A. Experimental Details

A.1. Observation and Action Space

The agent receives identical information as human players do. The observation space primarily comprises four components: 1) ego-centric RGB frames, 2) voxels (surrounding blocks), 3) GPS locations (the agent's three-dimensional coordinates), and 4) compass (pitch/yaw angles). These are shaped as $(3, 480, 640)$, $(3, 3, 3)$, $(3,)$, and $(2,)$, respectively. It is important to note that the agent **does not know** the precise location of the target object. Instead, the agent can only obtain information about the target object by examining the pixel image. The RGB frames are resized to a shape of $(3, 128, 128)$ using bilinear interpolation before being fed into the networks. At each step, the agent must execute a movement action, camera action, and functional action. A compound action space is employed, consisting of a multi-discrete space with six dimensions: 1) forward and backward, 2) move left and right, 3) jump, sneak and sprint, 4) camera delta pitch, 5) camera delta yaw, and 6) functional actions (attack and use). The original delta camera degree, which ranges from -180 to 180, is discretized into 11 bins. As this paper's primary focus is on resource collection rather than item crafting, actions related to crafting are omitted.

A.2. Data Collection Pipeline

Our data collection pipeline collects high-quality goal-conditioned demonstrations with actions. The core idea is to train a proxy policy with non-goal demonstrations and roll out in customized environments, then filter the demonstrations according to the achievement. Generally, the pipeline consists of six steps: 1) collect online videos, 2) clean and label the videos, 3) train a proxy policy, 4) customize the environments, 5) roll out the proxy policy, and 6) filter by the accomplishments.

Video-Pretraining [4] is ideally suited for stages 1-3. It begins by amassing a vast dataset of Minecraft videos, sourced from the web using relevant keywords. Given that collected videos often feature overlaid artifacts, the process filters out videos without visual artifacts and those from survival mode. Next, an Inverse Dynamics Model (IDM) is trained to label these videos with actions, yielding demonstrations for proxy policy training. We directly employ the pretrained VPT[4] as our proxy policy. In stage 4, we utilize APIs supplied by MineDojo[16] to create environments tailored to each task's success criteria. During stage 5, we deploy the proxy policy, recording successful trajectories and their corresponding achieved goals. The environment is reset once the episode concludes or the goal is accomplished, ensuring trajectory independence.

Notably, we execute the proxy policy rollout in parallel using 16 processes on 4 A40 GPUs, generating 0.5GB of demonstrations per minute (without leveraging video compression algorithm during storing frames). This approach minimizes human intervention and enhances data collection efficiency. In total, we have gathered 215GB, 289GB, and 446GB of goal-conditioned demonstrations from **`Plains`**, **`Flat`**, and **`Forest`** environments, respectively.

A.3. Implementation

Horizon discretization. As the horizon illustrates the number of steps required to attain the desired objective, it is infeasible to precisely determine the exact value. In practice, we suggest dividing the original horizon into 16 distinct segments: $[0, 10) \rightarrow 0$, $[10, 20) \rightarrow 1$, $[20, 30) \rightarrow 2$, $\cdots$, $[90, 100) \rightarrow 9$, $[100, 120) \rightarrow 10$, $[120, 140) \rightarrow 11$, $\cdots$, $[180, 200) \rightarrow 14$, and $[200, \infty) \rightarrow 15$. In this approach, each segment inherently represents a phase that signifies the level of task completion. Consequently, the horizon prediction issue can be framed as a multi-class problem. It is important to note that the method of discretization is not singular and merits further exploration in the future.

Training. The observation of RGB image is scaled into 128×128 where no data augmentation is adopted. We train the policy with the AdamW optimizer and a linear learning rate decay. We use an initial learning rate of 0.0001, a batch size of 32, and a weight decay of 0.0001. Besides, we also use a warmup trick that the learning rate linearly increases from 0 to 0.0001 in 10k iterations. The policy is trained for 500k iterations on our collected dataset. It takes one day on a single A40 GPU. To train the baseline policies BC (VPT) and BC (CLIP), we only finetune the bias terms of their backbones, which is widely adopted by previous works [7, 23, 50]. Also note that, to keep the architecture comparable, we only transfer model and weights of the backbone from vpt model and MineCLIP model while replace their transformer architecture with ours.

Evaluation. During the evaluation, the maximum episode length is empirically set to 600, 600, and 300 for the **`Flat`**, **`Plains`**, and **`Forest`** biomes, respectively. In most instances, the agent is able to complete the assigned tasks within these limits. Furthermore, in our adaptive horizon prediction module, the hyperparameter c is empirically set to 3. The model is evaluated every 10,000 gradient updates. During each evaluation round, each goal is assessed 10 times to compute the *Success Rate* and *Precision* metrics. For the ablation study, we utilize the checkpoint after 500,000 training iterations, evaluate each goal 200 times, and report the average metrics in Table 5 and Figure 5.

B. Horizon Distribution Analysis

To further emphasize the importance of our adaptive horizon prediction module, we have visualized the distribution of successful trajectory lengths for various tasks in Minecraft, as shown in Figure 6. These successful trajec-

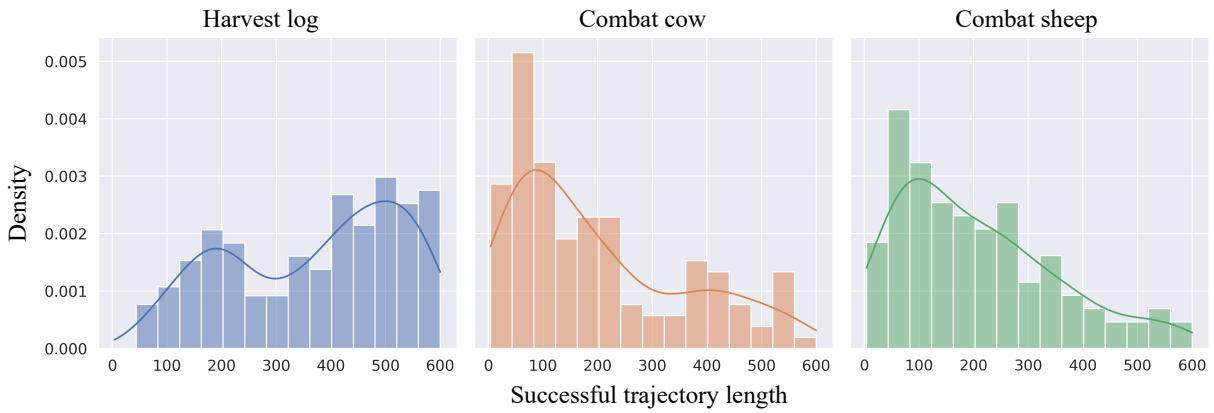

Figure 6. **Successful trajectory distribution of different tasks in open-ended Minecraft**. The distribution is long-tailed, making it hard to learn goal-conditioned policies with a fixed horizon.

tories were gathered from agents trained using single-task behavior cloning (with a randomly initialized Impala CNN as the backbone) in the **`Plains`** biome.

As depicted in Figure 6, the distribution of successful trajectory lengths in the open-world setting exhibits a long tail, making it challenging to train a policy with a fixed horizon. This can be attributed to Minecraft's extensive explorable space, partial observation properties, and non-stationary dynamics, which set it apart from other popular multi-task, closed-ended environments like Meta-World [49].

Consequently, the minimum number of steps needed for an agent to achieve its goal varies across different environments and episodes. The episode length typically hinges on the relative position and terrain constraints between the target object and the agent's initial position. An added layer of complexity arises when no target objects are near the agent's starting location, necessitating large-scale exploration (i.e., a larger horizon). Once the agent locates the target object, it must track it until the relevant skill can be executed on the object (e.g., killing or harvesting). This demands that the agent remain aware of its current stage.

Our proposed adaptive horizon prediction module incorporates the horizon as an additional condition for the policy. The policy explicitly takes into account the remaining time steps needed to achieve specific goals. Our experiments in Section 4.3 demonstrate that the adaptive horizon prediction module and the horizon loss $\mathcal{L}_h$ effectively enhance the success rate in open-world environments with such distributions.

C. Limitation and Future Work

In essence, our approach hinges on trajectories labeled with goals, which enables it to generalize across various domains, provided that such data is accessible. When only video segments labeled with actions are available, we can employ a goal predictor to assign goal labels to these clips. This can also be achieved by utilizing zero-shot models, such as CLIP. Moreover, if action labels are absent in these clips, we can resort to training an inverse dynamics model, as demonstrated in VPT. Undoubtedly, these present intriguing avenues for future exploration