JARVIS-VLA: Post-Training Large-Scale Vision Language Models to Play Visual Games with Keyboards and Mouse

Muyao Li[1]*   **Zihao Wang**[1]*   **Kaichen He**[1]   **Xiaojian Ma**[2]   **Yitao Liang**[1]†

[1]Peking University   [2]BIGAI

Abstract

Recently, action-based decision-making in open-world environments has gained significant attention. Visual Language Action (VLA) models, pretrained on large-scale web datasets, have shown promise in decision-making tasks. However, previous work has primarily focused on action post-training, often neglecting enhancements to the foundation model itself. In response, we introduce Act from Visual Language Post-Training (**ActVLP**), a novel training paradigm. **ActVLP** distinctively enhances the foundation model prior to action-specific tuning by first post-training it on a curated set of environment-specific visual and linguistic tasks using self-supervised learning. This initial stage significantly improves the model's capabilities in world knowledge, visual recognition, and spatial grounding. Subsequently, this strengthened VLM undergoes action post-training via imitation learning on trajectory datasets. Following this paradigm, we develop **JARVIS-VLA**, the first VLA model in Minecraft that can follow human instructions on over 1k different atomic tasks, including crafting, smelting, cooking, mining, and killing. Our experiments demonstrate that our **ActVLP** paradigm leads to a significant 40% improvement over the best agent baseline on a diverse set of atomic tasks. Furthermore, **JARVIS-VLA** surpasses traditional imitation learning-based policies in Minecraft, achieving state-of-the-art performance. We have open-sourced the code, models, and datasets to foster further research. The project page can be found at `https://craftjarvis.github.io/JarvisVLA`.

1 Introduction

Pretraining foundation models on large-scale, noisy internet datasets has become a mainstream approach in NLP and vision (Dosovitskiy, 2020; Achiam et al., 2023; Wang et al., 2024b; Team et al., 2023). The success of models like GPT and LLAMA (Touvron et al., 2023; OpenAI, 2023) has shown that large, capable language models can infer and execute tasks described by language prompts. However, this paradigm has yet to achieve similar success in the decision-making domain (Yang et al., 2023; Cheng et al., 2024). In particular, while OpenAI's Video Pre-Training (VPT) model (Baker et al., 2022) has attempted to apply a similar approach in Minecraft, it still relies heavily on imitation learning (IL) after collecting large-scale YouTube videos of human play. VPT's approach of pretraining with imitation learning, followed by downstream supervised fine-tuning and reinforcement learning, made significant strides—culminating in the successful `ObtainDiamond`, a key challenge in Minecraft[1].

Despite this success, the reliance on next-action prediction in imitation learning limits the development of robust, multi-task decision-making abilities (Brohan et al., 2022; Wu et al., 2023; Team et al., 2024; O'Neill et al., 2023). Moreover, this pretraining paradigm struggles to generalize to unseen environments or tasks due to the intricacies of the interactions between observations and behavior, whereas language tokens are more standardized. To overcome these challenges, a new approach has emerged that leverages pretrained Vision Language Models (VLMs) for decision-making. These models, known as Vision Language Action models (VLAs), integrate language understanding with action generation and can be further enhanced through post-training on visual-language tasks (Kim et al., 2024; Zhen et al., 2024). A more detailed discussion can be found in Figure 1 (left) and Section 2.2.

However, much like traditional imitation learn-

---

*Equal contribution

†Corresponding author. Email: yitaol@pku.edu.cn

[1]Diamond tools are considered a grand challenge, with experienced human players taking up to 20 minutes (24,000 actions) to craft them.
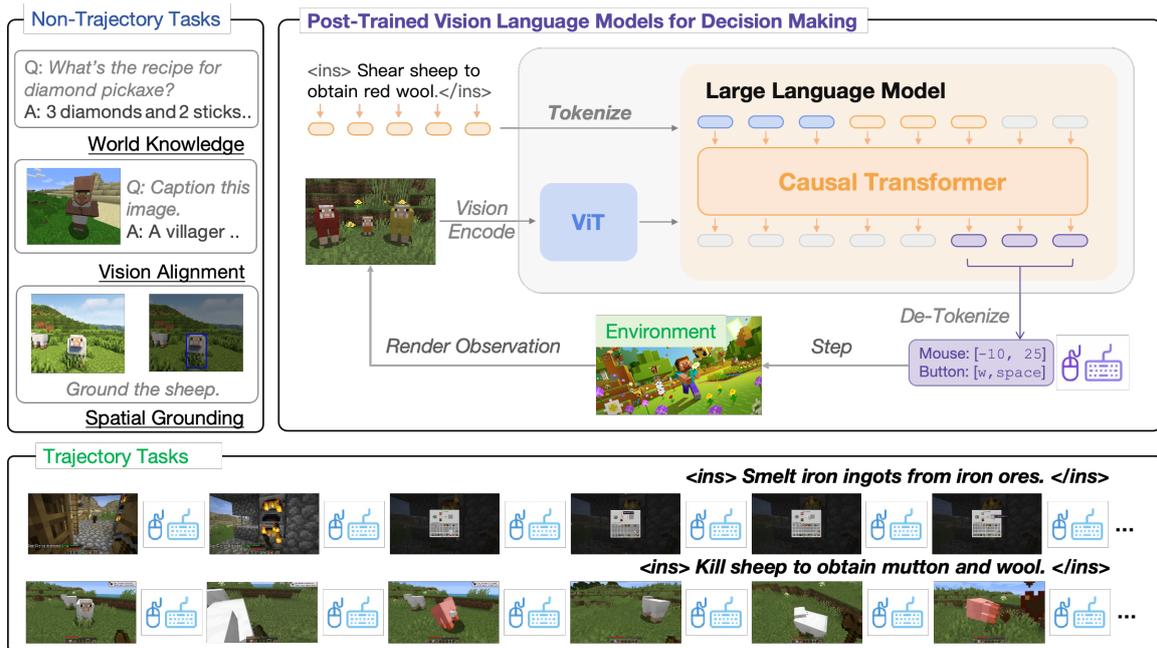
Figure 1: We present **JARVIS-VLA**, a novel Vision-Language-Action (VLA) model trained with **ActVLP** paradigm, post-trained on vision language tasks (non-decision-making tasks) before training on trajectory datasets to have better decision-making capabilities.

ing, current VLA approaches predominantly focus on action post-training. In these models, the learning objective is to generate correct actions based on large-scale cross-task imitation data. We propose that, in addition to action generation, understanding the environment and incorporating task-related knowledge could be equally important for achieving more flexible and generalizable decision-making. To this end, we introduce a novel training paradigm—Vision Language Post-Training (**ActVLP**)—which integrates visual-language tasks into the post-training phase of VLA models. Following the above paradigms, we obtain the first VLA models in Minecraft that can follow human instructions on over 1k different atomic tasks, including crafting, smelting, cooking, mining, and killing.

Our contributions are as follows: (1) We pioneer the use of VLA in the open-world environment of Minecraft by introducing **JARVIS-VLA**, a powerful model achieving state-of-the-art performance in action-based decision-making. (2) We introduce the concept of Vision Language Post-Training and identify key visual-language guidance strategies that enhance decision-making. (3) We investigate the scaling laws of VLA models, demonstrating that expanding the scale of non-trajectory vision-language tasks during post-training leads to significant improvements in downstream task performance. (4) We open-source the code, models, and

datasets to support further research in this area.

2 Learning to Act from Vision Language Post-Training

In this section, we present a detailed introduction to **ActVLP**, a new paradigm for training VLA models. One of the most significant improvements is that we investigate a post-training stage prior to imitation learning. Specifically, we instantiate this paradigm in our proposed model, **JARVIS-VLA**. We begin by discussing the architecture for **JARVIS-VLA** in Section 2.1, followed by an explanation of the training pipeline in Section 2.2 and the datasets used in Section 2.3.

2.1 Model Structure

As illustrated in Figure 1, **JARVIS-VLA** employs an architecture similar to Llava (Li et al., 2024a) but with slight modifications. The structural framework, consists of several key components:: 1) Visual Encoder: A Vision Transformer (Dosovitskiy, 2020) that processes raw image pixels and converts them into a sequence of fixed-size image patches. 2) Image Projection Module: A lightweight two-layer MLP that projects image patch embeddings into the same representational space as word embeddings. 3) Language Model Transformers (Touvron et al., 2023; Bai et al., 2023): A powerful autoregressive language model that serves as the

Traditional VLA Training

Next-Token Prediction Imitation Learning

ViT

<ins> Kill the bigger red sheep. </ins>

Large Language Models

Large-scale Multi-Task Pre-Training on Decision-Making Datasets

Our Multi-Stage Post-Training VLA Pipeline

Next-Token Prediction Supervised Finetuning

ViT

What tool should be used to kill sheep?

Large Language Models

Stage I: World Knowledge Post-Training

Next-Token Prediction Supervised Finetuning

ViT

How many sheep can be seen?

Large Language Models

Stage II: Visual Knowledge and Spatial Grounding Post-Training

Next-Token Prediction Imitation Learning

ViT

<ins> Kill the bigger red sheep. </ins>

Large Language Models

Stage III: Downstream Instruction Following Post-Training

Figure 2: Previous VLA methods usually directly use imitation learning to finetune original vision-language models on large-scale multi-domain decision-making datasets to predict the actions (Kim et al., 2024; Brohan et al., 2023). Our **ActVLP** training pipeline includes three stages: 1) post-training language models on text-only world knowledge with next-token prediction supervised fine-tuning, 2) post-training both vision encoder and language models on multimodal vision-language alignment and spatial grounding datasets with next-token prediction supervised fine-tuning, and 3) post-training only language models on multi-modal instruction following datasets with imitation learning.

core of the system, facilitating multimodal reasoning and decision-making.

Unlike OpenVLA (Kim et al., 2024), our framework is designed for partially observable environments. To accommodate this, we adopt a non-Markovian architecture by incorporating a history of observation images within the prompt. This approach ensures that the model retains temporal context, which is crucial for tasks requiring multi-step reasoning and long-horizon decision-making. In our experiments, we employ Llava-Next (Li et al., 2024a) and Qwen2-VL (Wang et al., 2024b) as base vision language models, as both models provide robust support for multi-image understanding, enabling enhanced perception and reasoning.

Another key distinguishing feature of **JARVIS-VLA** compared to prior VLA models is the integration of an action decoder. This module is responsible for decoding both discrete and continuous actions. For discrete actions, we consolidate related action dimensions into unified categories to reduce redundancy and improve efficiency. For continuous actions, we discretize the action space into bins, which are then mapped to discrete tokens. These tokens are subsequently appended to the vocabulary of the original foundation model, allowing the model to generate both textual and action-based outputs in a unified manner.

Instead of retraining the base VLM's tokenizer, we adopt a strategy inspired by RT-2 (Brohan et al., 2023), repurposing the least frequently used tokens from the language tokenizer's vocabulary to represent action semantics. Specifically, we replace the 51 least-used tokens, allocating 22 tokens for mouse control (e.g., cursor movements) and 29

tokens for special keyboard inputs (e.g., function keys and command shortcuts), which can be found in Appendix A. We introduce no other modifications to the original VLM architecture to maintain model generalizability and ensure broad compatibility across different foundation models. This design choice allows **JARVIS-VLA** to be easily integrated with various pre-trained multimodal models while preserving their inherent capabilities.

2.2 Training Pipeline

Traditional VLA methods typically employ pretrained VLMs and train them via imitation learning on large-scale trajectory data, which includes textual instructions, visual observations, and action token sequences, as illustrated in Figure 2(left). These methods assume that VLMs, pretrained on diverse internet-scale data, possess strong generalization and fitting capabilities. Consequently, they are fine-tuned directly on downstream decision-making tasks, leveraging multi-scenario data to enhance action understanding and generalization.

However, learning world knowledge from action-labeled trajectory data is inherently challenging (Baker et al., 2022). Moreover, the lack of large-scale action-labeled datasets makes it challenging to pretrain expansive models using only trajectory data (O'Neill et al., 2023).

To address these challenges, **ActVLP** enhances the VLM through a structured post-training process, utilizing data that follows the same format as pretraining but is more relevant to decision-making tasks. As shown in Figure 2(right), our training pipeline consists of three stages.

Stage I: Post-Training Language Models. We
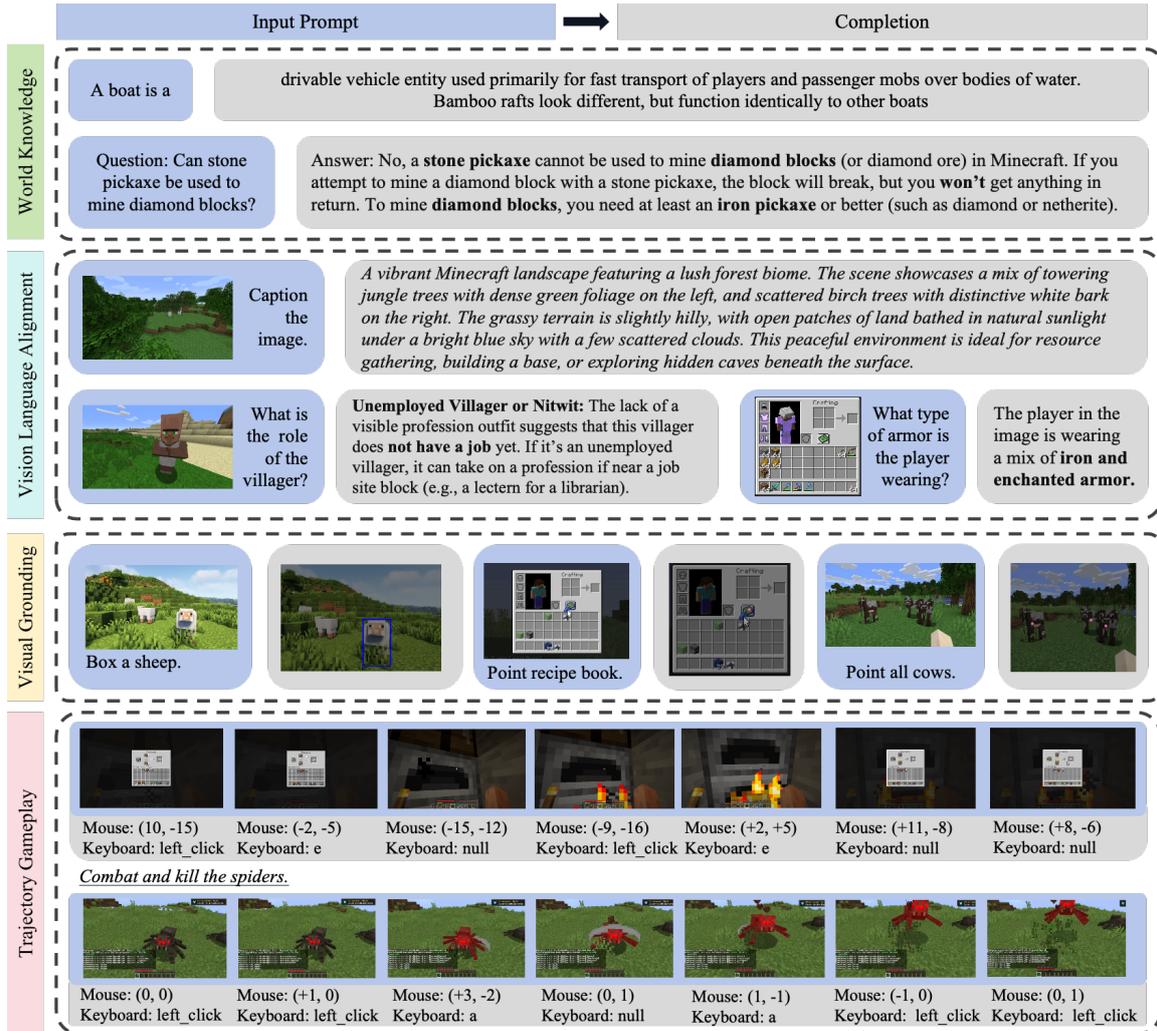
Figure 3: Illustration of various post-training datasets. Models can post-train on various vision-language datasets using a unified tokenizer and support diverse vision-language applications, such as question answering, image captioning, image/video question answering, visual grounding (including points and bounding box), and decision-making. More examples can be found in Appendix D.

first refine the language transformer of the VLM using large-scale textual datasets related to world knowledge in downstream environments, e.g., Minecraft. During this stage, vision-related components, including the ViT and vision adapter modules, are frozen. This step enhances the model's understanding of decision-making contexts before incorporating multimodal alignment.

Stage II: Post-Training Vision Encoder and Language Models. Following language post-training, we fully unfreeze the VLM and fine-tune it using captioning, visual question-answering (VQA), and spatial grounding datasets, which are multimodal and have images in datasets. This stage ensures improved vision-language alignment, enhancing the model's capacity to integrate world knowledge with visual perception. Both Stage 1 and Stage 2 employ next-token prediction through supervised

fine-tuning, with the optimization objective being:

$$\mathcal{L}_{\text{SFT}} = -\sum_{i=1} \log \mathcal{P}_\theta(x_i \mid x_v, x_{\text{ins}}, x_{1:i-1}) \quad (1)$$

where x_v denotes visual tokens, x_{ins} represents the instruction, and x corresponds to the answer. This loss function maintains consistency with the standard causal mask training approach.

Stage III: Action Post-Training for Interaction. The final stage of our pipeline is action post-training, where the VLM transitions from a passive observer to an active agent. During this action post-training, the VLM is fine-tuned on trajectory data ($\mathcal{D}$). The model learns to map textual instructions (x_{ins}) and current visual observations ($o_t \in \mathbb{R}^{H \times W \times 3}$) to action chunks $a_{t:t+\tau}$. The imitation learning objective is to maximize the likeli-

hood of these expert actions:

$$\mathcal{L}_{\text{IL}} = -\sum_{t=1} \log \pi_\theta(a_{t:t+\tau} \mid o_t, x_{\text{ins}}) \quad (2)$$

where π_θ is the learned policy parameterized by θ. We employ action chunking as this technique promotes temporally coherent and improves training efficiency (Kim et al., 2025). In this stage, the vision-related modules remain frozen. while the language tokenizer is modified to incorporate action tokens, and the language transformer undergoes full parameter fine-tuning.

This structured pipeline ensures that the VLM is progressively refined before being adapted to trajectory-based imitation learning, resulting in improved world knowledge acquisition, vision-language alignment and grounding, and action generalization in decision-making tasks.

2.3 Datasets

To support the **ActVLP** training pipeline, we constructed a large-scale multimodal dataset. This dataset includes both non-trajectory task datasets for post-training and trajectory datasets for downstream imitation learning. The non-trajectory datasets are divided into three categories: knowledge-based question answering, vision language alignment, and spatial grounding. These categories are designed to enhance the model's decision-making capabilities before trajectory fine-tuning. For trajectory datasets, we collected over 7.4 million frames of Minecraft gameplay data, including expert actions from diverse sources such as human-playing (Baker et al., 2022), youtube videos, and existing agents (Wang et al., 2024c).

The dataset for *world knowledge comprehension* comprises approximately 277K entries that significantly bolster textual understanding, employed during training stage I. The *visual-language alignment* dataset incorporates 35K keyframes enhanced with advanced Vision-Language Models to produce both captions and question-answer pairs, facilitating multimodal supervised fine-tuning in the subsequent training stage. The *spatial grounding* dataset focuses on detailed object localization, generating over 404K data points that are instrumental in refining spatial understanding for **ActVLP** models. Both the visual-language alignment datasets and the spatial grounding datasets primarily utilize Minecraft observations, which strengthen the VLM's understanding of the world and are used to support training stage II.

Imitation Learning Trajectory Dataset. VLA training is constructed on a dataset of human gameplay trajectories, particularly from the OpenAI contractor dataset in Minecraft (Baker et al., 2022), which includes diverse tasks. We also incorporated an additional 3M rollout frames from VPT (Baker et al., 2022) and JARVIS-1 (Wang et al., 2024c) agents. For structured GUI-based tasks like crafting and smelting, we synthesized 6.4M expert data entries to improve imitation learning. This trajectory data was partitioned into two distinct subsets. First, over 100 random trajectories for each MCU benchmark task were allocated for targeted fine-tuning. The remaining data, approximately 10 billion tokens, was then utilized for action post-training. Representative examples of our datasets are shown in Figure 3, with further details in Appendix D.

3 Experiments

Our experiments (starting from subsection 4.2) aim to address the following questions:

Q1: How do **JARVIS-VLA** compare to *sota* open-world agents and imitation learning methods?

Q2: Is vision language post-training the true cause of the performance improvement?

Q3: Whether VLAs exhibit scaling laws and how **ActVLP** influences them?

Q4: Is **ActVLP** sensitive to different VLM backbones? Due to space constraints, we quickly respond with an affirmative no, detailed experiment discussion deferred to Appendix F.

3.1 Experimental Setup

Evaluation Environment. We use Minecraft 1.16.5 as our experimental platform (Guss et al., 2019). As an open-world game with a substantial knowledge base on platforms such as Reddit and wiki (Fan et al., 2022), Minecraft poses significant challenges to agents while simultaneously offering rich resources for research. To ensure fair comparisons, we align the action and visual observation spaces with those of human players (Baker et al., 2022). Additionally, we hide information unavailable to human players as well, such as agent location and inventory stats.

Benchmark and Evaluation Metrics. We conduct evaluations using two broad benchmarks: (i) the agent's capacity to interact with the Minecraft environment to complete tasks; and (ii) vision-language tasks (e.g., question answering, spatial

Model	Model Size	Mine Blocks		Avg.	Kill Entities		Avg.	Craft Items			Avg.	Smelt Items		Avg.
VPT-BC (Baker et al., 2022)	248M	0.15	0.38	0.33	0.55	0.35	0.44	0.30	0.50	0.45	0.41	0.10	0.00	0.05
VPT-RL (Baker et al., 2022)	248M	0.05	0.35	0.25	0.35	0.25	0.28	0.50	0.30	0.62	0.55	0.05	0.35	0.20
STEVE-1 (Lifshitz et al., 2024)	248M	0.20	0.35	0.54	0.30	0.75	0.38	0.45	0.20	0.70	0.57	0.25	0.40	0.33
GROOT (Cai et al., 2024c)	248M	0.56	0.40	0.67	0.50	0.50	0.52	0.45	0.35	0.25	0.40	0.35	0.25	0.30
MineDreamer (Zhou et al., 2024)	7B	0.25	0.40	0.55	0.30	0.70	0.39	0.50	0.25	0.30	0.42	0.30	0.30	0.30
Qwen2-VL (raw)	7B	0.77	0.60	0.79	0.93	0.80	0.84	0.83	0.53	0.40	0.60	0.03	0.10	0.07
Qwen2-VL (IL)	7B	0.70	0.73	0.75	0.97	0.83	0.86	0.73	0.67	0.50	0.65	0.17	0.37	0.29
JARVIS-VLA-Qwen2	7B	**0.80**	**0.95**	**0.88**	**0.97**	**0.93**	**0.95**	**0.87**	**0.83**	**0.63**	**0.77**	**0.77**	**0.70**	**0.70**

Table 1: Evaluation results of different policies on Minecraft tasks, Each group includes multiple tasks (at least 5), and the Avg. column reports the average success rate within each group. Qwen2-VL, Qwen2-VL (IL) and **JARVIS-VLA**-Qwen2-VL represent the training on the original qwen checkpoint, post-training on only large-scale imitation learning trajectories, and post-trained on VLP intermediate model. Qwen2-VL (**ActVLP**) achieves the highest success rates across all task groups.

grounding) designed to assess the VLM's understanding of Minecraft-specific knowledge. For the instruction-following tasks, we adopt the MCU Benchmark (Lin et al., 2023), focusing on four categories—Mine Blocks, Kill Entities, Craft Items, and Smelt Items—that represent a wide range of typical game-play behaviors in Minecraft. Notably, Craft and Smelt require 2D GUI manipulation through the mouse (covering thousands of item categories), whereas Mine and Kill involve recognizing, navigating, and interacting with targets in a 3D environment. Each category contains at least 5 distinct tasks. For instance, the Mine Blocks category includes mining iron ore with a stone pickaxe, oak logs with bare hands, grass, dirt, and obsidian with a diamond pickaxe. Our evaluation set includes both simpler tasks (e.g., mining oak logs) and more complex ones (e.g., mining obsidian for over 10 seconds) that have proven challenging for prior state-of-the-art agents (Cai et al., 2024c; Lifshitz et al., 2024). We perform each task at least 30 times and report the success rate per task, as well as the average success rate within each category. To ensure fairness, maximum execution steps for selected tasks match those reported by Lin et al. (2023). For vision-language assessments, the task formulations are illustrated in Figure 3. We provide human-written ground-truth answers and employ an LLM-as-judge to evaluate the performance of various VLMs (GPT-4o, Llava, Qwen-VL, and our post-trained VLMs). Detailed information on these vision-language benchmarks and results can be found in Appendix E.

Training and VLA Configurations.. Our training pipeline follows the process described in Section 2.2: we first obtain a visual-language post-training intermediate model, then further train

it on trajectory tasks to produce the **JARVIS-VLA**. We conduct experiments using two popular frameworks: Qwen2-VL (Wang et al., 2024b) and Llava (Li et al., 2024a). We develop a discretized action tokenizer specific to Minecraft, comprising 51 tokens that represent camera movements and button actions. We utilize the trl SFT Trainer (von Werra et al., 2020) for finetuning and deploy the VLA with vLLM (Kwon et al., 2023). Training is carried out on 32 A800-80G GPUs, while inference runs on a single NVIDIA RTX 3090. Further training details are provided in Appendix B.

Baselines. We compare our model with: 1) VPT (Baker et al., 2022), including both the behavior cloning (VPT-BC) and reinforcement learning (VPT-RL) variants; 2) STEVE-1(Lifshitz et al., 2024), a text-conditioned policy that combines VPT and MineCLIP(Fan et al., 2022) for instruction following; 3) GROOT (Cai et al., 2024c), which uses video prompts as task instructions; and 4) MineDreamer (Zhou et al., 2024), which leverages a vision-language model and a diffusion model to guide the STEVE-1 policy. Each method follows the default configuration provided in the MCU benchmark for a fair comparison.

3.2 VLA Performance Evaluation

We present the performance results of our proposed model across four categories from the MCU benchmark (Lin et al., 2023), as shown in Table 1. For each MCU task, we collect over 100 random trajectories, which are used to fine-tune base VLMs to create our final VLA models.

We evaluate three variants of the VLMs as base models: 1) Qwen2-VL (raw): the original VLM checkpoint, fine-tuned directly on the task-specific dataset. 2) Qwen2-VL (IL): first undergoes ac-

tion post-training, then trained on the same task-specific fine-tuning dataset. 3) **JARVIS-VLA**-Qwen2-VL: first post-trained using our proposed **ActVLP** paradigm, then fine-tuned on the same dataset. Performance is measured by the average success rate across tasks within each category. Our results show that **JARVIS-VLA**-Qwen2-VL, post-trained using our approach, consistently outperforms prior methods across almost all tasks.

Remarkably, even without task-specific post-training, raw Qwen2-VL model, fine-tuned on downstream tasks, outperforms several previous baselines, including STEVE-1 (Lifshitz et al., 2024) and GROOT (Cai et al., 2024c), which were trained using large-scale imitation learning. This highlights the effectiveness of using a robust pre-trained VLM as the base model for the policy, leading to strong performance even without additional fine-tuning.

Notably, we observe a significant performance boost with **ActVLP** post-training. For tasks such as Craft Items and Smelt Items, where previous methods struggled, **JARVIS-VLA**-Qwen2-VL achieves success rates more than double those of the baseline models. This underscores the effectiveness of our off-trajectory vision-language task strategy. Furthermore, **JARVIS-VLA**-Qwen2-VL outperforms Qwen2-VL (IL) by over 15%, despite using only 21% of the training trajectory data. In crafting category tasks, the **JARVIS-VLA** model surpasses traditional baselines by more than double, outperforming models like VPT-BC (Baker et al., 2022) and STEVE-1 (Lifshitz et al., 2024) on tasks such as "Craft crafting table" (🗃). This significant improvement is primarily due to the use of ViT in VLM and high-resolution processing, which are crucial for tasks like crafting and smelting that demand precise control in the GUI interface. This suggests that integrating off-trajectory vision-language tasks into the training pipeline enhances decision-making capabilities, enabling more accurate action predictions in VLA models. Further analysis and additional experiments will be presented in the next section.

3.3 Ablation on Training Paradigm

To assess the impact of visual-language post-training, we designed an additional baseline, Qwen2-VL-7B (one-stage). It is co-trained on the original checkpoint of Qwen2-VL, using the combination of non-trajectory datasets and trajectory datasets in a single unified training stage. After

Model	🗡	🗃	🍖	🗆
Qwen2-VL(raw)	0.53	0.40	0.03	0.10
Qwen2-VL-7B (one-stage)	0.10	0.40	0.07	0.13
ActVLP-Qwen2-VL	0.83	0.63	0.77	0.70

Table 2: **Ablation results for different training paradigms.** We use Qwen2-VL (raw) as the baseline and compare it against Qwen2-VL-7B (one-stage) and **JARVIS-VLA**-Qwen2-VL on four Minecraft tasks: "craft diamond sword", "craft ladder", "cook beef", and "smelt iron ingot".

training for one epoch, Qwen2-VL-7B (one-stage) is fine-tuned on MCU tasks. We compare it with **JARVIS-VLA**-Qwen2-VL, which is post-trained in three separated steps.

The results presented in Table 2 indicate that separating visual-language post-training from action learning, rather than merging all stages into a single training phase, yields a significant performance boost. For instance, when comparing our **JARVIS-VLA**-Qwen2-VL to the one-stage co-finetuning approach, we observed substantial improvements, including approximately 57% on the "Smelt iron ingot 🗆" task and 23% on the "Crafting the ladder 🗃" task. We hypothesize that dedicated post-training on visual-language datasets enhances the foundational capabilities of the VLM, providing a more advantageous starting point for subsequent imitation learning.

3.4 Ablation on Non-Trajectory Datasets

In this section, we focus on the post-training of Qwen2-VL using various non-trajectory vision-language tasks to investigate the specific contributions to its enhanced performance.

To understand the impact of different task enhancements, we conduct an ablation study by dividing the non-trajectory datasets and training Qwen2-VL separately on three types of tasks: spatial grounding, vision language alignment, and knowledge-based question-answering, which are all related to Minecraft games. This results in three variants of the VLM, each augmented with one of these capabilities—spatial grounding, visual recognition, and world knowledge. All models are finetuned using the same gameplay dataset and imitation learning techniques. We also develop a benchmark, detailed in Appendix E, to evaluate these capabilities. For this evaluation, we select three long-sequence atomic tasks: "Craft the diamond sword" (🗡), "Mine the obsidian" (⬛), and "Cook the beef" (🍖), as downstream instruction-following tasks.
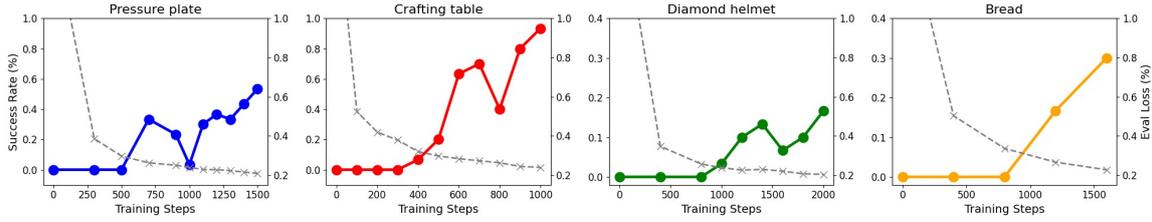
Figure 4: **The relation between downstream task success rate, training loss, and training steps.** The curve shows that scaling downstream finetuning trajectories can scale up the success rate when the loss is lower than 0.22.
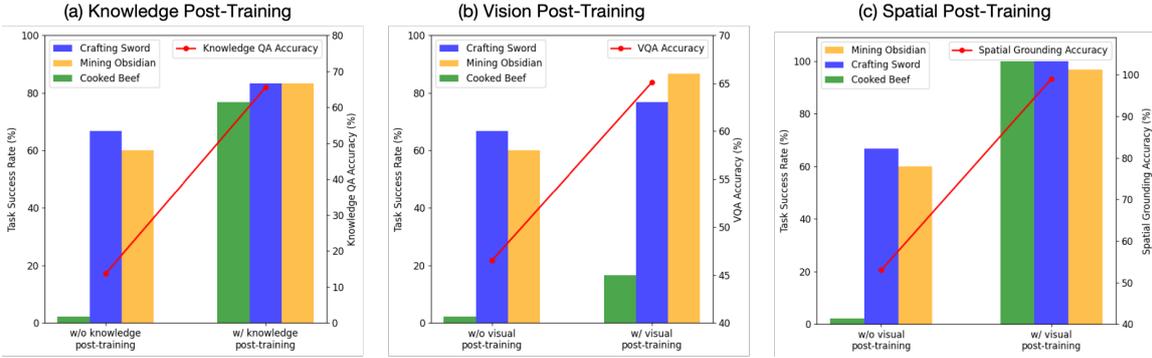
Figure 5: **Ablation results on different post-training datasets.** We select knowledge datasets, visual question-answering datasets, and spatial grounding datasets to conduct ablation experiments. Our goal is to evaluate which capabilities and post-training datasets most significantly influence downstream decision-making tasks.

The results of our ablation studies, presented in Figure 5, demonstrate that post-training with non-trajectory vision-language tasks significantly enhances the core capabilities of the VLM across the respective benchmarks. Notably, after fine-tuning, models enhanced with spatial grounding exhibit the most substantial improvement in downstream decision-making tasks. These findings underscore the effectiveness of non-trajectory post-training in boosting the performance of Vision-Language-Action models in decision-making tasks, even when the focus is on a single task. We find that non-trajectory vision-language tasks, which are essential for agent pipelines (Wang et al., 2023, 2024c), are more effective for fine-tuning end-to-end VLA models. This demonstrates the connection between developing LLM-based agent pipelines with separate modules and fine-tuning end-to-end VLA models.

3.5 Scaling Experiments

Recent work on large language models (LLMs) trained on vast amounts of text via next-token prediction has shown strong scaling laws (Wei et al., 2022; Du et al., 2024; Lin et al., 2024; Wang et al., 2024e). We investigate whether VLAs, obtained through post-training on VLMs, exhibit similar scaling behavior. Specifically, we explore two questions: Q1) Can scaling up downstream imitation
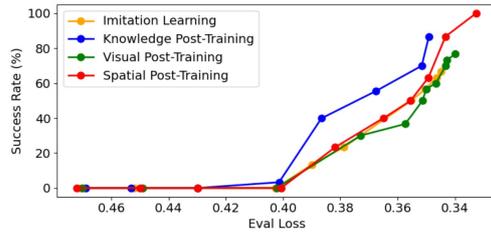
Figure 6: **The relationship between post-training loss and downstream task success rates.** Our findings indicate that increasing the size of post-training non-trajectory datasets can significantly enhance downstream task success rates, even with a fixed number of fine-tuning trajectories.

learning trajectories further improve the VLA's task success rate? Q2) Does increasing the amount of non-trajectory vision-language tasks used during post-training enhance task completion success?

The results for Q1 are shown in Figure 4. Using the same base model, we observe that increasing the number of downstream trajectories improves the VLA model's task success rate. However, since the success rate is a discrete metric, we find that tasks only show a non-zero success rate when the evaluation loss is below 0.30. This indicates that the dataset size for downstream fine-tuning must be sufficiently large enough. Furthermore, we observe that different tasks require varying amounts of downstream data to reduce the evaluation loss below 0.30, which correlates with the length and difficulty of the tasks.

The results for Q2 are illuminated in Figure 6. We also explore the relationship between the evaluation loss during post-training on non-trajectory vision-language tasks and task success rate in downstream tasks. We use base models from different stages of post-training (with different eval loss on post-training datasets), fine-tuning them with the same downstream trajectory dataset. The baseline represents post-training using imitation learning on cross-task trajectories. We find that, for nearly all tasks, the success rate in downstream tasks correlates linearly with evaluation loss in post-training, with the lowest loss yielding the best results. Notably, models post-trained with knowledge-based tasks exhibit the best downstream performance for a given evaluation loss. Models enhanced with spatial grounding show the lowest evaluation loss and the highest task success rates. These findings demonstrate scaling up off-trajectory vision language datasets directly enhances downstream task performance, which has been overlooked in previous VLA works (Kim et al., 2024; Brohan et al., 2023).

4 Related Works

4.1 Visual-Language-Action Models

Imitation learning (IL) involves learning by mimicking expert interactions with the environment, with the primary challenge being the collection of high-quality expert demonstration datasets. Numerous studies have sought to enhance traditional IL approaches (Chi et al., 2023; Team et al., 2024; Brohan et al., 2022). A promising direction is the use of Visual-Language-Action (VLA) models (Brohan et al., 2023; Kim et al., 2024; Zheng et al., 2024; Zhang et al., 2024; Zhong et al., 2025), which adopt end-to-end imitation learning by fine-tuning VLMs. OpenVLA (Kim et al., 2024) has demonstrated the importance of selecting a capable VLM backbone, a conclusion further reinforced by RoboVLM (Li et al., 2024b). Similarly, Brohan et al. (2023) highlighted that co-training with web-scale vision-language data significantly improves the generalization of VLA models. While previous works primarily focused on optimizing the selection of VLMs, several recent studies have begun to pay attention to the comprehension capabilities of VLA models (Zhang et al., 2025; Zhu et al., 2025; Zhou et al., 2025; Chen et al., 2025). However, few have explicitly focused on enhancing the VLM backbone itself through visual-language post-

training. Our work addresses this gap by proposing targeted visual-language post-training methods to enrich the capabilities of VLMs, thereby improving their performance on downstream VLA tasks.

4.2 VLM-based Agents in Minecraft

Existing Minecraft agents based on VLMs typically adopt hierarchical architectures (Wang et al., 2023; Zhou et al., 2024; Fan et al., 2022; Deng et al., 2025; Zhao et al., 2024; Cai et al., 2023). These methods leverage a VLM's world knowledge for planning via zero-shot or few-shot in-context learning, without modifying the VLM parameters during agent optimization (Wang et al., 2023, 2024c,a; Li et al., 2024c). STEVE-EYE (Zheng et al., 2023) fine-tuned Llama language models (Touvron et al., 2023) using internet text data, achieving improved planning over zero-shot prompting. MineDreamer (Zhou et al., 2024) employs the instruction-following capability of VLMs to predict future visual observations and generate actions based on STEVE-1 (Lifshitz et al., 2024). Omni-JARVIS (Wang et al., 2024d) uses a behavior tokenizer (Cai et al., 2024c,b) to model human trajectories in Minecraft with pretrained VLMs. While these approaches optimize VLMs, they still rely on additional policies for action grounding. In contrast, we propose a VLA-based agent model that generates actions directly from textual instructions and visual inputs, eliminating the need for extra grounding policies.

5 Conclusions

We present **ActVLP**, a novel training framework for visual-language-action models that leverages vision-language post-training to enhance decision-making capabilities in dynamic environments. Our experiments demonstrate that post-training on non-trajectory tasks significantly enhances foundation models' ability to understand complex environments, resulting in substantial improvements in downstream imitation learning on trajectory data. The effectiveness of this model is validated across multiple VLM architectures, providing strong evidence of its broad applicability and potential for visual-language-action model training, as exemplified by our state-of-the-art model, **JARVIS-VLA**.

Limitations

Looking ahead, there are several avenues for improvement in future work. First, it is cru-

cial to enhance the inference throughput of the **ActVLP** model, which is currently constrained by the large parameter size of the VLA based on VLM (Budzianowski et al., 2024). We believe that future integration with MoE (Jacobs et al., 1991; Fedus et al., 2022) could further improve the model's inference efficiency, with the goal of achieving gameplay performance levels exceeding 40Hz. Additionally, there remains potential for further performance gains. While **ActVLP** outperforms previous Minecraft policies, it still falls short of the performance demonstrated by top human players, who achieve success rates above 90%.

Acknowledgement

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Introducing the next generation of claude.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. 2022. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Paweł Budzianowski, Wesley Maa, Matthew Freed, Jingxiang Mo, Aaron Xie, Viraj Tipnis, and Benjamin Bolte. 2024. Edgevla: Efficient vision-language-action models. *environments*, 20:3.

Shaofei Cai, Zihao Wang, Kewei Lian, Zhancun Mu, Xiaojian Ma, Anji Liu, and Yitao Liang. 2024a. Rocket-1: Master open-world interaction with visual-temporal context prompting. *arXiv preprint arXiv:2410.17856*.

Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. 2023. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13734–13744.

Shaofei Cai, Bowei Zhang, Zihao Wang, Haowei Lin, Xiaojian Ma, Anji Liu, and Yitao Liang. 2024b. Groot-2: Weakly supervised multi-modal instruction following agents. *arXiv preprint arXiv:2412.10410*.

Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. 2024c. Groot: Learning to follow instructions by watching gameplay videos. In *The Twelfth International Conference on Learning Representations*.

Peng Chen, Pi Bu, Yingyao Wang, Xinyi Wang, Ziming Wang, Jie Guo, Yingxiu Zhao, Qi Zhu, Jun Song, Siran Yang, et al. 2025. Combatvla: An efficient vision-language-action model for combat tasks in 3d action role-playing games. *arXiv preprint arXiv:2503.09527*.

Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, et al. 2024. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint arXiv:2401.03428*.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2023. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. 2024. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*.

Jingwen Deng, Zihao Wang, Shaofei Cai, Anji Liu, and Yitao Liang. 2025. Open-world skill discovery from unsegmented demonstrations. *arXiv preprint arXiv:2503.10684*.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2024. Understanding emergent abilities of language models from the loss perspective. *arXiv preprint arXiv:2403.15796*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. 2019. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, page 79–87.

Moo Jin Kim, Chelsea Finn, and Percy Liang. 2025. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. 2024a. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. 2024b. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. 2024c. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *arXiv preprint arXiv:2408.03615*.

Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. 2024. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36.

Haowei Lin, Baizhou Huang, Haotian Ye, Qinyu Chen, Zihao Wang, Sujian Li, Jianzhu Ma, Xiaojun Wan, James Zou, and Yitao Liang. 2024. Selecting large language model to fine-tune via rectified scaling law. *arXiv preprint arXiv:2402.02314*.

Haowei Lin, Zihao Wang, Jianzhu Ma, and Yitao Liang. 2023. Mcu: A task-centric framework for open-ended agent evaluation in minecraft. *arXiv preprint arXiv:2310.08367*.

Meta. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models.

Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. 2023. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*.

OpenAI. 2023. Chatgpt: Optimizing language models for dialogue.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. 2024. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. 2024. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro,

Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024b. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, Yitao Liang, and Team CraftJarvis. 2023. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 34153–34189.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. 2024c. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Ma, and Yitao Liang. 2024d. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. *Advances in Neural Information Processing Systems*.

Zihao Wang, Haowei Lin, Ruilin Yan, Xiangyu Wang, Jiaqi Li, Weiye Shi, Xiaojian Ma, Anji Liu, Yitao Liang, et al. 2024e. Optimizing inference-time reasoning in llms via retrieval-augmented reflection. *arXiv preprint arXiv:2403.05313*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. 2023. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. 2023. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*.

Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. 2024. Hirt: Enhancing robotic control with hierarchical robot transformers. *arXiv preprint arXiv:2410.05273*.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. 2025. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*.

Guangyu Zhao, Kewei Lian, Haowei Lin, Haobo Fu, Qiang Fu, Shaofei Cai, Zihao Wang, and Yitao Liang. 2024. Optimizing latent goal by learning from trajectory preference. *arXiv preprint arXiv:2412.02125*.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 2024. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. 2024. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*.

Sipeng Zheng, Jiazheng Liu, Yicheng Feng, and Zongqing Lu. 2023. Steve-eye: Equipping llm-based embodied agents with visual perception in open worlds. *arXiv preprint arXiv:2310.13255*.

Yifan Zhong, Xuchuan Huang, Ruochong Li, Ceyao Zhang, Yitao Liang, Yaodong Yang, and Yuanpei Chen. 2025. Dexgraspvla: A vision-language-action framework towards general dexterous grasping.

Enshen Zhou, Yiran Qin, Zhenfei Yin, Yuzhou Huang, Ruimao Zhang, Lu Sheng, Yu Qiao, and Jing Shao. 2024. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*.

Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Ran Cheng, Yaxin Peng, Chaomin Shen, et al. 2025. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. *arXiv preprint arXiv:2502.14420*.

Minjie Zhu, Yichen Zhu, Jinming Li, Zhongyi Zhou, Junjie Wen, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. 2025. Objectvla: End-to-end open-world object manipulation without demonstration.

A Observation and Action Space

We rely solely on visual images for observation, without any symbolic information, similar to VPT (Baker et al., 2022). Notably, we increase the image resolution to 644×364, providing significantly more visual detail than VPT (128×128), which helps **JARVIS-VLA** better perceive the environment.

To closely mimic human behavior, our action space includes all possible player actions except for arbitrary text input. We allocate reserved tokens outside the original VLM vocabulary to represent keypress and click actions. For mouse movements, we follow VPT (Baker et al., 2022), using μ-law encoding to discretize X and Y axes separately into 21 bins (42 tokens total), each mapped to a reserved token. Although Qwen2-VL (Wang et al., 2024b) does not explicitly support reserved tokens like Llama3 (Meta, 2024), we can add special tokens by expanding the vocabulary, since its embedding table is underutilized. Table 3 shows the action space **JARVIS-VLA** uses. During inference, the model predicts actions token by token: key/button actions first, followed by camera movements (Pitch and Yaw).

Table 3: The action space we use in Minecraft.

Index	Action	Human Action	Description
1	Forward	key W	Move forward.
2	Back	key S	Move backward.
3	Left	key A	Strafe left.
4	Right	key D	Strafe right.
5	Jump	key Space	Jump. When swimming, keeps the player afloat.
6	Sneak	key left Shift	Slowly move in the current direction of movement.
7	Sprint	key left Ctrl	Move quickly in the direction of current motion.
8	Attack	left Button	Destroy blocks (hold down); Attack entity (click once).
9	Use	right Button	Interact with the block.
10	hotbar:[1-9]	keys 1 - 9	Selects the appropriate hotbar item.
11	Inventory	key E	Open/Close the inventory.
12	Yaw	move Mouse X	Turning; aiming; camera movement.
13	Pitch	move Mouse Y	Turning; aiming; camera movement.

B Training Configurations

The training configurations for both Visual-Language Post-Training and Action Post-Training are largely consistent. All experiment were conducted on NVIDIA A800-SXM4-80GB GPUs, utilizing CUDA version 12.1 and Hugging Face Transformers version 4.47.0. Both training stages utilized the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay were set to 0, and $\epsilon = 1 \times 10^{-8}$. A cosine learning rate schedule was adopted with the learning rate of 5×10^{-6} and a warmup of 200 steps. The training used `bfloat16` precision, a maximum gradient norm of 1.0, and a fixed random seed of 42. To accelerate training, DeepSpeed with ZeRO-1 (Rasley et al., 2020)optimization was employed. For **Visual-Language Post-Training**, the maximum token length was set to 3584, and we set a batch size per device of 2 and a gradient accumulation of 4 . For **Action Post-Training**, the maximum token length was set to 512, which allowed a batch size per device of 8 and a gradient accumulation of 1 step per update. Ensuring that the total batch size remained 256. Both stages were trained using 32 A800 GPUs, with the Visual-Language Post-Training phase running for 128 GPU hours and the Action Post-Training phase running for 512 GPU hours.

To enhance generalization, distinct data augmentation strategies were adopted for different training phases. In the **Visual-Language Post-Training** phase, modifications included adjustments to hue, saturation, brightness, contrast, as well as random translation, rotation, slight scaling variations, shearing, and occasional flipping. These adjustments extended to bounding box and pointing annotations, with necessary masking of instruction-following prompts. In contrast, the **Action Post-Training** phase focused on adjusting hue, saturation, brightness, contrast, and translation, applied only on images.

C Details of Inference

During inference, **JARVIS-VLA** is expected to output actions in a format consistent with the gameplay dataset, as illustrated below:

Example of JARVIS-VLA Interaction for One Turn of Iteration

Instruction: Craft a bread so I can use it.
Arrange the materials in the crafting grid according to the following pattern:
wheat | wheat | wheat
wheat | wheat | wheat
and get 1 bread.
Observation:

Action:
<|action_begin|><|cam_w_13|><|cam_h_5|><|action_end|>
<|action_begin|><|use|><|cam_w_3|><|cam_h_2|><|action_end|>

Action Chunking Strategy

To optimize both inference speed and predictive performance, we designed an adaptive action chunking strategy that varies the number of predicted actions per observation across different operational phases. The strategy begins with a chunk size of one action per observation during Action Post-Training to ensure precise policy alignment. In the subsequent fine-tuning stages, the chunk size is increased to three to accelerate learning. Finally, during inference, the model generates two actions per prediction step, balancing stability and efficiency.

To validate this multi-phase chunking approach, we conducted experiments on the MCU (Lin et al., 2023) tasks using Qwen2-VL-2B (Wang et al., 2024b) as the base model. The results are presented in Table 4. This ablation study suggests that employing longer action chunks during advanced training stages and a moderate chunk size for inference can improve both success rates and FPS.

Table 4: Ablation study on action chunk sizes. "APT Chunks" denotes Action Post-Training chunks, and "FT Chunks" denotes Subsequent Fine-tuning chunks.

APT Chunks	FT Chunks	Inference Chunks	FPS	Success Rate
1	1	1	8	0.47
1	3	1	8	0.53
1	3	2	15	**0.67**
1	3	3	**21**	0.20
3	3	1	8	0.03

As shown in Table 4, the chunking strategy of using one action per observation during Action Post-Training, three during Further Fine-tuning, and two during Inference ($1 \rightarrow 3 \rightarrow 2$) achieved the highest success rate (0.67) and a solid frame rate (15 FPS) in this specific ablation. We did not explore chunk sizes larger than three, as the Minecraft environment changes rapidly with each discrete action. Predicting multiple future actions in such a dynamic setting poses a significant challenge.

It is also important to clarify the choice of inference chunk size in our main results. To ensure a rigorous and fair comparison with baseline models such as Qwen2-VL (raw) and Qwen2-VL (IL)—which both operate with a single predicted action—we configured **JARVIS-VLA** to use an inference chunk size of one in our primary experiments. While this setting facilitates direct comparison, our ablation results 4 suggest that a chunk size of two at inference may further improve both efficiency and task success under alternative evaluation criteria.

Infrastructure for Efficient Inference

To facilitate efficient inference, **JARVIS-VLA** was served using the vLLM (Kwon et al., 2023) in `bfloat16` precision. This setup was typically deployed on four A800 GPUs. Compared to a standard Hugging Face Transformers implementation, vLLM increased the inference throughput of **JARVIS-VLA** by up to 1.7 times, without introducing any significant degradation in task performance.

Achieved Inference Performance

The final inference speed reflects both the action chunking strategy and infrastructure optimizations. In our main results, which use single-action inference for fair comparison, **JARVIS-VLA**-Qwen2-VL-7B runs at 5.5 FPS, while **JARVIS-VLA**-Qwen2-VL-2B achieves 8 FPS—consistent with the results in rows 1 and 2 of Table 4. Notably, **JARVIS-VLA**-Qwen2-VL-2B can reach 15 FPS with a chunk size of two, highlighting a trade-off we made in favor of direct benchmarking with baseline models.

D   Datasets

In this section, we will introduce the collection methods and specific forms of the visual language post-training datasets we used, including the world knowledge dataset, visual language alignment dataset, and visual grounding dataset.

D.1   World Knowledge Dataset

We sourced content from Wikipedia and the DigMinecraft website, and generating 202K question answering entries using the GPT-3.5-turbo model. The dataset have been adjusted to a uniform format. To increase training efficiency, we randomly merged the original one-question-one-answer data to organize multi-turn dialogue dataset with lengths of approximately 1500-2000 tokens each. Below is the specific format of the world knowledge dataset.

> **Example of World Knowledge Dataset**
>
> **Question**: "What are the different items that can be made using dark oak planks in Minecraft?"
> **Answer**: "In Minecraft, you can use dark oak planks to craft various items and blocks. Here are some of the items and blocks you can make with dark oak planks:
> 1. Dark Oak Wood Slabs: Used for flooring, roofs, and other building elements.
> 2. Dark Oak Wood Stairs: Ideal for creating inclined pathways or staircases.
> 3. Dark Oak Wood Fence: Useful for creating boundaries or enclosures around structures.
> 4. Dark Oak Wood Fence Gate: Allows for openings in your fences for easy access.
> 5. Dark Oak Wood Door: Adds entrances to your buildings.
> 6. Dark Oak Wood Trapdoor: Can serve as hidden entrances or functional hatches.
> 7. Dark Oak Wood Button: Can be placed on surfaces and used to activate various redstone mechanisms.
> 8. Dark Oak Wood Pressure Plate: Activates redstone contraptions when stepped on.
> 9. Dark Oak Wood Sign: Useful for displaying text and providing information in your world.
> 10. Dark Oak Wood Bed: Used to craft the frame of a bed, combined with wool blocks for a complete bed.
> 11. Dark Oak Wood Boat: Useful for water transportation.
> 12. Dark Oak Wood Chest: Can be used to craft storage chests.
> 13. Dark Oak Wood Bookshelf: Decorative and enhances enchantment tables.
> 14. Petrified Oak Slab: Made from dark oak logs submerged in water and then crafted into planks and slabs. These items enhance functionality and aesthetics in your Minecraft builds."
>
> ---
>
> **Question**: "How can players build and use automatic farms in Minecraft, and what are some efficient designs?"
> **Answer**: "In Minecraft, automatic farms are structures designed to automatically harvest resources without requiring constant player intervention. These farms can be extremely useful for gathering various resources efficiently. Here, I'll provide a general overview of how to build and use automatic farms, along with examples of efficient designs for different types of farms:
> 1. Crop Farms: Automated crop farms are used to grow and harvest crops like wheat, carrots, potatoes, and more. Efficient design: - Create a water-based system to hydrate the farmland. A water source in the middle with canals leading to the farmland is a common approach. - Use a dispenser to plant seeds or crops onto the farmland. - Attach a redstone clock to the dispenser to periodically activate it. - Collect the harvested items using hoppers and chests.
> 2. Mob Farms: These farms are used to automatically collect drops from hostile mobs, such as zombies, skeletons, and creepers. Efficient design: - Build a dark room where mobs will spawn naturally. Use trapdoors, pressure plates, or a combination to restrict mob movement. - Use water currents to funnel mobs into a kill chamber. - Create a killing mechanism using either lava blades, suffocation, or other methods. - Use water currents again to collect drops and funnel them into hoppers for collection.
> 3. Animal Farms: Automated animal farms can help you gather resources from passive mobs like cows, pigs, and chickens. Efficient design: - Create separate pens for each type of animal. - Use a breeding mechanism to maintain a stable population. Dispensers with food (like wheat for cows) can automatically breed animals. - Include a killing mechanism for excess animals, providing a steady supply of drops. - Use water currents and hoppers to collect drops into chests.
> 4. Tree Farms: Automated tree farms allow you to grow and harvest trees without manual effort. Efficient design: - Plant saplings in rows using dispensers and bone meal. - Use a mechanism to cut down the trees. This can be achieved using flying machines or TNT-based contraptions. - Use water currents to collect the dropped wood and saplings into hoppers and chests.
> 5. Fishing Farms: Automated fishing farms simulate fishing to gather loot and experience points. Efficient design: - Build a platform with water sources, a fishing rod dispenser, and a mechanism to reel in the fishing rod. - Create a hopper system to collect items caught from the fishing process.
> Remember that the efficiency of these designs can vary depending on your Minecraft version and specific game settings. Additionally, always consider ethical and environmental factors when designing and using automated farms in your gameplay."

D.2 Vision Language Alignment Dataset

For the construction of Vision Language Alignment Dataset, we employed a multi-step approach to gather and process data. Initially, we sourced images from a variety of online platforms, including YouTube gameplay videos, contractor-provided screenshots (Baker et al., 2022), and other internet resources. We carefully selected 35,000 keyframes from these sources based on criteria such as brightness and visual complexity to ensure the quality of the dataset. Subsequently, we utilized advanced Vision-Language Models—such as GPT-4o (OpenAI, 2023), Claude 3.5 Sonnet (Anthropic, 2024), and Molmo (Deitke et al., 2024)—to generate contextual image question-answer pairs, critical in creating a rich, semantically detailed dataset that bridges visual content and linguistic annotations. To enhance the reliability of the generated data, we implemented a robust query-validation pipeline. This pipeline employed Llama-3.1-70B (Dubey et al., 2024) to systematically filter out ambiguous questions and validate the accuracy of the generated answers. Additionally, it included a validation step to ensure that the question-answer pairs

were relevant to the associated images. To further diversify our dataset, we specifically allocated nearly half of the vision-question-answer pairs for caption generation, employing the advanced capabilities of GPT-4o. Through the implementation of our comprehensive pipeline, we successfully generated a Vision Language Alignment Dataset comprising 15,000 captions and 20,000 visual question answering dataset.

Example of Vision Language Alignment Datasets

Question: "What tool is the player holding?"

Answer: "The player is holding a pickaxe with an enchantment glow, likely a diamond pickaxe due to its color."

Question: "What type of block is being mined by the player?"

Answer: "The player is mining a stone block."

Question: "What color is the liquid at the top of the image?"

Answer: "The liquid at the top of the image is orange, indicating it is lava."

Question: "How many inventory slots does the player have in total (visible on the hotbar)?"

Answer: "The player has 9 inventory slots visible on the hotbar."

D.3 Visual Grounding dataset

For Visual Grounding Dataset, we adopted different collection methods for 3D environments and GUI interactions within Minecraft.

In 3D environment, we sample images from the contractor data. We initially identify objects that are interacting with the player at a specific moment in the video. Then implementing Backward Trajectory Relabeling (Cai et al., 2024a) by sam2 (Ravi et al., 2024) to accurately trace the positions of objects up to t frames backward, which tend to be in more random positions.

For 2D GUI interactions, we directly utilized inventory messages provided by Minecraft in rollouts to determine the positions of items within the inventory. Given that each slot in the inventory has a fixed position, we can directly ascertain the points of items. To enhance the randomness of the data distribution, we performed cropping on the data to make the point distribution more uniformly distributed.

In total, we have collected 404K point data from these efforts, mapping real-time entity and object positions during scripted tasks effectively.

Example of Vision Grounding Datasets

Question: "Show the location of torch in the provided image by outlining it with a bbox."

Answer: "<|object_ref_start|>torch<|object_ref_end|><|bbox_start|>(453,333),(563,528)<|bbox_end|>"

Question: "Illustrate the precise position of the iron boots in this image, returning the location in points."

Answer: "<|object_ref_start|>iron boots<|object_ref_end|><|point_start|>(356,446),(386,494)<|point_end|>"

E Benchmarks

In this section, we will introduce our benchmark to test the capability of the Visual Language Models. We also divide the benchmarks into three sections: world knowledge evaluation, visual recognition evaluation, and spatial grounding evaluation. For the first two datasets, we wrote most of them ourselves, while others were obtained from freely available examples on the Internet. We organized them into open question and answer formats. For the spatial grounding evaluation tasks, we adopted the same method to collect data as for the Visual Grounding Dataset D, ensuring that the test set does not overlap with the training dataset. Below is a detailed introduction:

E.1 World Knowledge Evaluation

For World Knowledge Evaluation, we divided it into eight subcategories: craft, plant, mobs, kill, enchant, potion, biomes, and architecture, with a total of 47 test questions. For each entry, we provided the correct answer. A detailed description is shown in Table 5.

Table 5: Summary of Minecraft knowledge questions and answers evaluation.

Category	Quantity	Example Question	Example Answer
Craft	9	What materials are needed to craft a jukebox in Minecraft?	8 Planks and 1 Diamond.
Plant	5	What is the maximum height sugarcane can reach in Minecraft?	3.
Mobs	12	What happens when a creeper gets struck by lightning in Minecraft?	A creeper becomes a charged creeper.
Kill	3	Can shield block crossbow attack in Minecraft?	No.
Enchant	3	What happens if the player puts on an item with the Curse of Binding enchantment in Minecraft?	It cannot be removed until the item breaks or the player dies.
Potion	4	What materials are needed to craft a potion of poison in Minecraft?	Water bottle, Nether wart, blaze powder, spider eye.
Biomes	4	Which biome is the only place you can find blue orchids in Minecraft?	Swamp.
Architecture	7	How many chests are typically found in a shipwreck in Minecraft?	1 to 3.

Knowledge Evaluation

Question: In Minecraft, what type of tool is best suited for digging Mycelium?
GPT-4o: A shovel is the best tool for digging Mycelium. [Explanation: In Minecraft, Mycelium is a block that can be quickly and efficiently broken using a shovel. Like other dirt-based blocks, using a shovel will speed up the process, making it much faster than breaking it by hand or with other tools. Additionally, enchantments like Efficiency on your shovel can further increase the speed at which you can dig Mycelium.]
Qwen2-VL: A Pickaxe. [Explanation: Mycelium is a special type of grass block in Minecraft that can only be found in The Nether, a dimension that can only be entered after the player has crafted the Nether Portal. To dig Mycelium, you will need to have a Pickaxe in your inventory. The Pickaxe is a tool that is used for digging, breaking, and mining blocks. It is the most efficient and effective tool for digging Mycelium because it has a higher efficiency for breaking blocks than other tools.]
JARVIS-VLA-Qwen2-VL: A shovel is the best tool for digging Mycelium in Minecraft.

E.2 Vision Understanding Evaluation

For Vision Understanding Evaluation, we divided it into five subcategories: scene, object, info, inventory, and OCR, comprising a total of 43 test questions. For each entry, we provided an image along with the correct answer. A detailed description can be found in Table 6. Specifically, the Scene category involves the game environment, including biomes, time, and weather; the Object category relates to objects within the player's view; and the Info category includes aspects of the player's experience, health, hunger, and hotbar status.

17896

Table 6: Summary of Vision Understanding Evaluation.

Category	Quantity	Image	Example Question	Example Answer
Scene	11		What biome you think is the player currently in?	Mushroom Fields biome.
Object	13		Are there any hostile mobs in the picture?	No, there are only 4 pigs in the picture.
Info	7		Is the player's hunger bar currently full?	No.
Inventory	6		Is there any oak wood in the inventory?	Yes, there are oak wood planks in the inventory.
OCR	6		What instructions are visible on the screen?	Open your inventory Press e.

Vision Understanding Evaluation Examples

Question: What time of day is it in the game?

GPT-4o: It is difficult to determine the exact time of day in the game due to the rainy weather, which can create a darkened sky. However, based on the darkness of the surroundings, it appears to be night or close to night. Rain in Minecraft makes it harder to see the exact position of the sun or the moon, but the overall dim lighting suggests it is either late evening or night time.

Qwen2-VL: It is daytime in the game. The bright light and the visible sky indicate that it is daytime.

JARVIS-VLA-Qwen2-VL: The time of day in the game is night. This can be determined by the dark sky and the presence of rain in the screenshot. In Minecraft, night begins when the sun sets and the moon rises. During nighttime, the sky darkens, and if it's raining, you'll see rain particles falling from the sky. Additionally, night is a more dangerous time in Minecraft as hostile mobs spawn more frequently.

E.3 Spatial Grounding Evaluation

For Spatial Grounding Evaluation, we used 100 GUI data entries and 236 embodied data entries, as seen in Table 7. We required the model to output the points location of a specified object in the image. If there were no points, the bounding box would be used as a substitute. The output results will be normalized to the range [0, 1000].

Table 7: Summary of spatial grounding evaluation results for visual grounding tasks.

Category	Quantity	Image	Example Question	Example Answer
GUI	100		Point the wheat_seeds	[284,206]
Embodied	236		Point the oak_leaves.	[315,174]

Model	Model Size	World Knowledge		Visual Understanding		Visual Grounding	
		Acc	Rank	Acc	Rank	Acc	Rank
GPT-4o (Achiam et al., 2023)	-	96.6	1	76.7	1	-	-
GPT-4o-mini (Achiam et al., 2023)	-	75.9	2	62.8	4	-	-
Llava-Next (Li et al., 2024a)	8B	19.0	8	41.9	10	-	-
Molmo-d-0924 (Deitke et al., 2024)	7B	12.1	10	58.1	5	24.8	3
Llama-3.2 (Meta, 2024)	11B	20.7	7	44.2	9	-	-
Qwen2-VL (Wang et al., 2024b)	7B	17.3	9	46.5	7	16.6	5
Qwen2-VL (Knowledge)	7B	65.5	5	46.5	7	16.6	5
Qwen2-VL (Vision)	7B	62.1	6	65.1	3	19.8	4
Qwen2-VL (Grounding)	7B	67.2	4	51.2	6	63.6	2
JARVIS-VLA-Qwen2-VL	7B	70.7	3	76.7	1	88.0	1

Table 8: We compared the performance of various VLMs using our benchmark, including commercial large models (GPT-4 and GPT-4-mini (OpenAI, 2023)), open-source models (Llava-Next (Li et al., 2024a), Molmo-d-0924 (Deitke et al., 2024), Llama-3.2 (Meta, 2024), and Qwen2-VL (Wang et al., 2024b)), as well as **JARVIS-VLA**. The results demonstrate that our method significantly enhances the core capabilities of these models, although there remains a gap when compared to state-of-the-art models.

E.4 Evaluation Metric and Result

We designed a customized evaluation method to assess the performance of models in answering the questions. For World Knowledge Questions and Visual Understanding Questions, we explore the utilization of LLMs as judges. We selected GPT-4o (OpenAI, 2023), a state-of-the-art LLM to serve as the judge. The judge model first reviews the responses and compares them to a set of expertly crafted reference answers. Subsequently, the judge assigns a score of correct or incorrect. For visual grounding tasks, we directly score the responses of the evaluated model based on a rule-based approach. Below are the performances of some models we are interested in under our benchmark: Table 8.

F Ablation with different Pre-trained VLMs

In this section, we examine the impact of prior training on a VLMs regarding the robustness of the model's backbone. VLMs vary in their decision-making capabilities due to differences in training data. We highlight this and emphasize the influence of the VLM training architecture on the VLA.

We compare two models, Llava-Next (Li et al., 2024a) and Qwen2-VL (Wang et al., 2024b), which utilize different pretraining datasets and image processing techniques. Their raw VLM performances and post-training results on various auxiliary tasks, along with downstream imitation learning outcomes, are presented in Table ??.

Both Llava-Next and Qwen2-VL demonstrated more than a 30% increase in downstream task success rates after undergoing **ActVLP** post-training. Indicating that improving model performance through visual language post-training is robust across different models.

Model	Diamond Sword	Ladder	Cooked Beef	Iron Ingot
Qwen2-VL(raw)	0.53	0.40	0.03	0.10
Qwen2-VL-7B (one-stage)	0.10	0.40	0.07	0.13
ActVLP-Qwen2-VL	0.83	0.63	0.77	0.70

Table 9: Success rates comparing different training paradigms on selected Minecraft tasks. 'raw' refers to the Qwen2-VL baseline fine-tuned directly. 'RT-2 (1-stage co-finetuning)' corresponds to the Qwen2-VL-7B (one-stage) baseline. 'Act-VLP (3-stage post-training)' represents our **ActVLP**-Qwen2-VL approach.