferences. Instead of just giving the model good examples, humans compare outputs and choose which one they prefer. Early on, this was done with RLHF, but now DPO is often used because it's simpler. This step makes models much better at being helpful, harmless, and aligned.

Google Gemini model white paper [65] provides a nice visual representation of these stages.

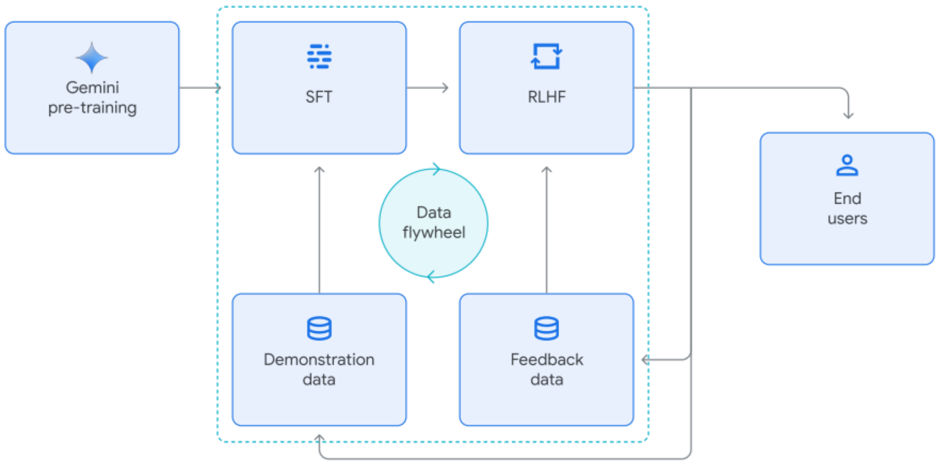


Figure 2.9: Different training stages of Google Gemini Model [65]

For given environment, DPO depends on a dataset of human preferences over pairs of model-generated responses [56]. The language model's direction towards intended behaviors depends critically on this preference data. The preference dataset D comprises of tuples $(x^{(i)}, y_w^{(i)}, y_l^{(i)})$:

- $x^{(i)}$ is the input or prompt directed toward the language model.
- $y_w^{(i)}$ is the preferred response selected by a human labeler for the prompt $x^{(i)}$ from a pair of produced responses.
- $y_l^{(i)}$ is the dis-preferred response from the pair for the same prompt $x^{(i)}$.

Usually, this preference data is gathered by having a base language model often a supervised fine-tuned model generate pairs of responses for different stimuli [56]. Human labeler then evaluate these two answers and indicate which one they would like based on particular standards (e.g., helpfulness, safety, quality). Without explicitly learning the reward function $r^*(x, y)$, the DPO training process seeks to directly maximize a language model policy $\pi_\theta(y|x)$ to align with the given preference data.

2.4 Simulator Overview

This thesis utilizes Virtual Home simulator [53], created by Puig et al. (2018), to simulate household activities. This section provides an overview of the simulator's key features and describes the environments used in our experiments.

Virtual Home is a 3D simulation environment built using the Unity 3D game engine. Its primary purpose is to simulate a wide range of human actions and common household activities by driving artificial agents to execute predefined programs. Given an indoor scene and an activity sequence consisting of atomic actions, the simulator generates the character's pose over time and, if needed, provides renderings from cameras within the environment. The Virtual Home simulator provides a range of indoor environments with varying room layouts and item availability. Virtual Home provides 6 environments in total. An example of environment is shown in Figure 2.10. All environments consist of three rooms. During the experiments, all 6 environments are utilized to generate simulations.

The core concept of Virtual Home revolves around representing activities as programs, which are sequences of simple, symbolic instructions. Each instruction references an atomic action for e.g., *sit*, *walk* or interaction for e.g., *pick-up object*, *Switch on/off* and specifies the objects involved (e.g., "pick-up juice"). This program-based representation offers a clear and non-ambiguous description of all the steps needed to complete a task. Atomic actions in the Virtual Home simulator are encoded in the following simulator-specific format.

`[action]` $\langle l_j \rangle (ID_j)$ ...

Here, l_j represents the semantic label of an item (e.g., "mug"), while ID_j denotes the item's unique identifier, as multiple instances of the same item may exist within a program. An action may be linked to zero, one, or multiple items.

Six **agents** are total accessible in the Virtual Home environment. These agents navigate in the surroundings and engage with different objects. Every agent is shown as an entity in the surroundings and specified by a different **charid**. Among the charid values might be *male1*, *female1*, *male2*, *female2*, and so on. Virtual Home's agents interact via pre-defined programs, which comprise ordered sets of instructions defining the actions each agent should take and the objects they should interact with. Such programs follow this kind of structure:

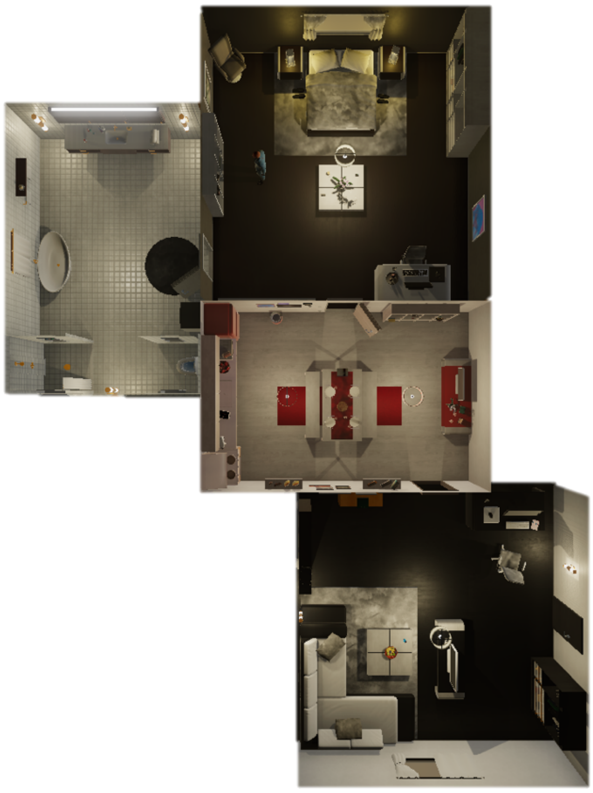


Figure 2.10: Virtual Home Environment Scene 1 the layout includes a washroom on the left, a bedroom at the top, a kitchen in the center, and a living room at the bottom. [53]

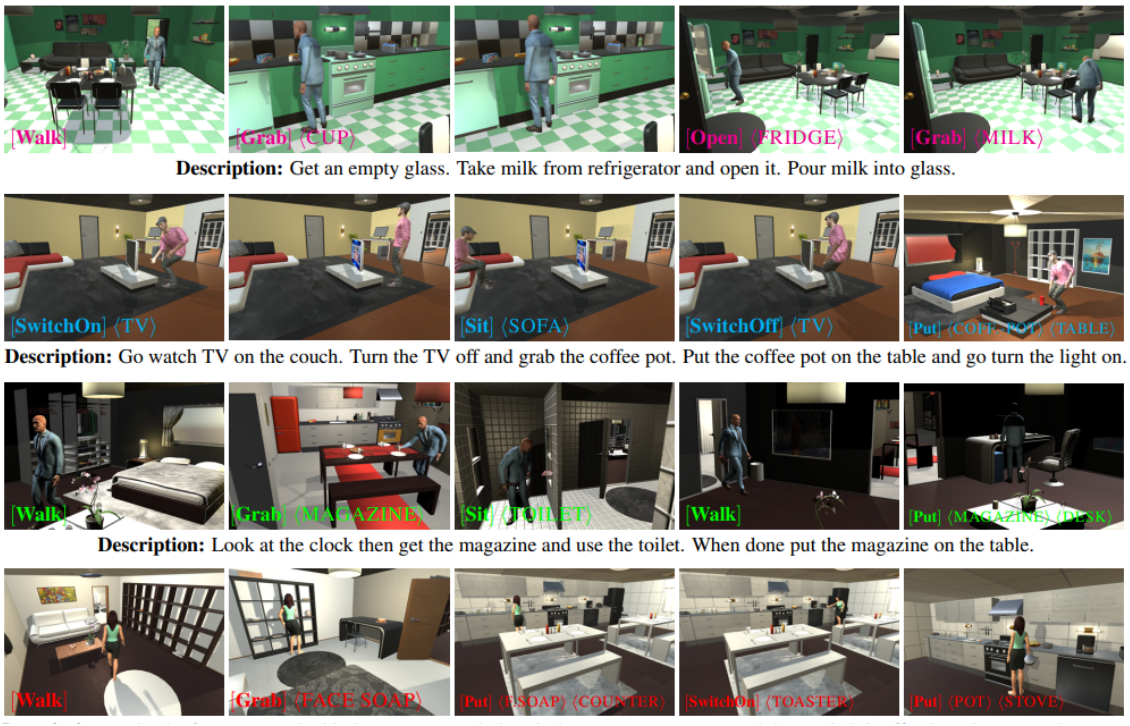
`<char{id1}>` $[action1]\ \langle l_j \rangle\ (\mathrm{ID}_j)$

In the simulation, a human agent will consistently refer to the same item if multiple instances share the same identifier within a given program. For instance, in the activity program below, the agent will select an arbitrary mug from the available mugs, walk towards it, and subsequently grasp that specific mug:

- `[Walk]` $\langle$mug$\rangle$(1)
- `[Grab]` $\langle$mug$\rangle$(1)

If the second instruction be written as `[Grab]` ⟨mug⟩(2), the agent would try to select an other mug than the one it first came upon. The item recognition does not create a fixed mapping to a particular simulation item. Rather, the simulated agent defines dynamically the objects to interact with. The simulator also controls inverse kinematics and path planning to create realistic character motions.

To enable thorough scene coverage, Virtual Home uses a system of several fixed cameras placed within every room. Based on agent visibility, the system dynamically moves between cameras during simulation recording and modulates camera settings to maximize agent-object interaction capture. The camera attributes including position, angle, and field of view help to improve the variety of the produced video data. Specifying the intended camera mode allows to manually choose the recording camera or activate multi-camera recording. This thesis concentrates on obtaining third-person perspective videos. Virtual Home defines its spatial coordinates (x, y, z) to enable functionality for positioning a camera at a given index inside a room. Then, by adjusting the camera mode, one can activate the designated camera.



Description: Get an empty glass. Take milk from refrigerator and open it. Pour milk into glass.

Description: Go watch TV on the couch. Turn the TV off and grab the coffee pot. Put the coffee pot on the table and go turn the light on.

Description: Look at the clock then get the magazine and use the toilet. When done put the magazine on the table.

Figure 2.11: Agent executing generated programs from descriptions, in Virtual Home. [53]

These examples highlight the level of detail captured in the programs and the capability of Virtual Home to animate these programs in a simulated 3D environment.

3 Related Works

This chapter provides a comprehensive overview of related work in the literature. Section 3.1 covers a general action anticipation overview. Section 3.1.1 presents an analysis of existing methodologies that utilize an egocentric perspective, focusing on approaches leveraging first-person visual input for action anticipation. Section 3.1.2 examines techniques based on an third person viewpoint, detailing methods that rely on third-person observations to predict human actions. At the end Section 3.2 describes the available datasets.

3.1 Action Anticipation

Action anticipation is the study of future human behavior characterized as actions represented by verb-noun pairs [80] [34]. With the advances in deep learning especially in LSTMs, the field has shifted towards learning temporal dependencies directly from raw data which achieves greater accuracy [83] [36].Most works are now focused on egocentric action anticipation with the rise of large-scale first-person view datasets and related difficulties like EPIC-Kitchens [12] and Ego4D [21]. Ego4D groups tasks involving action anticipation depending on the prediction horizon. Short-Term Object Interaction Anticipation specifically targets the prediction of approaching human-object interactions. The aim of this work is to predict the following important components.

- Spatial positions of the active objects the person will interact with next (e.g. bounding boxes marking the objects)
- Categories of the next active objects (e.g., “knife”, “tomato”)
- Actions, describing how the active objects will be used (e.g., “take”, “cut”)
- Time to contact, when the person will start the interactions by touching the objects (e.g., “in 0.5s second”)

In contrast, the long-term action anticipation (LTA) task does not stop at the next action, but aims to predict a whole sequence of up to 20 future actions [80] [34]. Due to the longer time horizon, the focus is not on anticipating all the details of the interactions, but rather on capturing the correct sequential order of the actions, which are formulated as verb-noun pairs (e.g., first “take dough”, next “knead dough” etc.).

Action Anticipation task can be categorized into egocentric and third person action anticipation based on the camera perspective. Egocentric action anticipation utilizes first-person video input, capturing the scene from the viewpoint of the acting individual, thereby providing direct visual and contextual cues about the person’s intent, hand movements, and interactions with objects [21]. This perspective is particularly useful for applications such as augmented reality. In contrast, third person's action anticipation relies on observing actions from an external viewpoint. This perspective is beneficial for developing assistive/mobile robots. The following sections will be a brief overview of various existing methods covering both types of action anticipation tasks.

3.1.1 Egocentric Action Anticipation

Action Anticipation using latent goal learning

Roy et al. [60] proposes a model that uses an abstract representation of the goal, termed a latent goal, to anticipate the next action. The concept of a latent goal has been previously applied in areas like pedestrian intent detection [5] and trajectory prediction [61], where it represents a single intent like reaching a destination. In complex activities like cooking, however, the original authors describes the latent goal can represent a sequential intent as the person interacts with various objects sequentially. The proposed action anticipation model effectively uses latent goal information. It employs a *latent goal* representation as a proxy for the *real goal* of the action sequence and uses this information to predict the next action. The model is designed to compute the latent goal representation from the observed video and use it in conjunction with the observed visual representation to predict the next action and its potential resulting visual representation. Their solution involves using stacked LSTMs (Recurrent Neural Networks), which can create levels of abstraction from input observations across different timescales. A stacked LSTM, with a number of layers related to the average number of actions per video in a dataset, is used. Each layer can be thought of as representing an intermediate action leading to the latent goal. The prediction of the next action depends on both the observed visual feature and the latent goal. Because multiple actions might be plausible next steps, an LSTM is used to sample possible action candidates. They train the model using three loss functions: Anticipation loss, observation loss & Goal Consistency loss. The losses are applied to the chosen candidate during training to ensure the model learns to follow the defined properties.

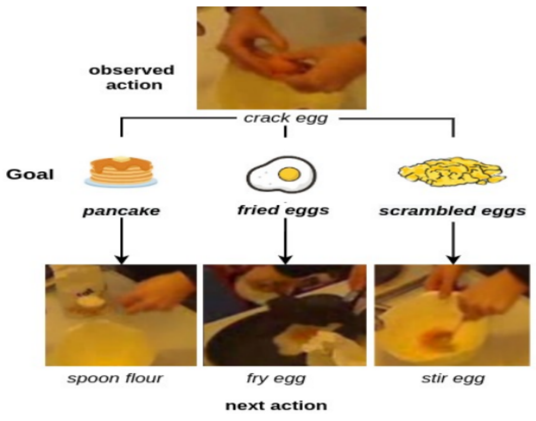


Figure 3.1: Next action anticipation depends on the underlying goal [60].

Anticipative Video Transformer (AVT)

Girdhar et al. [18] is an end-to-end video modeling architecture specifically designed for anticipating future actions from observed video. Unlike the above described methods, AVT leverages a purely attention-based transformer architecture to process video sequences, allowing it to maintain the sequential progression of observed actions while capturing long-range dependencies crucial for accurate anticipation. The AVT architecture consists of a two-stage process. First, a backbone network operates on individual frames or short clips of the input video, extracting a feature representation for each frame. While various video backbones can be utilized, the paper proposes an alternate backbone, AVT-b, which adopts the Vision Transformer (ViT) architecture. This backbone processes each frame to generate a sequence of feature vectors representing the observed video. Second, a head architecture takes these frame-level features as input and employs causal attention modeling to predict future actions and future frame features. This describes the model predicts the future based solely on the information from the frames observed so far. The entire AVT model is trained jointly to predict the next action in the video sequence while simultaneously learning frame feature encoders that are predictive of successive future frame features. This training is inspired by self-supervised learning objectives, encouraging the model to learn representations that are inherently predictive.

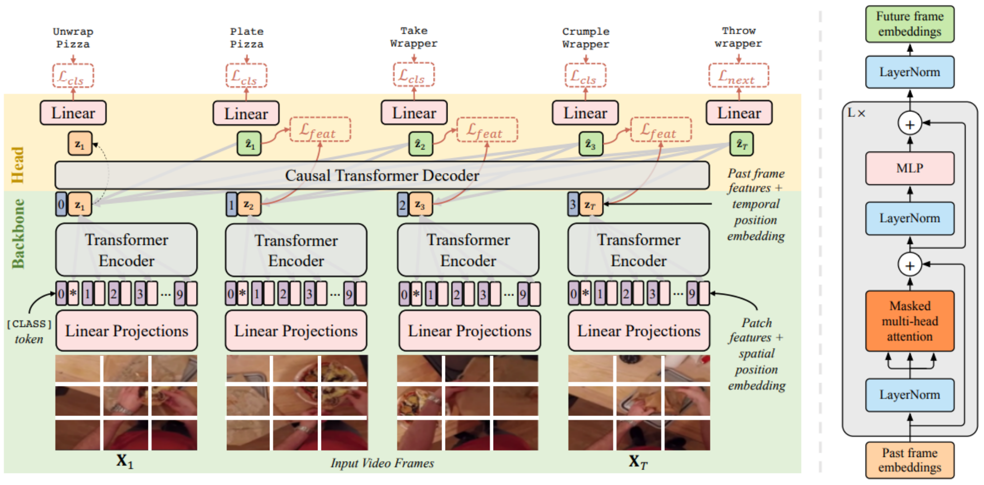


Figure 3.2: Anticipative Video Transformer [18].

AntGPT: Can Large Language Models Help Long-term Action Anticipation from Videos?

Zhao et al. [80] is a framework for long-term action anticipation (LTA) that leverages LLMs by representing egocentric video observations as sequences of human actions (verb-noun pairs) obtained from an action recognition model. The architecture adopts a dual approach–bottom-up and top-down. The bottom-up approach focuses on modeling temporal dynamics to predict the next actions one at a time. The top-down approach, on the other hand, first determines the person's overall goal and then predicts the actions necessary to achieve that goal. First they extract visual features from the video segments using a pre-trained vision backbone model, such as CLIP [54].

Subsequently, an action recognition model (implemented as a Transformer encoder) processes these visual embeddings to generate discrete action labels in the form of verb-noun pairs. In bottom-up approach, the sequence of recognized action labels is fed into an LLM, which is then tasked with autoregressively predicting the subsequent actions as a sequence completion problem. This allows the LLM to leverage its pre-trained knowledge of common action sequences and transitions to forecast future behaviors. In top-down approach, first they provide the recognized action labels to LLM which generates a textual-description of the inferred goal. This inferred goal can then be used to condition the prediction of future actions, effectively performing goal-conditioned procedure planning.

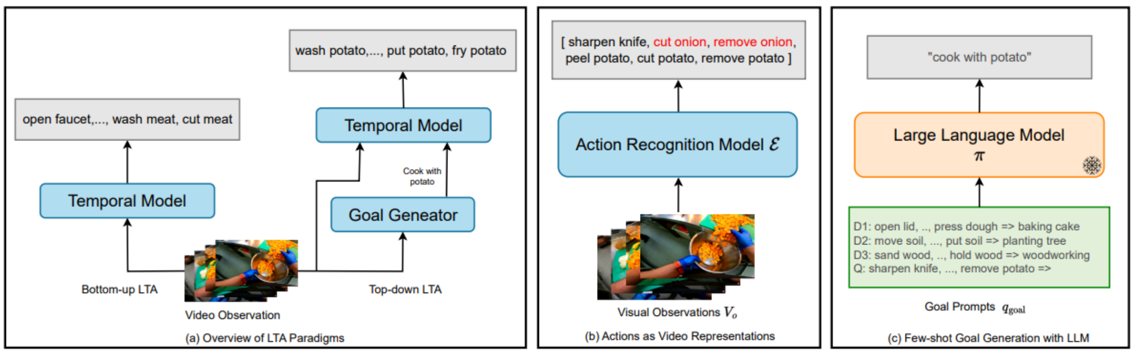

wash potato,..., put potato, fry potato
open faucet,..., wash meat, cut meat
Temporal Model
Temporal Model
Cook with potato
Goal Geneator
Bottom-up LTA
Top-down LTA
Video Observation
(a) Overview of LTA Paradigms
[sharpen knife, cut onion, remove onion, peel potato, cut potato, remove potato]
Action Recognition Model $\mathcal{E}$
Visual Observations V_o
(b) Actions as Video Representations
"cook with potato"
Large Language Model π
D1: open lid, .., press dough => baking cake
D2: move soil, ..., put soil => planting tree
D3: sand wood, .., hold wood => woodworking
Q: sharpen knife, ..., remove potato =>
Goal Prompts q_{goal}
(c) Few-shot Goal Generation with LLM

Figure 3.3: AntGPT architecture describing dual approach and Inferring with LLMs [80]

PALM: Prediction Actions with Language Models

Kim et al. [34] also a novel framework designed for LTA in egocentric videos. PALM leverages the power of Vision-language models (VLMs) and LLMs to forecast future action sequences over extended periods. Its architecture comprises three key modules as shown in Figure 3.4: an image captioning module(VLM), an action recognition module(ARM), and an action anticipation module based on LLMs. The method starts with an untrimmed egocentric video as input. Analyzing the recorded video up to a given moment, the action recognition module (ARM) creates labels for the past actions. Usually representing the sequence of actions the camera wearer performs, these labels are verb and noun pairs. PALM uses a sliding input video window of several consecutive action segments (empirically determined as four) to capture the temporal context, and aggregating predictions across many overlapping windows determines the top-1 action for every segment.

For the action recognition task, PALM employs a frozen video encoder from EgoVLP, a vision-language model pre-trained on egocentric data, followed by transformer layers and classification heads that are trained to predict the verb and noun components of the actions. Simultaneously, the image captioning module, which utilizes a frozen vision-language model (VLM), extracts relevant environmental details and context from the input video. Specifically, PALM formulates image captioning as a question-answering problem to enhance the descriptive power of the generated captions. By posing questions related to key concepts such as intention, interaction, and prediction to the VLM (empirically found to yield informative descriptions), PALM generates textual narrations (visual context) that complement the discrete action sequence. PALM employs VideoBLIP, a VLM fine-tuned on the Ego4D benchmark, for this image captioning module. The core of PALM lies in its action anticipation module, which uses an LLM such as LLaMa2 [68]. By providing the recognized actions and the caption, they prompt the LLM to forecast future actions. PALM designs a prompt that combines the recognized past action sequence and the generated textual narration. To enhance the LLM's predictive capabilities, PALM utilizes In-Context Learning (ICL) [6] by incorporating a few relevant examples from the training data into the prompt. The selection of these examples is crucial and PALM employs a Maximal Marginal Relevance (MMR) strategy to choose examples that are both semantically similar to the query and diverse.

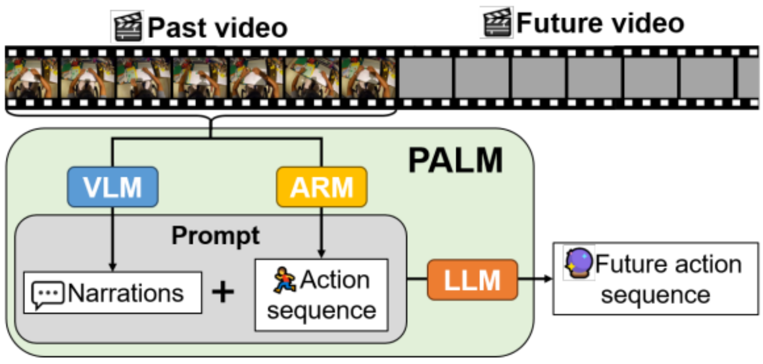


Figure 3.4: PALM architecture [34]

3.1.2 Action Anticipation from Third-Person View

When will you do what ?

Farha et al. [17] introduces two primary architectures, an Recursive Neural Network(RNN)-based and a CNN-based approach, for anticipating a considerable amount of future actions and their durations in videos. Both approaches follow a two-step process: First, inferring the activities from the observed part of the video, and then predicting the future activities based on these inferred past activities. For the first step, a hybrid RNN-HMM (Hidden Markov Method) approach is utilized to obtain a sequence of activity labels from the observed video frames. This inferred sequence of activities then serves as the input for future anticipation models. The RNN-based anticipation interprets future action prediction as a recursive sequence prediction task. The sequence of observed activity segments is sent to the RNN, which is represented by its normalized length and class label (as a 1-hot encoding). The RNN then predicts 3 outputs: the class label and length of the subsequent activity segment, as well as the remaining length of the final observed segment. The process is then repeated to make predictions farther into the future until the desired time horizon is reached, after which this predicted segment is appended to the observed sequence. Instead of using a recursive approach, CNN-based anticipation seeks to predict every future action in a single step. The inferred activity sequence is encoded into a matrix, with rows denoting temporal segments and columns denoting action classes. Depending on the intended prediction horizon, one may choose between RNN and CNN. RNN performs better for shorter horizons, while CNN is more reliable for longer predictions.

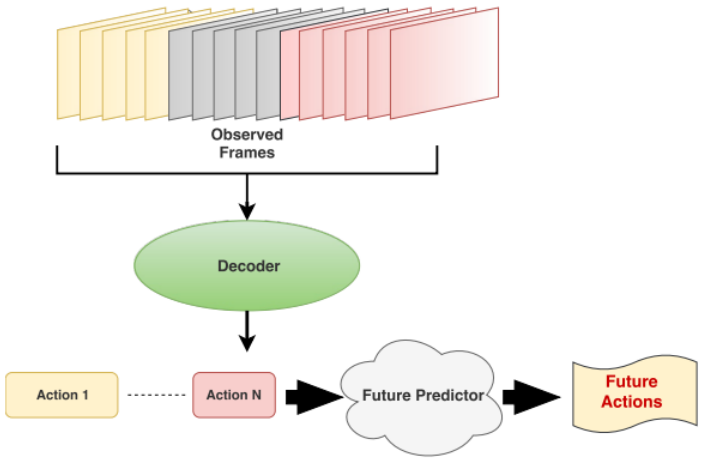


Figure 3.5: When will you do What? [17]

Geometrically Grounding Large Language Models for Zero-shot Human Activity Forecasting in Human-Aware Task Planning

Graule et al. [20] is a system intended for zero-shot human activity forecasting in human-aware task planning. As shown in Figure 3.7, the system operates in three sequential modules. This module gathers data from connected devices and observations from many sources, including video feeds, so producing a narration of past human activity. The next reasoning module's input comes from this narration; The foundation of GG-LLM is symbolic-geometric reasoning. It uses a pre-trained LLM, such LLaMA2 [68], to deduce, from past activity narration, the human's next most likely actions. One important factor is the geometrical grounding of these expected movements. GG-LLM connects the symbolically expected actions to particular places inside a semantic map of the surroundings. This is accomplished by asking the LLM to forecast where the human will travel next and subsequently mapping those forecasts to the semantic labels of objects in the surroundings. This stage is zero-shot, using the natural world knowledge of the LLM without requiring particular fine-tuning for activity prediction. This last module combines the spatial grounding, predictions of future human activities of the LLM, into a task planner for an assistive robot. This enables the robot to design its actions such that they consider the expected future behavior of the human, so reducing negative interactions between humans and robots.

Context-Aware Human Behavior Prediction Using Multimodal Large Language Models

Liu et al. [43] takes user instructions and historical visual observations as input and forecasts human behavior. The task is defined as predicting interaction labels, such as *verb-noun* pairs like *touch-table*, at a future time step (specifically 3 seconds in the future in the evaluation), based on the current visual scene context, historical interaction labels from the past and a set of ICL examples. An interaction label can represent human behavior as a list of multiple labels. The system's key components include a central MLLM module, input structure (which includes history of past interactions, visual input, captioning), prompt design using ICL, and an auto-regressive

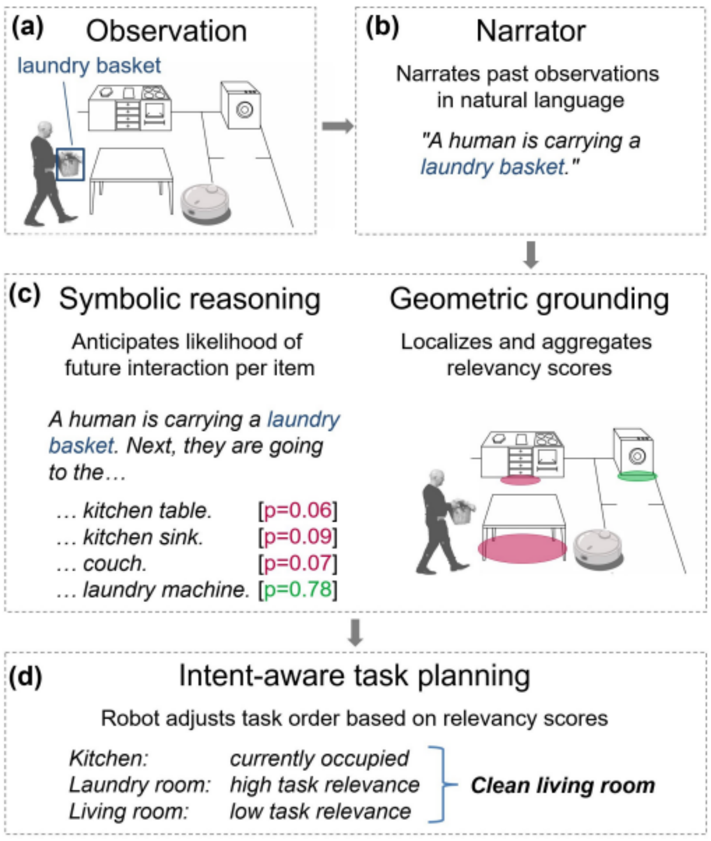

(a) Observation
laundry basket
(b) Narrator
Narrates past observations in natural language
"A human is carrying a laundry basket."
(c) Symbolic reasoning
Anticipates likelihood of future interaction per item
A human is carrying a laundry basket. Next, they are going to the…
… kitchen table. [p=0.06]
… kitchen sink. [p=0.09]
… couch. [p=0.07]
… laundry machine. [p=0.78]
Geometric grounding
Localizes and aggregates relevancy scores
(d) Intent-aware task planning
Robot adjusts task order based on relevancy scores
Kitchen: currently occupied
Laundry room: high task relevance
Living room: low task relevance
Clean living room

Figure 3.6: GGLLM-Architecture [20]

prediction step for intermediate predictions. Evaluation is performed on datasets derived from PROX and PROX-S, which provide RGB image sequences and semantic interaction labels from a third-person view in indoor environments.

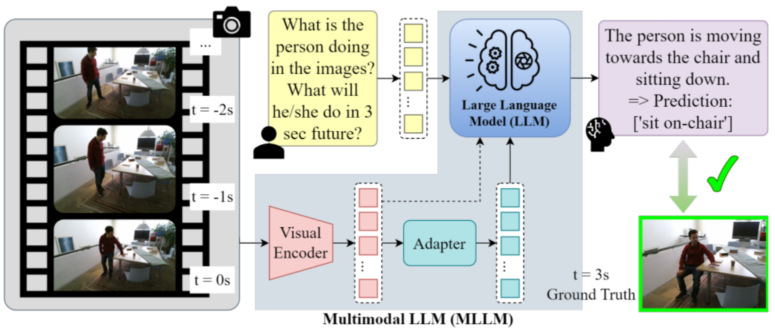

t = -2s
t = -1s
t = 0s
What is the person doing in the images? What will he/she do in 3 sec future?
Large Language Model (LLM)
The person is moving towards the chair and sitting down. => Prediction: ['sit on-chair']
Visual Encoder
Adapter
Multimodal LLM (MLLM)
t = 3s
Ground Truth

Figure 3.7: Context-Aware Human Behavior Prediction Using Multimodal LLMs [43]

3.2 Datasets

For this research, video representation of humans performing actions in indoor scenes is required. Existing datasets such as Ego4D [21] and EPIC-kitchens-55 [13] contain video scenes from an egocentric point of view. Ego4D as shown in figure 3.8 is the largest and most diverse egocentric (first-person) video dataset which has over 3,600 hours of densely narrated first-person video collected from around 926 unique participants (camera wearers) across 74 locations in 9 different countries and covers hundreds of real-world scenarios, including household, outdoor, workplace, and leisure activities [21]. EGTEA GAZE+[40] focuses on what a person is doing and where they are looking based on the analysis of video captured by a head-worn camera, which also mainly focuses on an egocentric point of view scenario. It contains approximately 28 hours of HD video Recorded from 86 unique cooking sessions involving 32 participants. [40]. There are a few existing datasets in kitchen scenarios such as Breakfast [37] and 50Salads [62], which focus on capturing videos from a third-person point of view, but they only consider a single human in a scene as shown in figure 3.9. The Breakfast dataset as shown in figure 3.9 contains 1,712 videos of breakfast preparation activities from a third-person view and recorded in 18 different kitchens, providing a variety of real-world environments [37]. Datasets such as Atomic Visual Actions (AVA)[22] contain clips from movies and TV shows which capture video scenes from a third-person point of view. The PROX dataset is designed for research in 3D human pose estimation within real-world indoor environments leveraging static 3D scene structure to improve the estimation of human body poses [82]. It contains video sequences of people performing everyday activities in furnished indoor rooms.

As the existing datasets are focused on egocentric and single human scenarios, there is a lack of dataset which contains multiple-humans in an indoor scene, hence a new dataset is generated which consists of multiple humans in the scene. This data is generated using a virtual simulator as described in Section 2.4. The details about the dataset are further mentioned in the Evaluation Chapter 5.

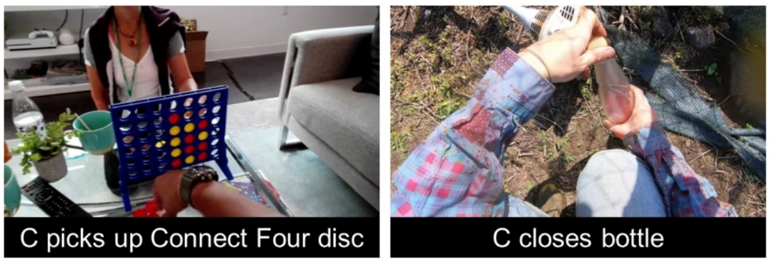


Figure 3.8: An example of Ego-4D dataset [21]

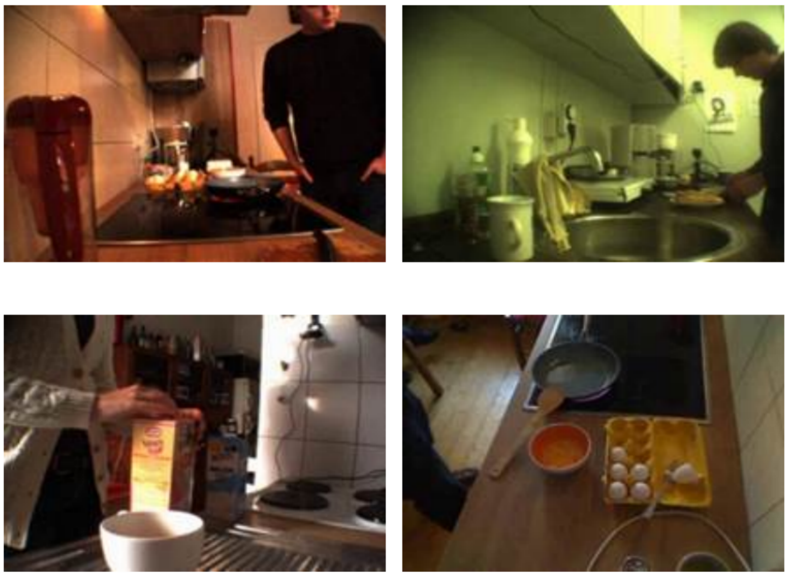


Figure 3.9: Breakfast dataset - Third Person View [37]

4 Methodology

This Chapter describes the proposed approach. Section 4.1 defines the task formulation for action anticipation. Section 4.2 outlines the system architecture and discusses its various components. The detailed two-stage training pipeline is presented in Section 4.3.

4.1 Task Formulation

The Action Anticipation task involves predicting a sequence of future actions for multiple individuals within a dynamic environment. Given an untrimmed video V, it consists of multiple frames. The video frames are denoted as $V_1, V_2, V_3, \ldots, V_T$, where T represents the total number of frames up to the time horizon T.

A Scene graph G contains node list N, where each node represents an object and a set of edges E which represents the semantic relationship between pair of objects. Given a scene graph G, the task is to predict N future actions starting from time step t: $A(t+1), A(t+2), A(t+3), \ldots, A_N$, where N denotes the prediction horizon in seconds, and A represents the predicted actions in natural language performed by each individual at a future time step.

The Future Action Anticipation (FA) task can be formally defined as the conditional probability:

$$\text{(4.1)} \quad \textbf{FA:} \quad P\big(A(t+1), A(t+2), \ldots, A_N \mid (V_1, V_2, \ldots, V_t), G\big)$$

where the objective is to model and predict the sequence of future actions based on the observed video frames and the scene graph representation.

4.2 System Architecture

This thesis presents VISTA, which stands for Vision and Scene aware temporal action anticipation, which leverages a VLM to predict interaction labels based on the given task formulation as shown in figure 4.1. Each core component of the system is described in detail below.

The input of VISTA contains a visual input in form of video data, scene graph & a prediction prompt and it outputs a list of predicted actions in natural language format. The central component of the system is a Fine Tuned Open Source Vision language model, which processes 2D video frames as contextual input to predict future action labels. The model fine tuning pipeline is described in the next section. The model operates based on natural language instructions provided in the prompt. Additionally, the prediction system requires a scene graph. Each component of the core prediction system is discussed in detail in the subsequent subsections.

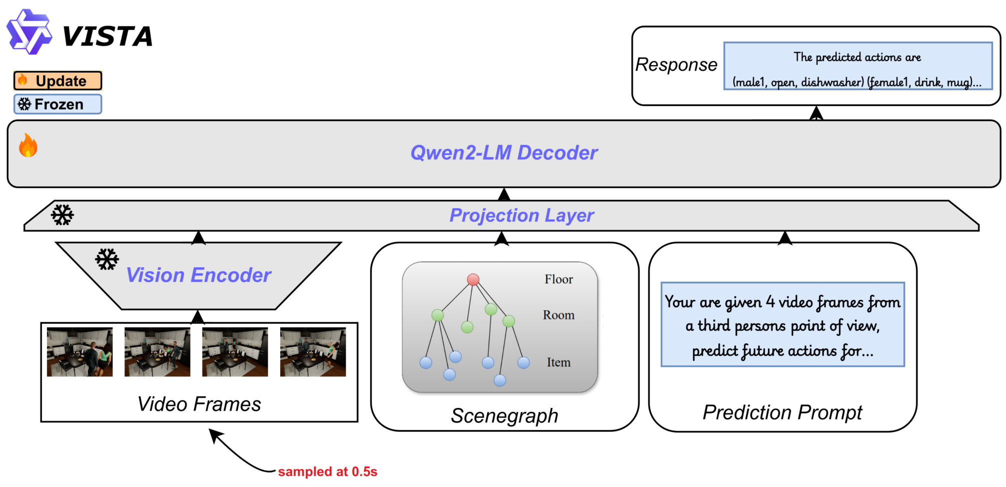

VISTA
Update
Frozen
Response
The predicted actions are
(male1, open, dishwasher) (female1, drink, mug)...
Qwen2-LM Decoder
Projection Layer
Vision Encoder
Floor
Room
Item
Your are given 4 video frames from a third persons point of view, predict future actions for...
Video Frames
Scenegraph
Prediction Prompt
sampled at 0.5s

Figure 4.1: VISTA Architecture: Given past video frames, a Scenegraph and a Prediction prompt, the fine tuned VLM generates predictions as textual representations.

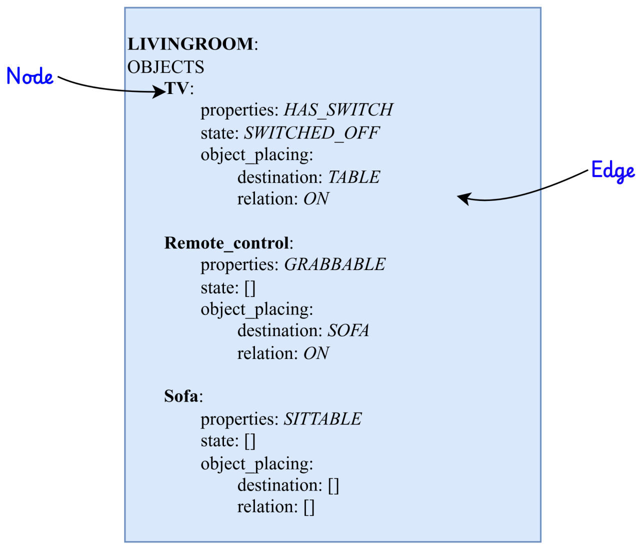
4.2.1 Visual Context

The visual context here is represented based on video frames sampled at 0.5 seconds. After an informal validation, 0.5 seconds of sampling technique effectively represents actions of each human in the scene .The video feed depicts humans in a scene. This work utilizes videos of multiple humans performing actions from three different scenes namely kitchen, bedroom and living room. The videos are synthetically generated using Virtual home [53]. Additionally, real world videos are recorded at Robert Bosch Campus in Renningen, Germany to test our pipeline on real world dataset. From the given visual context the VLM can extract information related to the body movements, the current position, available objects surrounding the agent, the number of agents, body orientation or gaze direction.

4.2.2 Scene Graph

A 3D scene graph (3DSG) is a modern 3D scene representation applied to graphically depict vast real-world areas. Armenian et al. [3] initially presented them as a hierarchical paradigm to link buildings, rooms, items, and people in several layers. After this introduction, Rosinol et al. [59] and Hughes et al. [31] investigated the 3DSG synthesis from sensor data. Other methods center on simulating semantic links between objects [70] [75] [72] from 3D point clouds. As LLMs emerge, 3DSGs also become open-vocabulary, providing a better knowledge of object relationships. 3DSGs have lately started to be included into robotic systems. Applications comprise task and motion planning [57] as well as navigation [24].

This work utilizes a 2D version of scene graphs for the following reason. First, Only a small number of recent studies have looked into integrating 3D data [19]; most VLMs currently use 2D inputs [71]. Secondly, It is more feasible and widely available to acquire scene data in 2D. For capturing 2D scene representations, a standard RGB camera which is frequently found in mobile robotic platforms is adequate. On the other hand, specialized sensors like LiDAR or depth cameras are usually needed for 3D data collection [5], and they might not always be accessible in deployment scenarios. While multiple versions of scene graphs exist, this work adheres to the version proposed by Liu et al. [44]. Scene graph contains information related to visual scene, describing objects, their properties, states, and spatial relationships in a readable format. Scene graphs contains a node list **N** where each nodes describes an object and for each object has properties, state and an edge. Properties define a way to interact with the object, state defines the current state of the object while edge defines the current location of the object with respect to another object. Figure 4.2 describes an example scene graph structure of living room, where objects and their properties are defined. The scene contains a TV, which has a switch, as indicated by the *HAS_SWITCH* property. The state of the TV is *SWITCHED_OFF*, which means that it is currently turned off. Additionally, the scene graph encodes spatial relationships, specifying that the TV is placed on a table using the *object_placing* field, with *destination* and *relation* describing the location.



```
LIVINGROOM:
OBJECTS
  TV:
      properties: HAS_SWITCH
      state: SWITCHED_OFF
      object_placing:
          destination: TABLE
          relation: ON

  Remote_control:
      properties: GRABBABLE
      state: []
      object_placing:
          destination: SOFA
          relation: ON

  Sofa:
      properties: SITTABLE
      state: []
      object_placing:
          destination: []
          relation: []
```

Figure 4.2: Scenegraph format

4.2.3 Prompt Design

It is necessary to provide a concise prompt as it contains general rules for the model to take care of while generating the output. The prediction prompt guides the model in its training and inference process which is designed to specify the task context and expected output format. In this thesis, prompt structure is created which includes the following information.

1. **Role:** "*You are expert in ...*"
2. **Overall Task:** "*Your task is to predict future human actions provided ...*"
3. **History Length:** The length of the provided history video frames
4. **Scene Graph Information:** Included with each prompt.
5. **Prediction Horizon:** The amount of future prediction labels to generate.

6. **Output Type:** Typically a python list
7. **Prediction Format:** The format of each prediction label as described in the Section 4.2.4.

By including this information in prompt, the model is conditioned to interpret the input and generate output in a relevant manner. The prompt is structured in such a way that it ensures clarity in the model's objective. The below listing shows how the prompt is constructed for this thesis.

```
You are expert in predicting human actions provided video frames.
You will be given {histlen} video frames as history input. The video frames describes humans
performing actions.
Your task is to predict {futrlen} future human actions based on the information provided in
the video frames.
You will be given scene graph which describes the properties of each object in the environment
.
The scenegraph of the environment is given below
{scenegraph}

The output should be the {futrlen} future action labels. An Action label for one character is
'(charid, action, objects)' and if there are two characters the action label is
'(charid1, action, objects) (charid2, action, objects)' and so on.

Understand the information from the video frames and generate {futrlen} future action labels,
each action label should be separated by a comma.
```

4.2.4 Prediction Representation

Human behavior can be represented in various ways, and to achieve a certain level of standardization, it is often encoded using **verb-noun pairs**. This approach is commonly employed in tasks such as action anticipation, as described in the previous section. When the behavior label specifically focuses on human-scene interactions, these verb-noun pairs can be expressed in the form of action-object pairs, such as *(sit, chair)* or *(drink, mug)*, which explicitly capture the interaction between the agent and the object.

This thesis adopts a textual verb-noun pairing scheme similar to prior work in AntGPT [80] and PALM [34]. However, previous studies primarily focused on scenarios involving a single human agent, meaning that their representations lacked a mechanism to specify which agent was performing a given action. To address this limitation, this work introduces an agent-specific representation by utilizing a character identifier (charid) in conjunction with the verb-noun pair. This thesis utilizes the format **charid-verb-noun**. The charid serves as a reference to a specific agent performing the action, allowing for a more precise and scalable representation of multi-agent environments. In this architecture, charid values are assigned based on the agents present in the scene, such as male1, female1, male2, and so on, depending on the number of individuals in the environment.

4.3 Fine Tuning Pipeline

This work introduces a two stage training pipeline as described in the Figure 4.3. The first stage consists of Supervised Fine Tuning (SFT) Method and the second stage consists of Direct Preference Optimization fine tuning method. This section will outline the detailed training method.

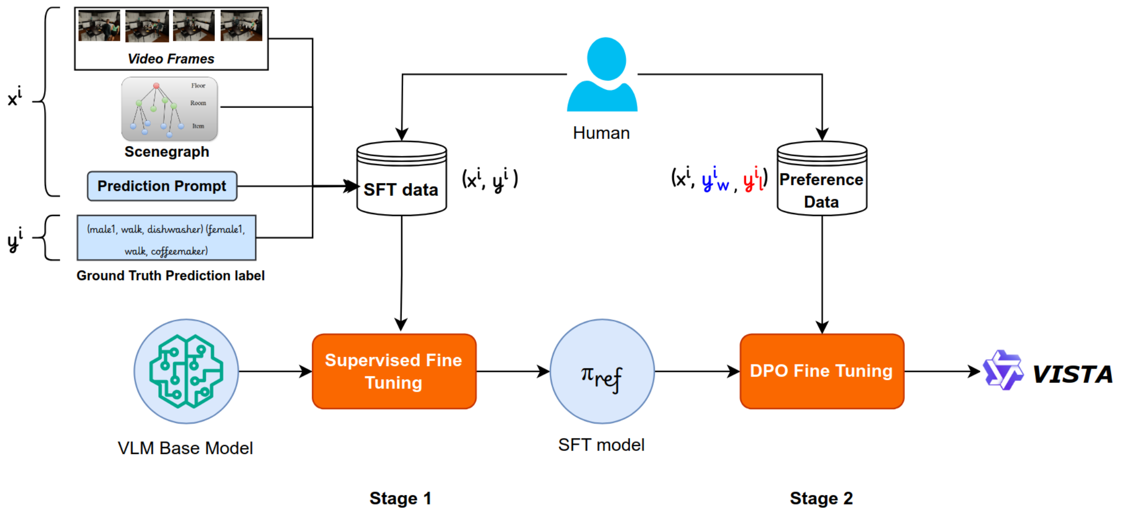


Figure 4.3: The two-stage training pipeline. Stage 1 is the Supervised Fine Tuning method, and Stage 2 is the Direct Preference Optimization-based Fine Tuning method.

4.3.1 Stage 1: Supervised Fine Tuning

For supervised fine-tuning, a pretrained open-source VLM is employed, and Supervised Fine-Tuning with LORA [28] approach is implemented. The open-source model is fine tuned pretrained using data produced by Virtual Home as described in Section 2.4. The fine-tuning dataset comprises image-text pairs. The data format is “Observed Images, Scene Graph => Future Actions”. A Visual input, scene graph and prediction prompt is provided as input, and the model generates text labels as output. The selection of VLM and SFT hyperparameters is addressed in the Evaluation Chapter 5.

A dataset consisting of input-output pairs for SFT fine tuning is typically referred as SFT data. Given a SFT data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the input to the model which contains a video frames, a Scene graph and a prediction prompt, the goal of Supervised Fine-Tuning (SFT) is to adapt a pre-trained language model f_{θ_0} to match the distribution of $\mathcal{D}$. Instead of updating all weights θ_0, LoRA fine tuning method is used as explained in Section 2.3.1 where only low-rank matrices are trained to reduce computational cost. The core idea of LoRA is to freeze the original model weights θ_0 and only train the selected layers. In this thesis, during training, a set of layers are explicitely defined to which LoRA adapters are applied.

During training, the loss is computed as the negative log-likelihood over the dataset:

$$\mathcal{L}(\phi) = -\sum_{i=1}^{N} \log P_{\phi}(y_i \mid x_i)$$

where $\phi = \theta_0 + \Delta\theta$ includes the frozen base parameters and the trainable low-rank components($\Delta\theta$). Only A and B are updated, keeping θ_0 unchanged.

After the training, it is possible to merge the low-rank adapters $\Delta\theta$ into the base model or it can also be kept separate to allow dynamic model switching [28].

4.3.2 Stage 2: Direct Preference Optimization

In the second stage of the pipeline, the model is fine-tuned using the Direct Preference Optimization approach. For the DPO fine-tuning, the model checkpoints from Stage 1 are first loaded. As the model has already seen the training data, overfitting on the training data is prevented by using a low number of epochs. The detailed parameters are explained in the Evaluation Chapter 5.

For the DPO fine tuning, preference data is provided as seen in Section 2.3.2. The preference dataset D have format $(x^{(i)}, y_w^{(i)}, y_l^{(i)})$. $x^{(i)}$ is the prompt provided to the model which consists of video frames, a Scene graph and a prediction prompt, $y_w^{(i)}$ is the chosen predicted label and $y_l^{(i)}$ is the rejected predicted label. The method to obtain preference data is explained in the the following subsection.

The core of DPO is its loss function, derived from the KL-constrained reward maximization objective used in RLHF but reformulated to directly optimize the policy [56]. The DPO loss function for a preference dataset $D = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^{N}$ is:

$$L_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

where:

- $\pi_\theta(y|x)$ is the current policy (the language model being trained) that assigns a probability to response y given prompt x.
- $\pi_{\text{ref}}(y|x)$ is the reference policy.
- β is a hyperparameter that controls the strength of the implicit reward and the deviation from the reference policy (similar to the KL divergence penalty in RL).
- $\sigma(z) = 1/(1 + \exp(-z))$ is the sigmoid function.
- The term $\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ represents the log ratio of probabilities assigned by the current policy and the reference policy to a given response.

DPO Preference Data Building Process

As seen in Section 2.3.2, a preference dataset is required. The detailed data building process is shown in Figure 4.4. Each instance of preference data consists of a preferred and a non-preferred response. To collect this preference data, multiple outputs are generated from the model initialized from the SFT checkpoint. Outputs are collected from 10 different episodes. From these, responses from two episodes are selected based on the Edit Distance criterion. More about the Edit Distance metric is explained in the Evaluation Section. Once the preferred and non-preferred outputs are selected, the preferred output is manually checked for any errors. Minimal modifications are made to preserve the integrity of the original output generated by the SFT model. This is the only stage where human intervention is required.

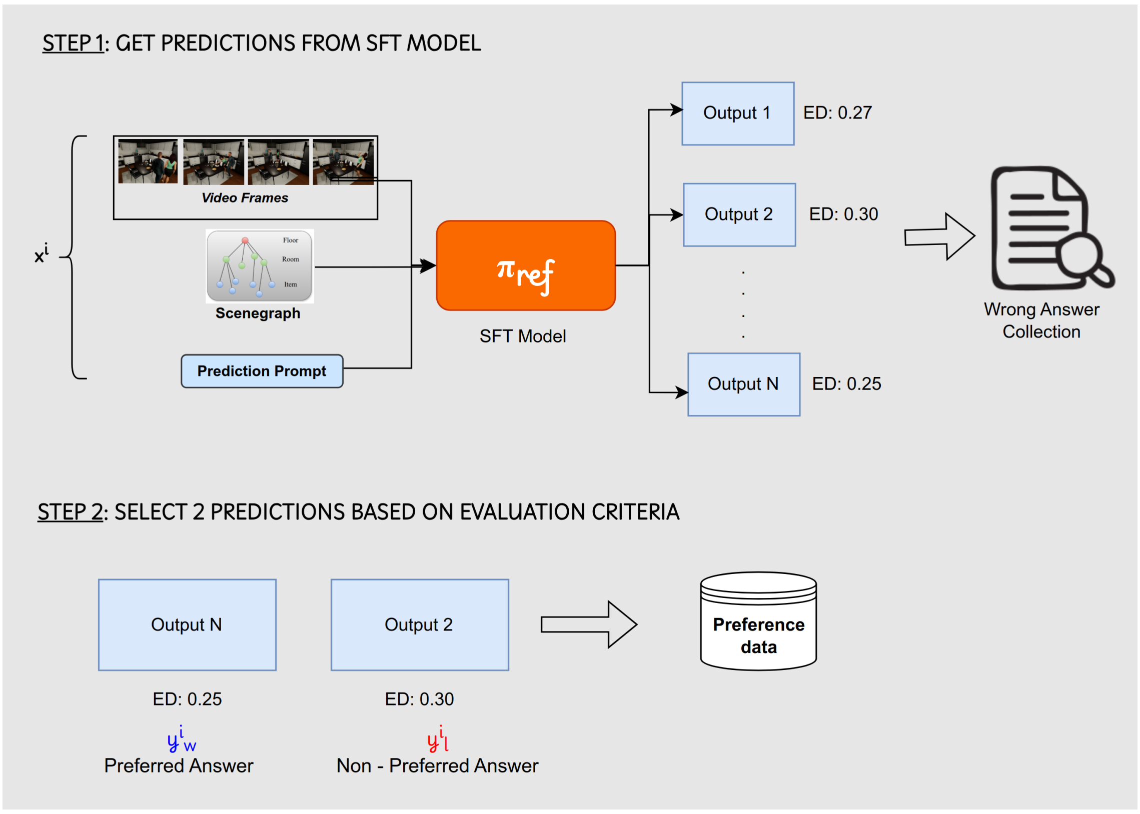


Figure 4.4: DPO Data Building Process: In the first step multiple sample of outputs are collected by loading the model from SFT checkpoint. In the second step outputs from two different episodes based on evaluation metric are collected and named it as Y_{win} (preferred) and Y_{lose} (non-preferred)

After finishing the two-stage training process, the trained model is used for inference, as shown in Figure 4.1. During inference, the model receives video frames along with the corresponding scene graph as input, and outputs the predicted textual activity labels.

5 Evaluation

This chapter will discuss the results from the pipeline described in the previous chapter. The first section describes the type of Models used in 5.1, which also details the hyperparameters used for training the models. Further details about our dataset is discussed in 5.2 classifying synthetic and real world datasets. Then the baselines are discussed in Section 5.3. After introducing Metrics in 5.4, n in depth Quantitative analysis is discussed in Section 5.6 where the proposed method is compared with the state-of-the-art baselines and also look at various ablation studies. At the end, we also look at the limitations of the approach in Section 5.7.

5.1 Model Selection

This section describes the models used in this study. Two family of models are selected; GPT-4 [50] that has state-of-the-art propriety models which excels in video understanding tasks and Qwen-VL [71] which is an open source family of models. In the GPT-4 family, both GPT-4o & GPT-4o-mini models are used and conduct several ablation studies. On this date, GPT-4o fine tuning is not allowed in Europe region, hence this work conducts ablation studies using In Context Learning approaches. In the open source family of models, Qwen2-VL is chosen as VLM which achieves state-of-the-art benchmarks in visual understanding tasks and it is also one of the top performing models in Hugging Face leaderboard [30].

5.1.1 Qwen-VL

Qwen2-VL [71] is a flagship vision-language model (VLM) that builds upon the Qwen series, demonstrating advancements in multimodal understanding and interaction. The architecture of Qwen2-VL is composed of three primary components which includes a LLM, Vision Transformer(ViT) and MLP-based language merger as explained in Section 2.1.2. This framework is designed to process multimodal inputs, including images and videos, at their native resolutions and frame rates as shown in Fig. 5.1. Qwen2-VL is available in three sizes: 3B, 7B, and 72B parameters which varies in their internal structure depending on the number of layers, hidden size and vocabulary size. This work specifically uses the 72B version and fine tune with method as described in Section 2.3.

SFT Implementation

This work adapts the Qwen-VL model [71] from the Hugging Face Transformers library [74] to handle video-format data given as sequences of image frames for supervised fine-tuning. To maximize training efficiency, the model is fine-tuned for 30 epochs using the AdamW optimizer [35] with fused Torch implementation (adamw_torch_fused). Constant learning rate scheduler is

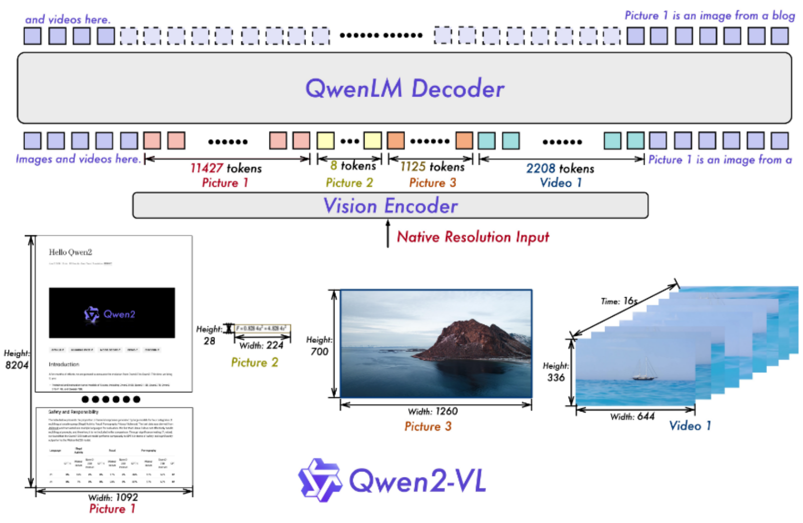


Figure 5.1: Qwen VL architecture [71]

applied and the learning rate is set to 3e-4 that remain constant over training process. Low-Rank Adaptation is used with a rank of 64 and a LoRA alpha of 128, added into the model's attention layers, to lower training costs while preserving performance. The remaining weights of the model are freezed as it adds more trainable parameters and it also enables effective adaptation free from overfitting the whole model [28]. The maximum number of newly generated tokens is limited to 512, and a temperature of 1.0 is used to generate more diverse responses and to collect output from multiple episodes. Bfloat16 (bf16) precision is enabled for reduced memory usage and faster computation, and TensorFloat-32 (TF32) support for matrix operations on compatible hardware. All fine-tuning experiments were conducted on **H100** Tensor Core GPUs deployed at Robert Bosch GmbH at Renningen, Germany.

Hyperparameter	**Value**
Model	Qwen-VL 72B
Input Format	Scene graph & Video Frames
Output Format	String (list of predicted labels)
Epochs	30
Learning Rate	3e-4
Learning Rate Scheduler	Constant
Optimizer	AdamW (Torch Fused)
LoRA Rank (r)	64
LoRA Alpha (α)	128
Maximum New Tokens	512
Temperature	1.0
Gradient Accumulation Steps	8
Precision	bfloat16 (bf16), TensorFloat-32 (TF32)
Hardware	NVIDIA H100 GPUs

Table 5.1: Hyperparameter Settings for SFT with Qwen-VL

DPO Implementation

For the DPO fine tuning, the checkpoints are first loaded from SFT adapters and Hugging face Transformers library is used for training. [74]. As the model as already seen the dataset during the SFT training, number of epochs is kept to a lower value at 5 to avoid overfitting. AdamW optimizer [35] is used and also enable the gradient checkpointing for memory optimization during the training. Bfloat16 (bf16) precision is enabled for reduced memory usage and the gradient accumulation steps is set to 8 and the same hardware is used as mentioned in the above section. The table 5.2 describes all the hyperparameters that are used for DPO training.

Hyperparameter	**Value**
Model	Qwen-VL 72B
Input Format	Scene graph & Video Frames
Output Format	String (list of predicted labels)
Epochs	5
Optimizer	AdamW
Temperature	1.0
Gradient Accumulation Steps	8
Precision	bfloat16 (bf16)
Hardware	NVIDIA H100 GPUs

Table 5.2: Hyperparameter Settings for DPO with Qwen-VL

5.1.2 GPT-4o

GPT-4o is the latest family of AI models from OpenAI [50] which is a multimodal model meaning that they can natively handle text, audio and images. GPT4o-mini is a comparitively smaller model. It is faster and cheaper that GPT-4o, while still being more powerful than its predecessors like GPT-3.5 Turbo [50]. It is built upon a Transformer-style architecture [69]. The model is pre-trained to predict the next token in a document, using a massive Internet Data. Following the pre-training phase, GPT-4 undergoes fine-tuning using Reinforcement Learning from Human Feedback (RLHF). As GPT-4 is a proprietary model, the official reports do not provide any information about architecture, hardware, training method or compute power. Thus, the exact number of parameters or the precise details of the pre-training dataset and method are not disclosed. However GPT4o has limited context window which means that the model can only effectively consider certain amount of input (text, images or videos) at any given time. This study utilizes GPT4o and GPT4o-mini which are hosted on Azure Services at Robert Bosch GmbH.

This work performs several experiments using the two described models. For the in-context learning pipeline, GPT-4o is used as the VLM model, which is also one of the state-of-the-art models available in the VLM community. To achieve the best results while maintaining cost-effectiveness, experiments are also conducted using open-source VLMs like Qwen-VL, which are fine-tuned using the Supervised Fine Tuning method. While the main architecture in this thesis consists of a fine-tuned Qwen-VL model, results are also compared against GPT-4o as a baseline along with other state-of-the-art methods in later sections.

5.2 Dataset Description

As the existing datasets did not meet the requirements for multi-human actions within an environment, custom dataset using a virtual simulator is generated. The simulator, detailed in Section 2.4, comprises six distinct environments, each containing a bedroom, washroom, living room, and kitchen. 30 synthetic video sequences from these environments are generated, featuring a wide range of human actions across various scenarios. Each video has a duration of 20 to 30 seconds, recorded at a frame rate of 15 FPS, and consisting two agents interacting within the scene, with additional sequences for the living room and kitchen scenarios involving three agents to capture more complex interactions. A figure depicting human agents across three different scene types is provided in the Figure 5.2.

In addition to the synthetic video data, evaluation on real world scenarios is also conducted. 10 real-world video sequences are recorded featuring two and three individuals interacting within kitchen and common area environments as shown in Figure 5.3. These recordings were conducted at the Robert Bosch Campus in Renningen, Germany. The videos were captured using an iOS device, with each sequence ranging from 50 seconds to 1 minute in duration and recorded at a frame rate of 30 FPS. The details about the videos and their properties are shown in Table 5.3. Figure 5.3 shows an example of video frames in the Kitchen and Common Zone scenarios. The faces are blurred for privacy purposes.

Type	Environment	#Humans	#Videos	Avg. Duration	Avg. Frames	#Actions
Synthetic	Kitchen	1	4	30s	2500	6
	Bedroom	1	3	23s	1728	4
	Living room	1	3	20s	1368	4
	Kitchen	2	4	30s	2966	13
	Bedroom	2	3	23s	1698	9
	Living room	2	3	20s	1368	9
	Kitchen	3	5	25s	3224	13
	Living room	3	5	23s	1677	9
Real world	Kitchen	2	4	60s	1015	9
	Comm Zone	2	2	32s	984	8
	Kitchen	3	2	50s	936	9
	Comm Zone	3	2	60s	1015	8
Total			40	1hr		

Table 5.3: Diverse characteristics of our dataset

5.3 Baselines

Two baselines are selected for comparison. The first is Context-Aware Human Behavior Prediction [43], which closely aligns with this research as it uses visual inputs to predict human behavior. The second baseline is AntGPT [80], a state-of-the-art method for action anticipation in egocentric

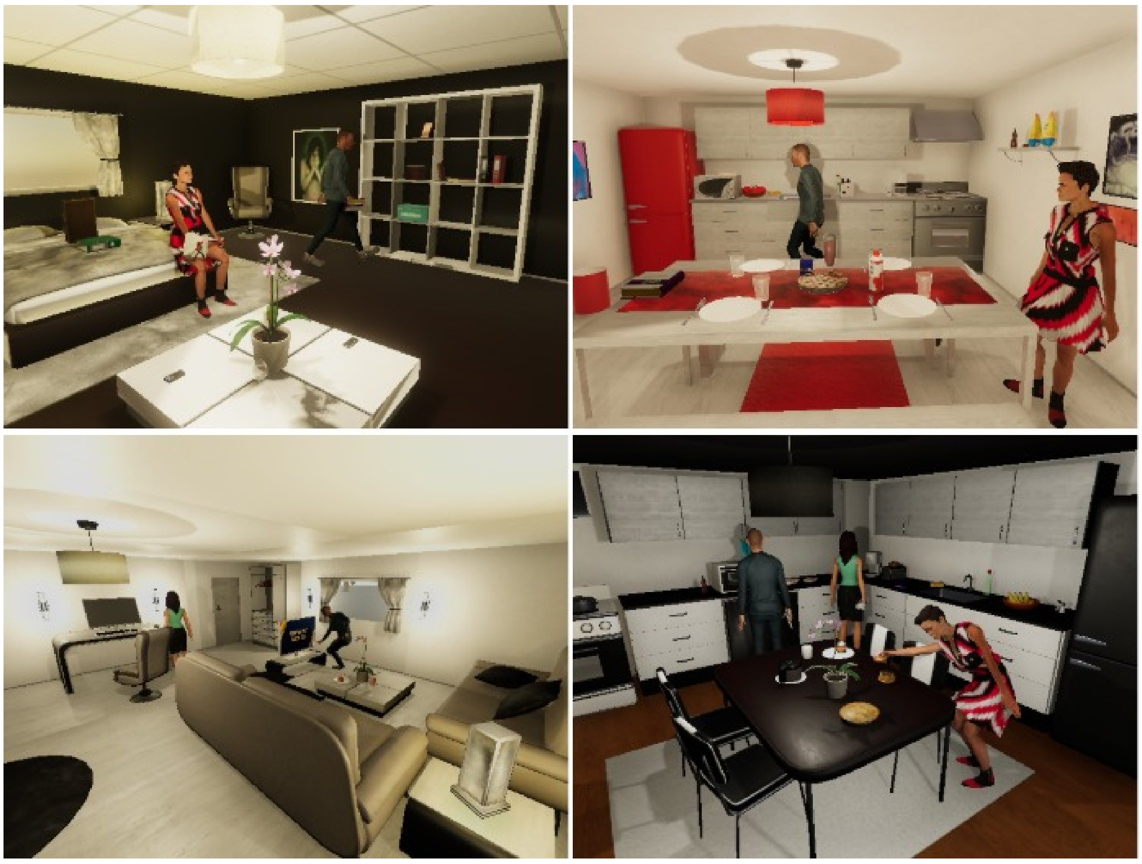


Figure 5.2: Sample video frames from our data. 1) The top left frame describes two agents in bedroom scenario. 2) The top right frame describes agents in Kitchen Scenario. 3) The bottom left depicts the livingroom scenario. 4) Bottom right is another kitchen scenario with three agents.

view. As baselines, only LLM-based approaches are considered. As previously discussed, traditional approaches based on CNN and RNN focus on egocentric action anticipation and a single human in the scene, hence their architecture is more suitable in single-agent environments. However, AntGPT and Liu et al [43] mentions that architectures are comparatively flexible with multiple-agent action anticipation, hence results can be produced using this dataset.

5.3.1 Context-Aware Human Behavior Prediction

The work Context-Aware Human Behavior Prediction (CAHP) by Liu et al. is used as a baseline.[43], which majorly focuses on the In-Context Learning (ICL) paradigm. GPT-4o is used as the main VLM, since the original work shows the best prediction accuracy using GPT-4o. Performance is also assessed for comparison using GPT-4o-mini. The experiments use an input-output pattern with structure "Observed Video Frames => Predicted Actions". The related findings are shown in the part on Analysis Section5.6.

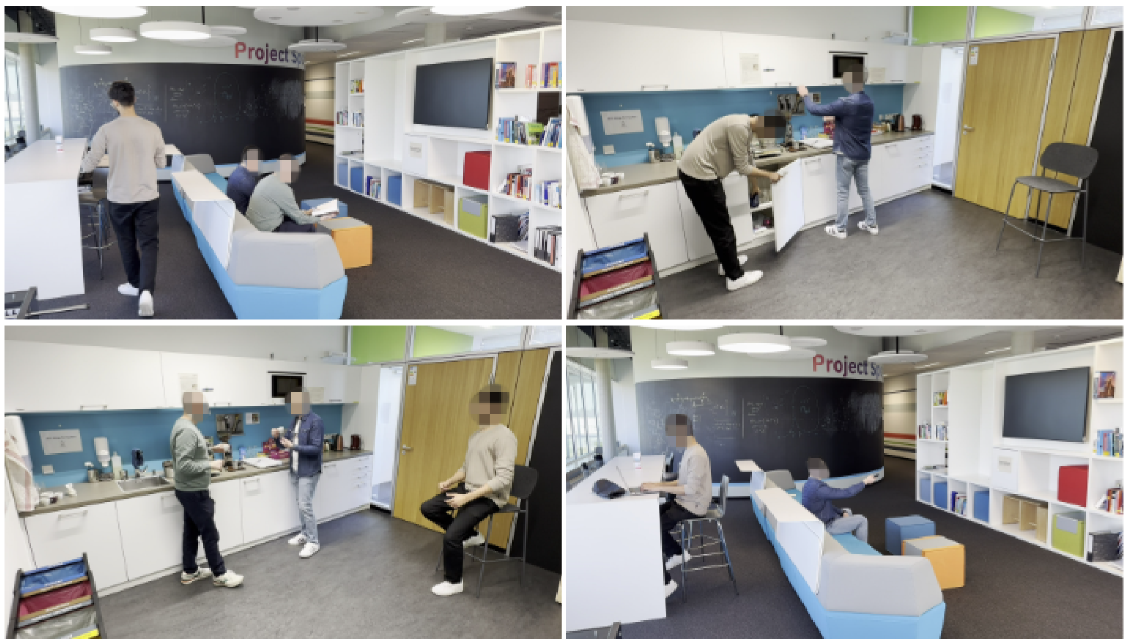


Figure 5.3: Sample video frames from our Real world recorded data in Kitchen and Communication Zone Area at Robert Bosch Campus in Renningen.

5.3.2 AntGPT

The open-source model LLaMa-7B model [68] is fine tuned for the AntGPT baseline using the parameters outlined in the original paper [80]. While AntGPT does not utilize third person views in their original work, it is included as a baseline as it is a state-of-the-art work in action anticipation and to evaluate its generalizability. The data structure "Observed Actions => Future Actions" is employed for input and output. The observed behaviors are separated as textual labels, as AntGPT initially transforms the visual frames into textual representations. AntGPT does not utilize Scene graphs hence they are not provided during fine-tuning. The dataset generated by Virtual Simulator already includes textual labels, which are used directly for fine-tuning AntGPT. The model is trained on Nvidia H100 GPUs at Robert Bosch GmbH, and the findings are discussed in Section 5.6.

5.4 Metrics

5.4.1 Accuracy

It computes an accuracy-like score by measuring the degree of overlap between the predicted and ground-truth interaction labels for a given future frame. This metric accounts for exact matches within the label sets. The score ranges from 0 to 1, where a value of 1 indicates a complete match between the prediction and the ground truth. The metric is formally defined as:

$$\text{Accuracy} = \frac{2 \times |P \cap G|}{|P| + |G|} \tag{5.1}$$

where P and G denote the sets of predicted and ground-truth interaction labels, respectively.

While accuracy metric provides an intuitive indication of exact prediction match, it is generally strict when applied to string comparisons, particularly in scenarios involving long strings or partially correct predictions. Small deviation, such as minor label mismatches or ordering differences, a missing comma or a bracket, can lead to disproportionately low accuracy scores. Therefore, it is important to take this metric as a low priority and consider alternative evaluation measures such as Edit Distance or Cosine Similarity which offer a more nuanced understanding of prediction performance, particularly in cases where partial matches are relatively important.

5.4.2 Edit Distance (Levenshtein Distance)

Another important metric for evaluating the similarity between predicted and ground-truth interaction labels is the *edit distance*, also known as the Levenshtein distance. It quantifies the minimum number of single-character operations insertions, deletions, or substitutions required to transform one string into another.

Formally, given two strings A and B, the Levenshtein distance $lev(A, B)$ represents the minimal cost needed to convert A into B.

$$\text{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \text{lev}(\text{tail}(a), \text{tail}(b)) & \text{if } \text{head}(a) = \text{head}(b), \\ 1 + \min \begin{cases} \text{lev}(\text{tail}(a), b) \\ \text{lev}(a, \text{tail}(b)) \\ \text{lev}(\text{tail}(a), \text{tail}(b)) \end{cases} & \text{otherwise.} \end{cases} \tag{5.2}$$

In this equation:

- head(x) denotes the first character of string x, i.e. $\text{head}(x_0x_1 \ldots x_n) = x_0$.
- tail(x) represents the string obtained by removing the first character of x, i.e. $\text{tail}(x_0x_1 \ldots x_n) = x_1x_2 \ldots x_n$.
- The notation $x[n]$ or x_n refers to the n^{th} character of string x, indexed from 0.

In the recursive case:

- The first term in the minimum corresponds to a **deletion** (removing a character from a).
- The second term corresponds to an **insertion** (adding a character to a to match b).

- The third term corresponds to a **substitution** (replacing a character in a with the corresponding character in b).

Edit distance is particularly valuable in scenarios where exact string matches are overly strict, as it accounts for minor differences between predicted and ground-truth sequences. Unlike simple accuracy measures, which treat any mismatch equally regardless of its extent, edit distance provides a graded measure of similarity based on the number of modifications required.

5.4.3 Cosine Similarity

Cosine similarity is a widely-used metric for measuring the similarity between two non-zero vectors in an inner product space. It is particularly effective in high-dimensional spaces and is commonly applied to compare feature vectors in tasks such as document similarity, embedding-based classification, and activity recognition.

Given two vectors A and B of dimension n, the cosine similarity $\text{sim}_{\cos}(A, B)$ is defined as:

$$\text{sim}_{\cos}(A, B) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.3}$$

Here:

- $A \cdot B$ represents the dot product of vectors A and B.
- $\|A\|$ and $\|B\|$ denote the Euclidean norms (or magnitudes) of vectors A and B, respectively.

The cosine similarity value ranges between -1 and 1, where:

- A value of 1 indicates that the two vectors are perfectly aligned (i.e. identical in direction).
- A value of 0 implies orthogonality, indicating no similarity.
- A value of -1 indicates that the two vectors are diametrically opposed.

Unlike discrete metrics such as accuracy or edit distance, cosine similarity captures the relative orientation of feature vectors in a high-dimensional space, making it particularly suitable for evaluating embedding-based representations where absolute values may be less informative than directional alignment. For our study this metric indicates that our fine tuned model output has a similar meaning to the ground truth. In other words, it serves as a validation that the model's predictions convey meaningful content rather than arbitrary or nonsensical values.

While this thesis report three evaluation metrics, Edit distance is used as a primary metric for evaluation considering it the most informative metric over exact accuracy and cosine similarity.

5.5 Training Performance

The Figure 5.4 below shows the results of the SFT loss for VISTA-72B, the training was conducted on NVIDIA H100 GPUs. Overall, the train loss shows a general downward trend, decreasing from 1.0 at starting of the training to 0.16 at the last epoch. There are also some oscillations during the training which is caused by temporary instabilities during the optimization process. The overall decrease in train loss suggest that the model is progressively minimizing the error on training data. The validation loss remains relatively stable which suggest that model is maintaining good generalization and it is not overfitting on the training data.

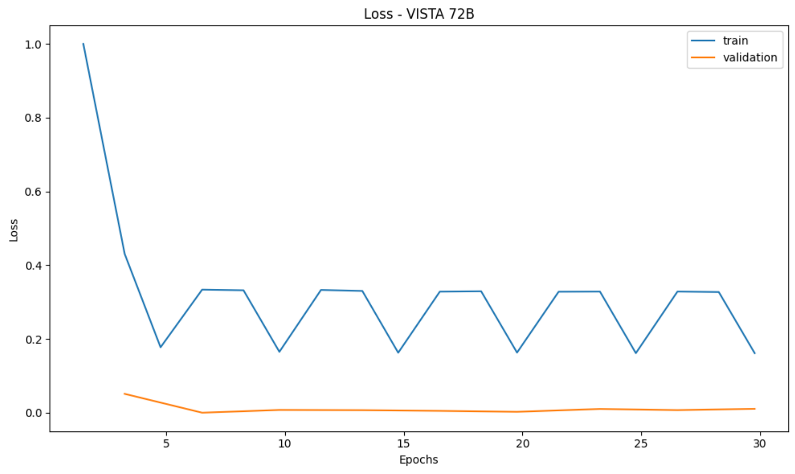


Figure 5.4: Training Performance - VISTA 72B

Figure 5.5 shows the loss for DPO when the preference data is provided. The results shows the loss of chosen and rejected answers from DPO fine tuning. The plot reveals that the chosen curve is consistently higher that the rejected curves. This suggests that model is learning to prefer the chosen responses over the rejected ones. The margin shows the difference between the chosen and rejected values. This is key metric that shows that the DPO algorithm aims to make the chosen response more likely. The positive margin indicates the stronger model performance for chosen responses.

5.6 Quantitative Analysis

In this section, quantitative results of the method are presented. At first, the method (VISTA) is compared with state-of-the-art results as discussed in the baselines above. In the next section, a detailed guide of ablation studies is also presented. In the ablation studies, the method is initially assessed with and without the inclusion of scene graphs, followed by an examination of the model's performance as the prediction horizon is increased. Subsequently, performance is assessed by increasing the number of training parameters, lastly analyzing the performance across multiple model versions. At the end, the limitations of the approaches are discussed.

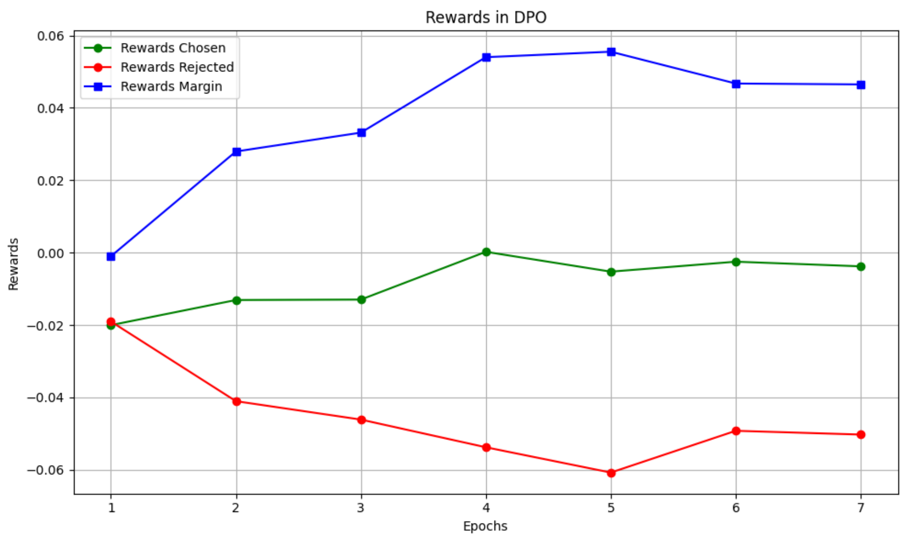


Figure 5.5: DPO Rewards Chosen and Rejected Loss

5.6.1 Comparision with State-of-the-Art

Table 5.4 shows the overall comparison of the method with state-of-the-art models. The method is compared in three different scenes with the baselines, and a consistent and significant performance advantage is observed over existing baseline models in all evaluated scenes.

Scene	Method	Accuracy ↑			Cos. Sim. ↑	ED ↓
		Full Acc.	Verb Acc.	Noun Acc.		
Kitchen	AntGPT[80]	0.053	0.108	0.246	0.961	0.503
	CAHP [43]	0.205	0.466	0.307	0.963	0.351
	VISTA-72B	**0.482**	**0.587**	**0.610**	**0.967**	**0.262**
Living Room	AntGPT[80]	0.079	0.342	0.173	0.953	0.349
	CAHP [43]	0.183	0.368	0.203	0.956	0.347
	VISTA-72B	**0.276**	**0.468**	**0.384**	**0.965**	**0.259**
Bedroom	AntGPT[80]	0.128	0.403	0.309	0.969	0.291
	CAHP [43]	0.213	0.453	0.318	0.967	0.297
	VISTA-72B	**0.398**	**0.568**	**0.513**	**0.965**	**0.203**

Table 5.4: Comparison with baselines with 2 agents in the scene from different environments of Virtual Home: Kitchen, Living Room and Bedroom

In the Kitchen scenario, the model exhibited the highest overall improvement achieving 0.482 Complete Accuracy, surpassing the next-best model CAHP by a substantial margin of 0.277.

In the Living room Scenario, although gains were more modest, VISTA led with 0.276 Complete Accuracy, outperforming CAHP-4o (0.183). The improvements in Noun Accuracy (+0.181 over CAHP-4o) shows better contextual prediction. Especially in the living room scenario, the dataset contains more diverse actions than Kitchen & Bedroom environments, here a significant increase is seen, when the actions are not repeated over a long time.

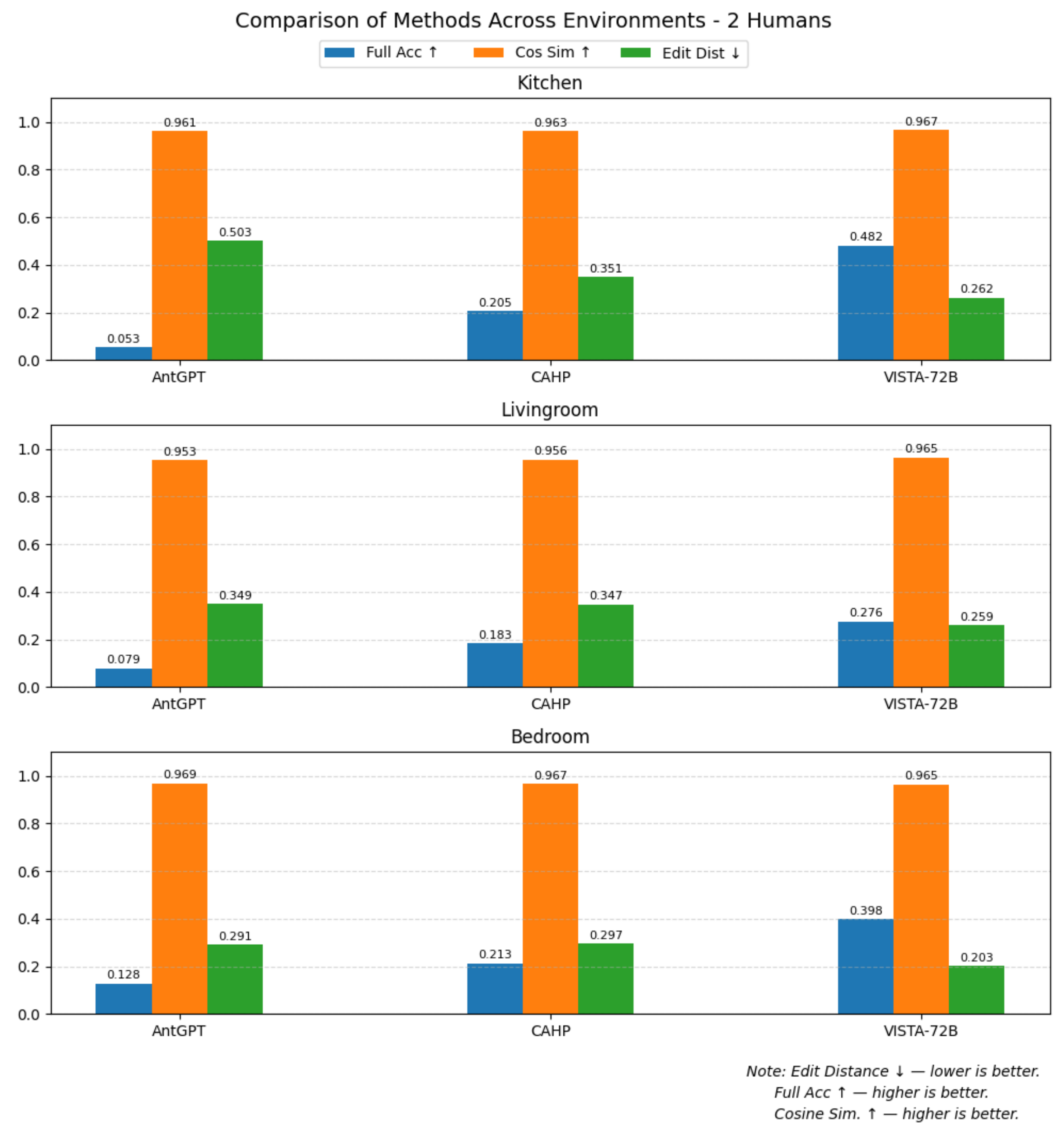


Figure 5.6: Metrics with two humans in scene

In the Bedroom Scenario, VISTA again led with 0.398 Complete Accuracy which is significantly better than CAHP-4o. Its Edit Distance of 0.203 was notably lower than all baselines, reflecting more precise scene label generation. As in the bedroom scenario there are more repeated actions, it can be seen that almost all the baselines are performing well in the Edit Distance Metric.

The Table 5.5 shows metrics with 3 agents in the scene. For this, the agents in the Kitchen and Living Room scenarios are considered. It can be seen that VISTA surpasses the CAHP and AntGPT architecture in all metrics. Compared to the previous table, as the number of agents increases, a drop in metrics is observed specifically, a drop of 25% in Edit Distance in the Kitchen scenario and a 17% drop in the Living Room scenario.

Scene	Method	Accuracy ↑			Cos. Sim. ↑	ED ↓
		Full Acc.	Verb Acc.	Noun Acc.		
Kitchen	AntGPT [80]	0.093	0.115	0.154	0.952	0.408
	CAHP [43]	0.137	0.309	0.320	0.962	0.374
	VISTA-72B	**0.301**	**0.473**	**0.439**	**0.963**	**0.328**
Living Room	AntGPT [80]	0.103	0.152	0.137	0.947	0.451
	CAHP [43]	0.172	0.286	0.301	0.961	0.324
	VISTA-72B	**0.227**	**0.395**	**0.316**	**0.963**	**0.305**

Table 5.5: Comparision of metrics with 3 agents in the scene

The Table 5.6 below shows the evaluation metrics from the real-world recorded videos. The model outperforms the previous baseline with a gain of +0.109 in Full Accuracy and a 17% improvement in Edit Distance in the Kitchen scenario. In the Communication Zone Scene, there is a gain of +0.105 in Full Accuracy and a 10% improvement in Edit Distance. Notably, there is a slight drop in the metrics when transitioning from synthetic evaluation to real-world video data because the model being majorly trained on synthetic data.

Scene	Method	Accuracy ↑			Cos. Sim. ↑	ED ↓
		Full Acc.	Verb Acc.	Noun Acc.		
Kitchen	CAHP [43]	0.318	0.324	0.308	0.953	0.395
	VISTA-72B	**0.427**	**0.432**	**0.396**	**0.965**	**0.324**
Comm. Zone	CAHP [43]	0.247	0.281	0.327	0.952	0.392
	VISTA-72B	**0.352**	**0.316**	**0.408**	**0.957**	**0.349**

Table 5.6: Metrics from Real World Data

5.6.2 Ablation Studies

Prediction without Scene graph

Inspired by Liu et al [43] GPT-4o[50] is utilized as the VLM in the In-Context Learning (ICL) pipeline. In this pipeline, past video frames are provided as input images, with the number of past frames determined by the parameter *historylen*. The parameter *historylen* defines the past horizon length, indicating the extent of historical human activity information included in the pipeline. *Predictionlen* indicates the prediction horizon which is the number of future action labels to predict based on the given past information .The dataset used in this particular experiment contains videos of two humans performing actions in a kitchen environment.

Notably, for this particular experiment, no scene graph information is incorporated. An ablation study is conducted by varying the number of ICL examples added, using values of 3, 6, and 7. Each ICL example consists of *historylen* video frames. In this experiment, *historylen* is fixed to 6, corresponding to 3 seconds of past human activity data. Consequently, for 3 ICL examples, the total number of input images, including the current input, is calculated as:

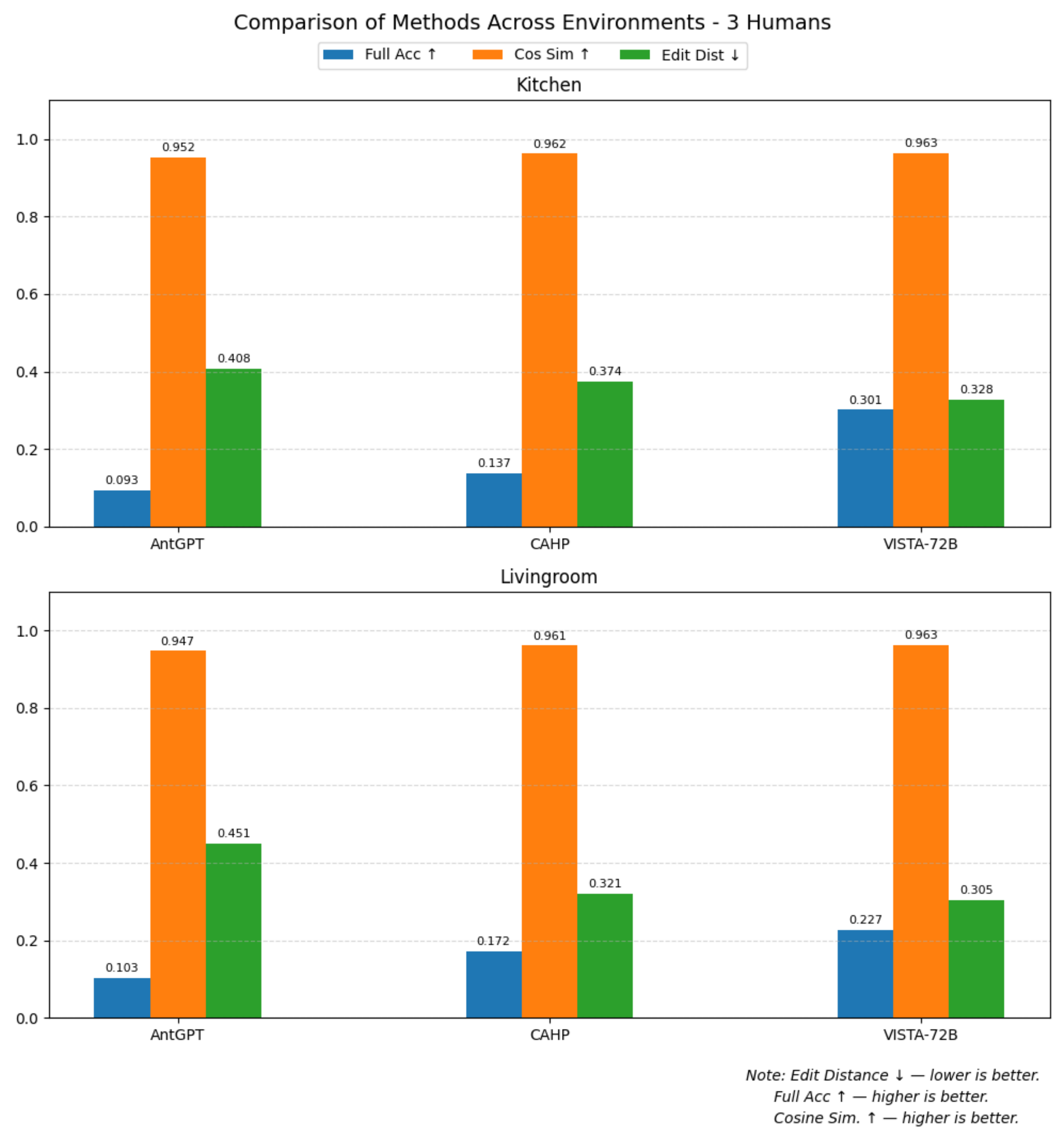


Figure 5.7: Metrics with three humans in scene

Total input images = $(3 \times \textit{historylen})$ + current input = $(3 \times 6) + 6 = 24$.

In GPT-4o and GPT-4o Mini, a maximum of 50 images can be processed as input due to computational constraints. When providing 7 ICL examples, the number of ICL images becomes:

$7 \times \textit{historylen} = 7 \times 6 = 42$,

with an additional 6 images from the current input, leading to a total of 48 images. Hence, with *historylen* set to 6, it is not possible to provide more than 7 ICL examples without exceeding the model's input limitations.

In this experiment, the predictive capabilities of GPT-4o are analyzed. Model performance is evaluated using three key metrics: Accuracy, Cosine Similarity, and Edit Distance (ED). Additionally, the inference time for each prompt is recorded. The results of this study are presented in Table 5.7

Num. ICL	Accuracy↑			Cos. Sim.↑	ED↓	Inference Time(sec)
	Full Acc.	Verb Acc.	Noun Acc.			
1	0.052	0.186	0.023	0.893	0.447	**9.50**
3	0.096	0.203	0.026	0.893	0.435	12.15
5	0.103	0.314	0.059	0.912	0.396	31.19
7	**0.105**	**0.318**	**0.072**	**0.918**	**0.398**	34.79

Table 5.7: Metrics without Scene Graph with 3-second prediction horizon

The results indicate that the pipeline struggles with action prediction accuracy across all configurations suggesting that very few of the predicted actions has an exact character level match with the ground truth actions. Increasing the number of ICL examples reduces edit distance but comes at the cost of significantly higher inference time which is not feasible for real world applications. The consistently low accuracy suggests that the model requires additional data or richer contextual information, such as scene graphs, to improve action prediction performance effectively.

Prediction with Scene graph

From the previous experiment, it is concluded that without incorporating scene specific information, the model fails to predict accurate actions. Upon analyzing the model's predictions in the pipeline described in the above section, instances were observed where actions were described as *drink dishwasher*, which is not a semantically valid action. Although the words *drink* and *dishwasher* appear within the scene, their combination does not yield a logical action. This highlights a key limitation in the model's understanding of contextual object affordances.

To address this issue, the model's predictions are enhanced by integrating environmental scene graph information. Scene graphs provide supplementary information by explicitly defining valid interactions between objects and actions. For example, the object *dishwasher* is constrained to a set of predefined actions such as *open, close, switch on, and switch off.* By embedding this structured knowledge, the model is expected to generate more contextually appropriate predictions.

In this experiment, the same pipeline as in the previous study is employed, but environmental scene graph information is incorporated to refine the model's action predictions. The following table 5.8 presents the results obtained from this study. The number of in-context learning (ICL) examples is systematically varied while maintaining a fixed past horizon and prediction horizon of six video frames, corresponding to a temporal span of three seconds. The ICL examples are varied across three experimental settings with values of 1, 3, 5, and 7.

These results demonstrate the impact of scene graph integration on action prediction accuracy. The inclusion of structured scene knowledge enables the model to constrain its predictions to semantically valid actions, thereby reducing the occurrence of illogical action-object pairings.

Num. ICL	Accuracy↑			Cos. Sim.↑	ED↓	Inference Time(sec)
	Full Acc.	Verb Acc.	Noun Acc.			
1	0.063	0.306	0.195	0.955	0.422	**11.12**
3	0.117	0.294	0.236	0.962	0.405	17.92
5	0.152	0.381	0.268	0.971	0.368	32.94
7	**0.183**	**0.462**	**0.305**	**0.971**	**0.355**	37.15

Table 5.8: Metrics with Scene Graph with 3-second prediction horizon

From the table 5.8, it is observed that the evaluation metrics show improved performance compared to previous results. As the number of ICL examples are increased, the metrics increases which shows a good sign. Due the maximum limit of GPT-4o, it is not possible to provide more than 7 examples. Even though it has a good performance in the edit distance, it is seen that as the ICL examples are increased, the inference time also increases as shown in figure 5.8 which leads to a longer processing time. This processing time is not suitable for real world applications that require low-latency predictions.

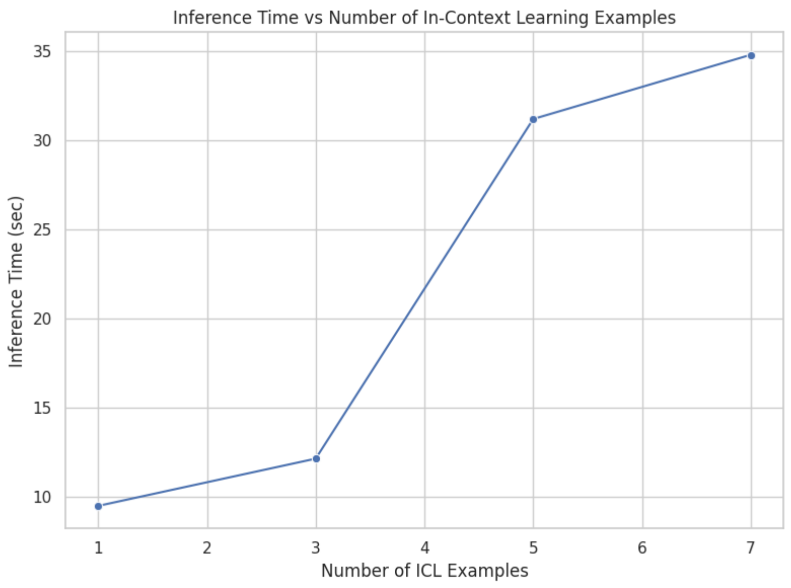


Figure 5.8: Inference time vs Num. ICL

Increasing Prediction Horizon

For this study, the focus is specifically on predicting future activity horizons, as shown in Table 5.9. The history length is set to a fixed value of 6, which corresponds to 3s, and the prediction length is systematically varied. The prediction horizon is varied across 2s, 3s, 4s, and 5s. The results show a consistent trend: the proposed pipeline achieves higher accuracy and lower edit distance scores for shorter prediction horizons. However, as the prediction length increases, performance degrades across both metrics. This suggests that while the pipeline is effective for short-term future prediction, its capability decreases for horizons exceeding 5 seconds.

Pred. Horizon(s)	Accuracy↑			Cos. Sim.↑	ED↓
	Full Acc.	Verb Acc.	Noun Acc.		
2s	**0.266**	**0.471**	**0.302**	**0.971**	**0.321**
3s	0.203	0.469	0.309	0.971	0.354
4s	0.145	0.293	0.213	0.965	0.410
5s	0.148	0.274	0.107	0.954	0.423

Table 5.9: Metrics with respect to Prediction Horizon

Increasing number of Humans in a Scene

The table 5.10 shows report metrics when increasing the number of humans in Kitchen and Living room Scenario. All the three accuracy metrics drops as the number of humans increases. The full accuracy in Kitchen scenario falls from 0.612 (1 human in scene) to 0.301(3 humans in scene), which indicates 50.8% in performance degradation. The cosine similarity remains stable which suggests that while the exact matches degrades, the model is generating semantically relevant actions. Similarly the edit distance increases which suggests that the model's predictions become less textually similar to the ground truth when the number of humans increase in the scene.

Scene	#Humans	Accuracy ↑			Cos. Sim. ↑	ED ↓
		Full Acc.	Verb Acc.	Noun Acc.		
Kitchen	**1**	**0.612**	**0.601**	**0.662**	**0.965**	**0.186**
	2	0.482	0.587	0.610	0.967	0.262
	3	0.301	0.473	0.439	0.963	0.328
Living room	**1**	**0.531**	**0.603**	**0.558**	**0.963**	**0.192**
	2	0.276	0.468	0.384	0.965	0.259
	3	0.227	0.395	0.316	0.963	0.305

Table 5.10: Metrics with respect to increasing number of humans with 3s prediction horizon

Increasing LORA parameters

For this study, the goal is to observe how the edit distance performs when increasing LoRA parameter values such as LoRA Rank (r) and LoRA Alpha (α). As the Qwen2-72B model is utilized, the total number of parameters is **74,247,697,920**. The values of LoRA Rank and Alpha are systematically varied, as shown in Table 5.11. Increasing these values leads to an increase in the number of trainable parameters, as explained in Section 2.3.1. The results indicate that increasing the number of trainable parameters has very little impact on the edit distance. Across experiments, variations in

LoRA parameter values resulted in only marginal changes, with a maximum edit distance difference of 0.1, which indicates that changing the value of LoRA parameters does not have a significant effect on the output.

Nevertheless, some research indicates that a high LoRA rank will raise the model's degrees of freedom and increase the possibility of overfitting the model on downstream tasks [41]. As trainable parameters increase, more capacity is introduced for the model to memorize the fine-tuning dataset. Increasing the value of LoRA parameters also gives the model more capacity to adapt to new data during supervised fine-tuning, but it leads to higher memory usage and longer training times. Studies suggest a rank of 32-64 for 70 billion parameter models [28]. Hence, for all fine-tuning experiments, the value of LoRA Rank is set to 64 and LoRA Alpha to 128, indicating that only 1.13% of the total model parameters are trained.

LoRA Rank (r)	LoRA Alpha(α)	Trainable Params	Trainable Params (%)	ED↓
16	32	210,534,400	0.28%	0.203
32	64	421,068,800	0.57%	0.192
64	**128**	**842,137,600**	**1.13%**	**0.189**

Table 5.11: Changing LORA parameters effect on Edit Distance with 3 seconds prediction horizon

Comparing different fine-tuning strategies

This ablation study evaluated the effectiveness of different fine tuning methods. As seen in table 5.12, the pretrained VLM model shows the weakest performance in multiple human behavior prediction. Notably, no In-context-Learning examples are provided to the base model. Some results also indicate that the model struggles with understanding the task and fails to generate correct output format. In the next step, model fine tuned using SFT only has better improvements across all metrics as compared to pretrained model. Significant enhancements are attained when the model is fine-tuned on both SFT+DPO in a complete training pipeline as described in the Section 4.3. This combined approach achieves best performance among all the individual configurations. Specifically, in the Edit Distance, 16% improvement is seen when fine tuning the models with SFT+DPO as compared to using SFT alone.

	Accuracy↑			Cos. Sim.↑	ED↓
	Full Acc.	Verb Acc.	Noun Acc.		
Pretrained	0.103	0.056	0.127	0.874	0.692
SFT	0.294	0.392	0.328	0.958	0.315
SFT+DPO	**0.482**	**0.587**	**0.610**	**0.967**	**0.262**

Table 5.12: Metrics comparing different Fine Tuning Strategies in Kitchen Scenario

Different Variants of Qwen-VL

In this section, the pipeline is evaluated using different variants of the Qwen-VL architecture, specifically the 2B, 7B, and 72B models, in the kitchen scenario, which contains more diverse actions. The results are shown in Table 5.13. Supervised fine-tuning is performed using the same parameters as described earlier. The results clearly indicate that the smaller models underperform relative to the 72B variant. This discrepancy may be attributed to the fact that, when trained on video data, smaller models struggle to capture dependencies across multiple frames. As model size increases, the ability to understand complex relationships between video frames improves. Conversely, as model size decreases, the edit distance tends to increase, suggesting a greater deviation in predictions over broader contexts.

Variant	Accuracy↑			Cos. Sim.↑	ED↓
	Full Acc.	Verb Acc.	Noun Acc.		
VISTA-2B	0.127	0.153	0.074	0.941	0.437
VISTA-7B	0.144	0.106	0.098	0.942	0.425
VISTA-72B	**0.482**	**0.587**	**0.610**	**0.967**	**0.262**

Table 5.13: Metrics with respect to Different Variants of VLM in Kitchen Scenario with 3s of prediction horizon.

5.7 Limitations

5.7.1 Limitations for In Context Learning

In context learning has shown great potential for human behavior prediction task, but it comes with a drawback. As previously discussed, the in context learning images for GPT-4o & GPT-4o-mini does not scale beyond 50. Hence in case of diverse environments it is not possible to include sufficient input-output pairs from all the environments to make accurate predictions. Furthermore there are instants where the input-output pairs are mainly used to derive the specific output format and the information & mapping the logic to scene graph is not fully utilized, hence which results in low metrics. In Context learning has significantly more difficulties when considering visual inputs, as the each image is converted into large number of visual tokens, which quickly fills up the context length and results in significantly longer inference time. As the context window approaches its limit, this increases the risk of model hallucination, where the generated outputs may diverge from grounded, contextually relevant information [29].

5.7.2 Limitations with VLM

Although Supervised fine tuning method provides us with state-of-the-art results. There are still some limitations with these approach which needs to be addressed. The performance of VLMs is highly sensitive to how prompts are constructed. Small changes in prompt structure or content can lead to significant variability in predictions. Fine-tuning VLMs for specific human behavior prediction tasks is computationally expensive and resource-intensive. For the fine tuning & experiments, NVIDIA H100 GPUs were used, hence the LoRA fine tuning was completed in 2 hours. However this training time depends on the type of GPU available. The improvement of SFT is also limited by the size of the training dataset, which can result the model not being able to generalize well in diverse scenarios where enough training data is not available.

While VLMs can simulate or predict some aspects of human decision-making, they often fail to capture deeper cognitive mechanisms, such as theory of mind or human cognitive biases or take any other modality into account such as speech [49]. There is a risk that VLMs base their predictions on superficial features or provided data rather than deep understanding, making them vulnerable to manipulation and reducing trust in their outputs

5.7.3 Limitations when Providing Spatial Data

The experiments indicate that achieving accurate predictions of action labels strongly depends on incorporating scene graphs as supplementary contextual knowledge. Similarly, Palm [34] report their improved performance while providing video frames and captions together by utilizing another Vision Language Model. Liu et. al [43] also report their best results when providing video frames and textual information together. Constructing such two-dimensional (2D) representations from 3D scene graphs remains an open research direction. In this work, inspiration is taken from Delta Framework [44] and assume the availability of 2D scene graphs.

6 Conclusion

This thesis presents study on Multiple Human behavior prediction. Current state of the art work focuses on egocentric action prediction and major work that studies third person action anticipation considers single human in a scene. This thesis aims to build a framework for multiple human behavior prediction from a third person view, by using video data and scene graphs to forecast actions in various indoor environments. Due to the limited availability of suitable datasets, this work also presents a synthetic dataset that comprises of multiple-human actions in an indoor household scenario.

This thesis presents VISTA which stands for Vision and Scene aware Temporal Action anticipation, which is an open-source vision-language model based framework that utilizes video data and scene graph to forecast the actions of multiple humans in a collaborative environment. The model is fine tuned using Supervised Fine Tuning (SFT) method and aligned using Direct Preference Optimization (DPO) method. By training the model on these methods, VISTA achieves 13% of improvement as compared to current baselines. The thesis conducts study on synthetic data and real world recorded data upto 3 humans in a shared environment. The research also outlines several ablation studies to evaluate different components of the system. This research also evaluates different sizes of model including 2B, 7B and 72B and finally conclude that the 72B model surpasses the state-of-the-art model such as GPT-4o and GPT-4o-mini. This work also examines inference times over several hardware setups and compare VISTA with models using In- Context Learning (ICL) [43]. The findings draw attention to important limits of ICL-based methods, especially in cases when using models like GPT-4o, which show great reliance on high-quality textual inputs.

At the end, the limitations of the architecture are also highlighted. Despite achieving significant results as compared to baselines, the architecture presented in this thesis is still dependent on scene graphs, which are not directly available from the environment and often require additional preprocessing steps.

Future Work

Although this thesis achieved significant results with VISTA, there are still improvements that can be made for efficient real world deployment.

The research presented in this thesis utilizes 72B models, however efficiently storing these models and getting continuous outputs from this huge models is not ideal in case when there limited hardware availability. Future work can explore techniques such Knowledge Distillation and quantization methods or using hybrid architectures for efficient deployment while keeping consistent results.

Although this work predicts the multiple-human actions upto 5 seconds. Future work can focus on anticipating on longer horizons. There are some works which focuses on predicting actions upto 60 seconds [19] but the architectures are not suitable for real world deployment. To achieve this, two different of loss functions could be introduced which are focused on short-term prediction loss and long-term prediction loss. By this the model can be trained by keeping more weight on the long-term horizon loss.

Another work which would be interesting will be to explore multiple modalities. In this research, future actions are predicted purely based on history video data and scene graphs. However human actions are not only dependent on past actions. Modalities such as audio signals, verbal communication between two humans and even environment sensor data can be considered as an additional contextual cues to enhance prediction capabilities.

Bibliography

[1] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, K. Simonyan. *Flamingo: a Visual Language Model for Few-Shot Learning*. 2022. arXiv: 2204.14198 [cs.CV]. URL: https://arxiv.org/abs/2204.14198 (cit. on p. 20).

[2] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, É. Goffinet, D. Hesslow, J. Launay, Q. Malartic, et al. "The falcon series of open language models". In: *arXiv preprint arXiv:2311.16867* (2023) (cit. on p. 21).

[3] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, S. Savarese. *3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera*. 2019. arXiv: 1910.02527 [cs.CV]. URL: https://arxiv.org/abs/1910.02527 (cit. on pp. 13, 40).

[4] L. Aymerich-Franch, I. Ferrer. *Socially assistive robots' deployment in healthcare settings: a global perspective*. 2021. arXiv: 2110.07404 [cs.RO]. URL: https://arxiv.org/abs/2110.07404 (cit. on p. 11).

[5] M. Azarmi, M. Rezaei, H. Wang, S. Glaser. *PIP-Net: Pedestrian Intention Prediction in the Wild*. 2024. arXiv: 2402.12810 [cs.CV]. URL: https://arxiv.org/abs/2402.12810 (cit. on pp. 11, 30, 41).

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901 (cit. on pp. 13, 15, 20, 33).

[7] L. Bruckschen, K. Bungert, N. Dengler, M. Bennewitz. "Human-Aware Robot Navigation by Long-Term Movement Prediction". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11032–11037. DOI: 10.1109/IROS45743.2020.9340776 (cit. on p. 11).

[8] D. Caffagni, F. Cocchi, L. Barsellotti, N. Moratelli, S. Sarto, L. Baraldi, L. Baraldi, M. Cornia, R. Cucchiara. *The Revolution of Multimodal Large Language Models: A Survey*. 2024. arXiv: 2402.12451 [cs.CV]. URL: https://arxiv.org/abs/2402.12451 (cit. on pp. 17, 18).

[9] Z. Cao, H. Gao, K. Mangalam, Q.-Z. Cai, M. Vo, J. Malik. *Long-term Human Motion Prediction with Scene Context*. 2020. arXiv: 2007.03672 [cs.CV]. URL: https://arxiv.org/abs/2007.03672 (cit. on pp. 12, 13).

[10] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, D. Amodei. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 24).

[11] Z. Dai, H. Liu, Q. V. Le, M. Tan. *CoAtNet: Marrying Convolution and Attention for All Data Sizes*. 2021. arXiv: 2106.04803 [cs.CV]. URL: https://arxiv.org/abs/2106.04803 (cit. on p. 17).

[12] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, M. Wray. *Scaling Egocentric Vision: The EPIC-KITCHENS Dataset*. 2018. arXiv: 1804.02748 [cs.CV]. URL: https://arxiv.org/abs/1804.02748 (cit. on p. 29).

[13] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, M. Wray. *The EPIC-KITCHENS Dataset: Collection, Challenges and Baselines*. 2020. arXiv: 2005.00343 [cs.CV]. URL: https://arxiv.org/abs/2005.00343 (cit. on pp. 12, 36).

[14] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805 (cit. on pp. 18, 19, 21, 24, 25).

[15] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, Z. Sui. *A Survey on In-context Learning*. 2024. arXiv: 2301.00234 [cs.CL]. URL: https://arxiv.org/abs/2301.00234 (cit. on p. 20).

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929 (cit. on pp. 13, 15–17).

[17] Y. A. Farha, A. Richard, J. Gall. *When will you do what? - Anticipating Temporal Occurrences of Activities*. 2018. arXiv: 1804.00892 [cs.CV]. URL: https://arxiv.org/abs/1804.00892 (cit. on pp. 12, 33, 34).

[18] R. Girdhar, K. Grauman. *Anticipative Video Transformer*. 2021. arXiv: 2106.02036 [cs.CV]. URL: https://arxiv.org/abs/2106.02036 (cit. on pp. 12, 31).

[19] N. Gorlo, L. Schmid, L. Carlone. "Long-Term Human Trajectory Prediction Using 3D Dynamic Scene Graphs". In: *IEEE Robotics and Automation Letters* 9.12 (2024), pp. 10978–10985. DOI: 10.1109/LRA.2024.3482169 (cit. on pp. 11, 12, 40, 67).

[20] M. A. Graule, V. Isler. *GG-LLM: Geometrically Grounding Large Language Models for Zero-shot Human Activity Forecasting in Human-Aware Task Planning*. 2023. arXiv: 2310.20034 [cs.RO]. URL: https://arxiv.org/abs/2310.20034 (cit. on pp. 34, 35).

[21] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. "Ego4d: Around the world in 3,000 hours of egocentric video". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 18995–19012 (cit. on pp. 12, 29, 36).

[22] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, J. Malik. *AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions*. 2018. arXiv: 1705.08421 [cs.CV]. URL: https://arxiv.org/abs/1705.08421 (cit. on p. 36).

[23] F. Gu, M.-H. Chung, M. Chignell, S. Valaee, B. Zhou, X. Liu. "A Survey on Deep Learning for Human Activity Recognition". In: *ACM Computing Surveys* 54 (Aug. 2021). DOI: 10.1145/3472290 (cit. on p. 11).

[24] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, L. Paull. *ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning*. 2023. arXiv: 2309.16650 [cs.RO]. URL: https://arxiv.org/abs/2309.16650 (cit. on p. 40).

[25] J. He, P. Li, G. Liu, G. He, Z. Chen, S. Zhong. *PeFoMed: Parameter Efficient Fine-tuning of Multimodal Large Language Models for Medical Imaging*. 2025. arXiv: 2401.02797 [cs.CL]. URL: https://arxiv.org/abs/2401.02797 (cit. on pp. 21, 22).

[26] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, L. Sifre. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. URL: https://arxiv.org/abs/2203.15556 (cit. on p. 15).

[27] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly. *Parameter-Efficient Transfer Learning for NLP*. 2019. arXiv: 1902.00751 [cs.LG]. URL: https://arxiv.org/abs/1902.00751 (cit. on pp. 21, 22).

[28] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685 (cit. on pp. 22–24, 43, 44, 48, 63).

[29] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu. “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions”. In: *ACM Transactions on Information Systems* 43.2 (Jan. 2025), pp. 1–55. ISSN: 1558-2868. DOI: 10.1145/3703155. URL: http://dx.doi.org/10.1145/3703155 (cit. on p. 64).

[30] Huggingface. *Open LLM leaderboard - A hugging face space by open-LLM-leaderboard*. URL: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard (cit. on p. 47).

[31] N. Hughes, Y. Chang, L. Carlone. *Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization*. 2022. arXiv: 2201.13360 [cs.RO]. URL: https://arxiv.org/abs/2201.13360 (cit. on p. 40).

[32] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361 (cit. on p. 15).

[33] J. Kasai, K. Sakaguchi, Y. Takahashi, R. L. Bras, A. Asai, X. Yu, D. Radev, N. A. Smith, Y. Choi, K. Inui. *RealTime QA: What’s the Answer Right Now?* 2024. arXiv: 2207.13332 [cs.CL]. URL: https://arxiv.org/abs/2207.13332 (cit. on p. 19).

[34] S. Kim, D. Huang, Y. Xian, O. Hilliges, L. V. Gool, X. Wang. *PALM: Predicting Actions through Language Models*. 2024. arXiv: 2311.17944 [cs.CV]. URL: https://arxiv.org/abs/2311.17944 (cit. on pp. 12, 29, 32, 33, 42, 65).

[35] D. P. Kingma, J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980 (cit. on pp. 47, 49).

[36] Y. Kong, Y. Fu. “Human action recognition and prediction: A survey”. In: *International Journal of Computer Vision* 130.5 (2022), pp. 1366–1401 (cit. on p. 29).

[37] H. Kuehne, A. B. Arslan, T. Serre. “The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities”. In: *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*. 2014 (cit. on pp. 36, 37).

[38] C. Li, H. Farkhoor, R. Liu, J. Yosinski. *Measuring the Intrinsic Dimension of Objective Landscapes*. 2018. arXiv: 1804.08838 [cs.LG]. URL: https://arxiv.org/abs/1804.08838 (cit. on p. 23).

[39] J. Li, D. Li, S. Savarese, S. Hoi. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. 2023. arXiv: 2301.12597 [cs.CV]. URL: https://arxiv.org/abs/2301.12597 (cit. on p. 17).

[40] Y. Li, M. Liu, J. M. Rehg. *In the Eye of the Beholder: Gaze and Actions in First Person Video*. 2020. arXiv: 2006.00626 [cs.CV]. URL: https://arxiv.org/abs/2006.00626 (cit. on pp. 12, 36).

[41] Y. Lin, X. Ma, X. Chu, Y. Jin, Z. Yang, Y. Wang, H. Mei. *LoRA Dropout as a Sparsity Regularizer for Overfitting Control*. 2024. arXiv: 2404.09610 [cs.LG]. URL: https://arxiv.org/abs/2404.09610 (cit. on p. 63).

[42] H. Liu, C. Li, Q. Wu, Y. J. Lee. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV]. URL: https://arxiv.org/abs/2304.08485 (cit. on p. 18).

[43] Y. Liu, L. Lerch, L. Palmieri, A. Rudenko, S. Koch, T. Ropinski, M. Aiello. *Context-Aware Human Behavior Prediction Using Multimodal Large Language Models: Challenges and Insights*. 2025. arXiv: 2504.00839 [cs.RO]. URL: https://arxiv.org/abs/2504.00839 (cit. on pp. 34, 35, 50, 51, 56, 58, 65, 67).

[44] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, M. Aiello. *DELTA: Decomposed Efficient Long-Term Robot Task Planning using Large Language Models*. 2025. arXiv: 2404.03275 [cs.RO]. URL: https://arxiv.org/abs/2404.03275 (cit. on pp. 41, 65).

[45] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, S. Xie. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545 (cit. on p. 17).

[46] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, J. Gao. “MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts”. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=KUNzEQMWU7 (cit. on p. 18).

[47] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, K. Yadav, Q. Li, B. Newman, M. Sharma, V. Berges, S. Zhang, P. Agrawal, Y. Bisk, D. Batra, M. Kalakrishnan, F. Meier, C. Paxton, S. Sax, A. Rajeswaran. “OpenEQA: Embodied Question Answering in the Era of Foundation Models”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024 (cit. on p. 19).

[48] R. Möller, A. Furnari, S. Battiato, A. Härmä, G. M. Farinella. *A Survey on Human-aware Robot Navigation*. 2021. arXiv: 2106.11650 [cs.RO]. URL: https://arxiv.org/abs/2106.11650 (cit. on pp. 11, 12).

[49] T. N. Nguyen, K. Jamale, C. Gonzalez. *Predicting and Understanding Human Action Decisions: Insights from Large Language Models and Cognitive Instance-Based Learning*. 2024. arXiv: 2407.09281 [cs.AI]. URL: https://arxiv.org/abs/2407.09281 (cit. on pp. 11, 12, 65).

[50] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774 (cit. on pp. 18, 25, 47, 49, 58).

[51] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155 (cit. on p. 21).

[52] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744 (cit. on p. 24).

[53] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, A. Torralba. "VirtualHome: Simulating Household Activities Via Programs". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8494–8502. DOI: 10.1109/CVPR.2018.00886 (cit. on pp. 26–28, 40, 77).

[54] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020 (cit. on pp. 16–18, 31).

[55] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. "Language Models are Unsupervised Multitask Learners". In: 2019. URL: https://api.semanticscholar.org/CorpusID:160025533 (cit. on p. 17).

[56] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, C. Finn. "Direct preference optimization: Your language model is secretly a reward model". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 53728–53741 (cit. on pp. 24–26, 44).

[57] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, N. Suenderhauf. *SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning*. 2023. arXiv: 2307.06135 [cs.RO]. URL: https://arxiv.org/abs/2307.06135 (cit. on p. 40).

[58] A. Rasouli, I. Kotseruba, J. K. Tsotsos. *Pedestrian Action Anticipation using Contextual Feature Fusion in Stacked RNNs*. 2020. arXiv: 2005.06582 [cs.CV]. URL: https://arxiv.org/abs/2005.06582 (cit. on pp. 11, 12).

[59] A. Rosinol, A. Gupta, M. Abate, J. Shi, L. Carlone. *3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans*. 2020. arXiv: 2002.06289 [cs.RO]. URL: https://arxiv.org/abs/2002.06289 (cit. on p. 40).

[60] D. Roy, B. Fernando. "Action anticipation using latent goal learning". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 808–816. DOI: 10.1109/WACV51458.2022.00088 (cit. on p. 30).

[61] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, K. O. Arras. "Human motion trajectory prediction: a survey". In: *The International Journal of Robotics Research* 39.8 (June 2020), pp. 895–935. ISSN: 1741-3176. DOI: 10.1177/0278364920917446. URL: http://dx.doi.org/10.1177/0278364920917446 (cit. on pp. 11, 30).

[62] S. Stein, S. Mckenna. "Combining embedded accelerometers with computer vision for recognizing food preparation activities". In: Sept. 2013, pp. 729–738. DOI: 10.1145/2493432.2493482 (cit. on p. 36).

[63] C. Sun, A. Shrivastava, C. Vondrick, R. Sukthankar, K. Murphy, C. Schmid. *Relational Action Forecasting*. 2019. arXiv: 1904.04231 [cs.CV]. URL: https://arxiv.org/abs/1904.04231 (cit. on p. 12).

[64] N. Tax. "Human Activity Prediction in Smart Home Environments with LSTM Neural Networks". In: *2018 14th International Conference on Intelligent Environments (IE)*. 2018, pp. 40–47. DOI: 10.1109/IE.2018.00014 (cit. on p. 12).

[65] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. "Gemini: a family of highly capable multimodal models". In: *arXiv preprint arXiv:2312.11805* (2023) (cit. on pp. 21, 25).

[66] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. "Gemma: Open models based on gemini research and technology". In: *arXiv preprint arXiv:2403.08295* (2024) (cit. on p. 21).

[67] S. Tong, E. Brown, P. Wu, S. Woo, M. Middepogu, S. C. Akula, J. Yang, S. Yang, A. Iyer, X. Pan, Z. Wang, R. Fergus, Y. LeCun, S. Xie. *Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs*. 2024. arXiv: 2406.16860 [cs.CV]. URL: https://arxiv.org/abs/2406.16860 (cit. on pp. 18, 19).

[68] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971 (cit. on pp. 15, 17, 25, 32, 34, 52).

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on pp. 13, 15, 16, 49).

[70] J. Wald, H. Dhamo, N. Navab, F. Tombari. *Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions*. 2020. arXiv: 2004.03967 [cs.CV]. URL: https://arxiv.org/abs/2004.03967 (cit. on p. 40).

[71] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, Y. Fan, K. Dang, M. Du, X. Ren, R. Men, D. Liu, C. Zhou, J. Zhou, J. Lin. *Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution*. 2024. arXiv: 2409.12191 [cs.CV]. URL: https://arxiv.org/abs/2409.12191 (cit. on pp. 16, 17, 40, 47, 48).

[72] Z. Wang, B. Cheng, L. Zhao, D. Xu, Y. Tang, L. Sheng. *VL-SAT: Visual-Linguistic Semantics Assisted Training for 3D Semantic Scene Graph Prediction in Point Cloud*. 2023. arXiv: 2303.14408 [cs.CV]. URL: https://arxiv.org/abs/2303.14408 (cit. on p. 40).

[73] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le. *Finetuned Language Models Are Zero-Shot Learners*. 2022. arXiv: 2109.01652 [cs.CL]. URL: https://arxiv.org/abs/2109.01652 (cit. on p. 21).

[74] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: https://arxiv.org/abs/1910.03771 (cit. on pp. 47, 49).

[75] S.-C. Wu, J. Wald, K. Tateno, N. Navab, F. Tombari. *SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences*. 2021. arXiv: 2103.14898 [cs.CV]. URL: https://arxiv.org/abs/2103.14898 (cit. on p. 40).

[76] Y. Xian, C. H. Lampert, B. Schiele, Z. Akata. *Zero-Shot Learning – A Comprehensive Evaluation of the Good, the Bad and the Ugly*. 2020. arXiv: 1707.00600 [cs.CV]. URL: https://arxiv.org/abs/1707.00600 (cit. on pp. 19, 20).

[77] Z. Yang, Z. Gan, J. Wang, X. Hu, Y. Lu, Z. Liu, L. Wang. *An Empirical Study of GPT-3 for Few-Shot Knowledge-Based VQA*. 2022. arXiv: 2109.05014 [cs.CV]. URL: https://arxiv.org/abs/2109.05014 (cit. on p. 21).

[78] X. Yue, Y. Ni, K. Zhang, T. Zheng, R. Liu, G. Zhang, S. Stevens, D. Jiang, W. Ren, Y. Sun, C. Wei, B. Yu, R. Yuan, R. Sun, M. Yin, B. Zheng, Z. Yang, Y. Liu, W. Huang, H. Sun, Y. Su, W. Chen. "MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI". In: *Proceedings of CVPR*. 2024 (cit. on p. 18).

[79] C. Zhang, A. Delitzas, F. Wang, R. Zhang, X. Ji, M. Pollefeys, F. Engelmann. *Open-Vocabulary Functional 3D Scene Graphs for Real-World Indoor Spaces*. 2025. arXiv: 2503.19199 [cs.CV]. URL: https://arxiv.org/abs/2503.19199 (cit. on p. 13).

[80] Q. Zhao, S. Wang, C. Zhang, C. Fu, M. Q. Do, N. Agarwal, K. Lee, C. Sun. "AntGPT: Can Large Language Models Help Long-term Action Anticipation from Videos?" In: *ICLR* (2024) (cit. on pp. 12, 13, 29, 31, 32, 42, 50, 52, 56, 58).

[81] T. Z. Zhao, E. Wallace, S. Feng, D. Klein, S. Singh. *Calibrate Before Use: Improving Few-Shot Performance of Language Models*. 2021. arXiv: 2102.09690 [cs.CL]. URL: https://arxiv.org/abs/2102.09690 (cit. on p. 21).

[82] Y. Zhao, T. Kwon, P. Streli, M. Pollefeys, C. Holz. "EgoPressure: A Dataset for Hand Pressure and Pose Estimation in Egocentric Vision". In: *arXiv preprint arXiv:2409.02224* (2024) (cit. on p. 36).

[83] Z. Zhong, M. Martin, M. Voit, J. Gall, J. Beyerer. "A survey on deep learning techniques for action anticipation". In: *arXiv preprint arXiv:2309.17257* (2023) (cit. on p. 29).

A Appendix

A.1 Virtual Home Simulator Programs

This section describes the programs that are used to generate the synthetic videos using Virtual Home Simulator [53]. The format of virtual home programs is described in section 2.4. The scripts are used across 6 different environments. Each environment has unique spatial configurations and object layouts. In this work, videos are simulated from three primary domestic scenarios: kitchen, living room, and bedroom. The following programs are used to generate videos across various environments.

A.1.1 Scripts for Kitchen Scenario

Script 1

This is an example of two humans in a kitchen scene, one human heats up food using a microwave and another human drinks a milkshake.

```
-"<char0> [walk] <salmon> ({salmon_id}) | <char1> [walk] <kitchentable> ({kitchentable_id})",
-"<char0> [grab] <salmon> ({salmon_id}) | <char1> [grab] <milkshake> ({milkshake_id})",
-"<char0> [open] <microwave> ({microwave_id}) | <char1> [drink] <milkshake> ({milkshake_id})",
-"<char0> [putin] <salmon> ({salmon_id}) <microwave> ({microwave_id}) | <char1> [walk] <sink>
({sink_id})",
-"<char0> [close] <microwave> ({microwave_id}) | <char1> [putback] <milkshake> ({milkshake_id
}) <sink> ({sink_id})",
-"<char0> [open] <microwave> ({microwave_id}) | <char1> [switchon] <faucet> ({faucet_id})",
-"<char0> [grab] <salmon> ({salmon_id}) | <char1> [switchoff] <faucet> ({faucet_id})",
-"<char0> [close] <microwave> ({microwave_id}) | <char1> [grab] <milkshake> ({milkshake_id})",
-"<char0> [walk] <kitchentable> ({kitchentable_id}) | <char1> [walk] <kitchentable> ({
kitchentable_id})",
-"<char0> [put] <salmon> ({salmon_id}) <kitchentable> ({kitchentable_id}) | <char1> [put] <
milkshake> ({milkshake_id}) <kitchentable> ({kitchentable_id})",
-"<char0> [grab] <glass> ({glass_id})",
-"<char0> [walk] <sink> ({sink_id})",
-"<char0> [putback] <glass> ({glass_id}) <sink> ({sink_id})",
-"<char0> [switchon] <faucet> ({faucet_id})",
-"<char0> [switchoff] <faucet> ({faucet_id})",
-"<char0> [grab] <glass> ({glass_id})",
-"<char0> [walk] <kitchentable> ({kitchentable_id})",
-"<char0> [put] <glass> ({glass_id}) <kitchentable> ({kitchentable_id})",
```

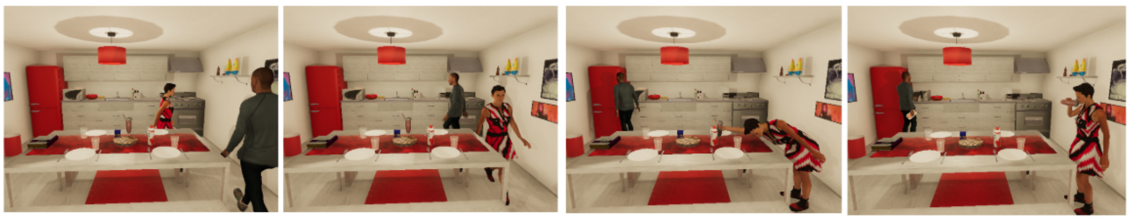


Figure A.1: Sample video frames for Kitchen Script 1

Script 2

This is another example of two humans in kitchen scene, where one human is performing cooking action and another human is interacting with objects on the kitchen table.

```
    -"<char0> [walk] <fryingpan> ({fryingpan_id}) | <char1> [walk] <cutleryknife> ({
curleryknife_id})",
    -"<char0> [grab] <fryingpan> ({fryingpan_id}) | <char1> [grab] <cutleryknife> ({
curleryknife_id})",
    -"<char0> [put] <fryingpan> ({fryingpan_id}) <stove> ({stove_id}) | <char1> [walk] <sink>
({sink_id})",
    -"<char0> [walk] <salmon> ({salmon_id}) | <char1> [put] <cutleryknife> ({curleryknife_id})
 <sink> ({sink_id})",
    -"<char0> [grab] <salmon> ({salmon_id}) | <char1> [walk] <cutleryfork> ({curleryfork_id})"
,
    -"<char0> [walk] <fryingpan> ({fryingpan_id}) | <char1> [grab] <cutleryfork> ({
curleryfork_id})",
    -"<char0> [put] <salmon> ({salmon_id}) <fryingpan> ({fryingpan_id}) | <char1> [walk] <sink
> ({sink_id})",
    -"<char0> [switchon] <stove> ({stove_id}) | <char1> [put] <cutleryfork> ({curleryfork_id})
 <sink> ({sink_id})",
    -"<char0> [switchoff] <stove> ({stove_id}) | <char1> [walk] <plate> ({plate_id})",
    -"<char1> [grab] <plate> ({plate_id})",
    -"<char1> [walk] <sink> ({sink_id})",
    -"<char1> [put] <plate> ({plate_id}) <sink> ({sink_id})",
    -"<char1> [switchon] <faucet> ({faucet_id})",
    -"<char1> [grab] <dishwashingliquid> ({dishwashingliquid_id})",
    -"<char1> [put] <dishwashingliquid> ({dishwashingliquid_id}) <sink> ({sink_id})",
    -"<char1> [switchoff] <faucet> ({faucet_id})",
    -"<char1> [open] <kitchencabinet> ({kitchencabinet_id})",
    -"<char1> [grab] <plate> ({plate_id})",
    -"<char1> [put] <plate> ({plate_id}) <kitchencabinet> ({kitchencabinet_id})",
    -"<char1> [grab] <cutleryknife> ({curleryknife_id})",
    -"<char1> [put] <cutleryknife> ({curleryknife_id}) <kitchencabinet> ({kitchencabinet_id})"
,
    -"<char1> [grab] <cutleryfork> ({curleryfork_id})",
    -"<char1> [put] <cutleryfork> ({curleryfork_id}) <kitchencabinet> ({kitchencabinet_id})",
```

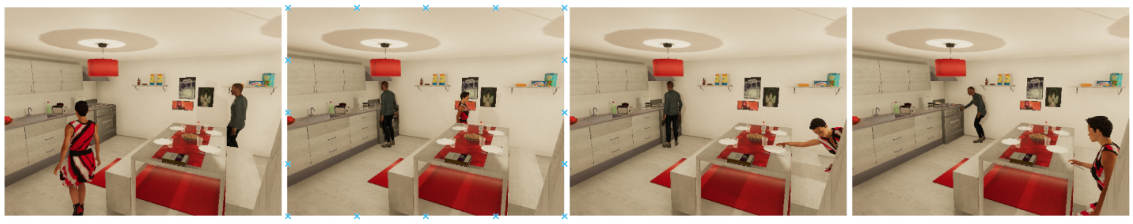


Figure A.2: Sample video frames for Kitchen Script 2

Script 3

This is an example of three humans in Kitchen Scene, where one human is washing the dishes using a dishwasher, one human is making coffee and another human is putting items inside the refrigerator.

```
- '<char0> [walk] <kitchentable> ({kitchentable_id})',
- '<char0> [grab] <plate> ({plates}) | <char1> [grab] <mug> ({mug})',
- '<char0> [walk] <dishwasher> ({dishwasher}) | <char1> [walk] <coffeemaker> ({coffeemaker}) |
 <char2> [walk] <kitchentable> ({kitchentable_id})',
- '<char0> [open] <dishwasher> ({dishwasher}) | <char1> [put] <mug> ({mug}) <coffeemaker> ({
coffeemaker}) | <char2> [grab] <cupcake> ({cupcake1})',
- '<char0> [putin] <plate> ({plates}) <dishwasher> ({dishwasher}) | <char1> [lookat] <
coffeemaker> ({coffeemaker}) | <char2> [walk] <fridge> ({fridge})',
- '<char0> [walk] <stove> ({stove}) | <char1> [lookat] <coffeemaker> ({coffeemaker}) | <char2>
 [open] <fridge> ({fridge})',
- '<char0> [grab] <cookingpot> ({cookingpot}) | <char1> [lookat] <coffeemaker> ({coffeemaker})
 | <char2> [putin] <cupcake> ({cupcake1}) <fridge> ({fridge})',
- '<char0> [putin] <cookingpot> ({cookingpot}) <dishwasher> ({dishwasher}) | <char2> [walk] <
kitchentable> ({kitchentable_id})',
- '<char0> [close] <dishwasher> ({dishwasher}) | <char2> [grab] <cupcake> ({cupcake2})',
- '<char0> [switchon] <dishwasher> ({dishwasher}) | <char2> [putin] <cupcake> ({cupcake2}) <
fridge> ({fridge})',
- '<char0> [switchoff] <dishwasher> ({dishwasher}) | <char1> [grab] <mug> ({mug}) | <char2> [
close] <fridge> ({fridge})',
- '<char0> [open] <dishwasher> ({dishwasher}) | <char1> [walk] <kitchentable> ({
kitchentable_id})',
- '<char0> [grab] <plate> ({plates}) | <char1> [drink] <mug> ({mug})',
- '<char0> [open] <kitchencabinet> ({kitchencabinets[0]}) | <char1> [walk] <sink> ({sink})',
- '<char0> [putin] <plate> ({plates}) <kitchencabinet> ({kitchencabinets[0]}) | <char1> [put]
<mug> ({mug}) <sink> ({sink})',
- '<char0> [grab] <cookingpot> ({cookingpot}) | <char1> [switchon] <faucet> ({faucet})',
- '<char0> [close] <dishwasher> ({dishwasher})',
- '<char0> [putin] <cookingpot> ({cookingpot}) <kitchencabinet> ({kitchencabinets[0]})',
- '<char0> [close] <kitchencabinet> ({kitchencabinets[0]})',
```

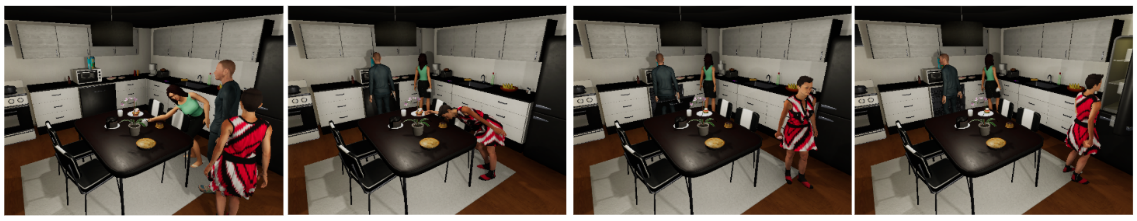


Figure A.3: Sample video frames for Kitchen Script 3

A.1.2 Scripts for Livingroom Scenario

Script 1

This is an example of Livingroom scenario where one person is watching television and another person is working on a computer.

```
    -'<char0> [walk] <tv> ({tv}) | <char1> [walk] <computer> ({computer})',
    -'<char0> [switchon] <tv> ({tv}) | <char1> [switchon] <computer> ({computer})',
    -'<char0> [walk] <coffeetable> ({coffeetable})',
    -'<char0> [grab] <milkshake> ({milkshake}) | <char1> [walk] <chair> ({chairs[-1]})',
    -'<char0> [walk] <sofa> ({sofa}) | <char1> [sit] <chair> ({chairs[-1]})',
    -'<char0> [sit] <sofa> ({sofa})',
    -'<char0> [drink] <milkshake> ({milkshake})',
```

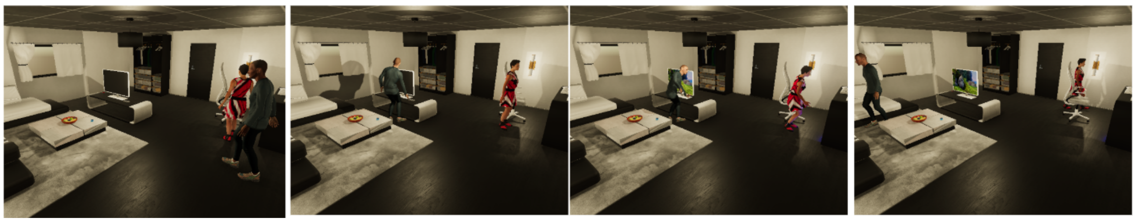


Figure A.4: Sample video frames for Livingroom Script 1

Script 2

This is an example script of three humans in Livingroom scenario. One human is arranging folders in bookshelf, one human is working at computer and one human is watching television.

```
- "<char0> [walk] <coffeetable> ({livingroomcoffeetable_id}) | <char1> [walk] <coffeetable> ({
livingroomcoffeetable_id}) | <char2> [walk] <tv> ({tv})",
- "<char0> [grab] <folder> ({folder_id}) | <char1> [grab] <mug> ({mug_id})",
- "<char0> [walk] <bookshelf> ({bookshelf_id}) | <char1> [walk] <computer> ({computer_id}) | <
char2> [switchon] <tv> ({tv})",
- "<char0> [putin] <folder> ({folder_id}) <bookshelf> ({bookshelf_id}) | <char1> [switchon] <
computer> ({computer_id}) | <char2> [walk] <sofa> ({sofa_id})",
- "<char0> [walk] <coffeetable> ({livingroomcoffeetable_id}) | <char1> [walk] <chair> ({
chair_id}) | <char2> [sit] <sofa> ({sofa_id})",
- "<char0> [grab] <book> ({book_id}) | <char1> [sit] <chair> ({chair_id})",
- "<char0> [walk] <sofa> ({sofa_id})",
- "<char0> [sit] <sofa> ({sofa_id})",
```

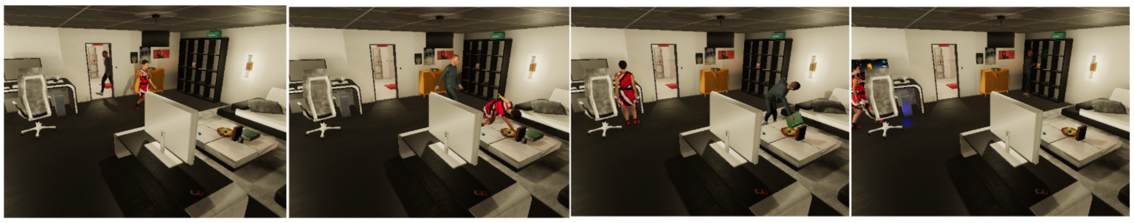


Figure A.5: Sample video frames for Livingroom Script 2

A.1.3 Scripts for Bedroom Scenario

Script 1

This is an example of two humans in bedroom scene, where one human is arranging clothes in a closet and another human reads a book.

```
- "<char0> [walk] <bed> ({bed_id})",
- "<char0> [grab] <clothespile> ({clothpile_id}) | <char1> [walk] <tablelamp> ({tablelamp1_id
})",
- "<char0> [walk] <closet> ({closet_id}) |  <char1> [switchon] <tablelamp> ({tablelamp1_id})",
- "<char0> [putin] <clothespile> ({clothpile_id}) <closet> ({closet_id}) | <char1> [walk] <
tablelamp> ({tablelamp2_id})",
- "<char0> [walk] <bed> ({bed_id}) | <char1> [switchon] <tablelamp> ({tablelamp2_id})",
- "<char0> [grab] <clothespants> ({clothespant_id}) | <char1> [grab] <book> ({book_id})",
- "<char0> [walk] <closet> ({closet_id}) | <char1> [walk] <chair> ({chair_id})",
- "<char0> [putin] <clothespants> ({clothespant_id}) <closet> ({closet_id}) | <char1> [sit] <
chair> ({chair_id})",
- "<char0> [walk] <bed> ({bed_id})",
- "<char0> [grab] <clothesshirt> ({clothesshirt_id})",
- "<char0> [walk] <closet> ({closet_id})",
- "<char0> [putin] <clothesshirt> ({clothesshirt_id}) <closet> ({closet_id})"
```

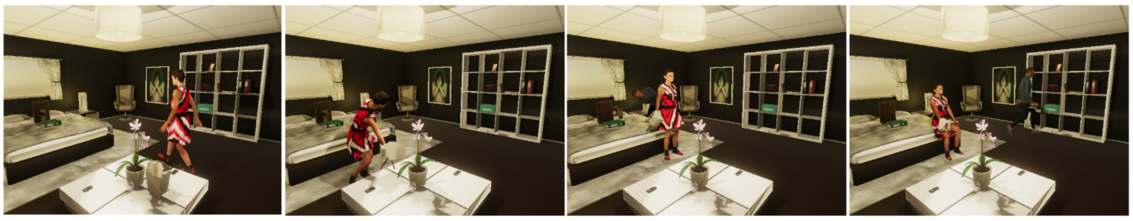


Figure A.6: Sample video frames for Bedroom Script 1

Script 2

Another example script of two humans in bedroom scenario, where one human is arranging stationary in bookshelf and another human is eating food.

```
- "<char0> [walk] <bed> ({bed_id}) | <char1> [walk] <coffeetable> ({bedroomcoffeetable_id})",
- "<char0> [grab] <book> ({book_id}) | <char1> [grab] <chinesefood> ({chinesebox})",
- "<char0> [walk] <bookshelf> (bookshelf_id) | <char1> [walk] <bed> ({bed_id})",
- "<char0> [putin] <book> ({book_id}) <bookshelf> (bookshelf_id) | <char1> [sit] <bed> ({
bed_id})",
- "<char0> [walk] <bed> ({bed_id}) | <char1> [sit] <bed> ({bed_id})",
- "<char0> [grab] <magazine> ({magazine_id})",
- "<char0> [walk] <bookshelf> (bookshelf_id)",
- "<char0> [putin] <magazine> ({magazine_id}) <bookshelf> (bookshelf_id)",
- "<char0> [walk] <bed> ({bed_id})",
- "<char0> [grab] <folder> ({folder_id})",
- "<char0> [walk] <bookshelf> (bookshelf_id)",
- "<char0> [putin] <folder> ({folder_id}) <bookshelf> (bookshelf_id)",
- "<char0> [walk] <bed> ({bed_id})",
- "<char0> [grab] <journal> ({journal_id})",
- "<char0> [walk] <bookshelf> (bookshelf_id)",
- "<char0> [putin] <journal> ({journal_id}) <bookshelf> (bookshelf_id)",
```

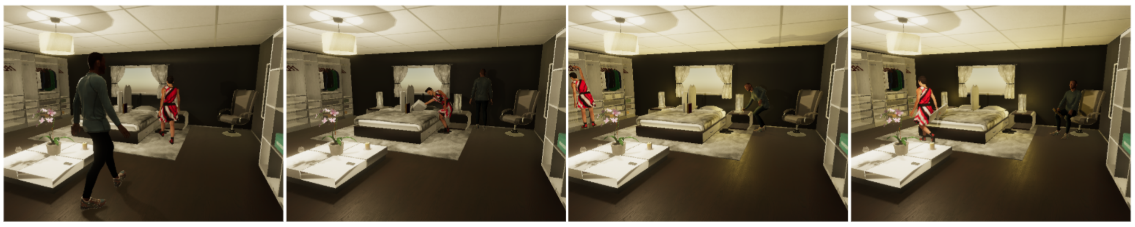


Figure A.7: Sample video frames for Bedroom Script 2

Declaration