A Algorithms

We present our algorithm sketches for STG Transformer offline pretraining and online reinforcement learning with intrinsic rewards respectively.

Algorithm 1 STG Transformer Offline Pretraining

```
Input: STG Transformer $T_\sigma$, feature encoder $E_\xi$, discriminator $D_\omega$, expert dataset $D^e = \{\tau^1, \tau^2, \ldots, \tau^m\}, \tau^i = \{s_1^i, s_2^i, \ldots\}$, buffer $\mathcal{B}$, loss weights $\alpha, \beta, \kappa$ .
 1: Initialize parametric network $E_\xi, T_\sigma, D_\omega$ randomly.
 2: for $e \leftarrow 0, 1, 2 \ldots$ do                                    ▷ epoch
 3:     Empty buffer $\mathcal{B}$.
 4:     for $b \leftarrow 0, 1, 2 \ldots |\mathcal{B}|$ do                   ▷ batchsize
 5:         Stochastically sample state sequence $\tau^i$ from $D^e$.
 6:         Stochastically sample timestep $t$ and $n$ adjacent states $\{s_t^i, \ldots, s_{t+n-1}^i\}$ from $\tau^i$.
 7:         Store $\{s_t^i, \ldots, s_{t+n-1}^i\}$ in $\mathcal{B}$.
 8:     end for
 9:     Update $D_\omega$: $\omega \leftarrow \text{clip}(\omega - \epsilon \nabla_\omega \mathcal{L}_{dis}, -0.01, 0.01)$.
10:     Update $E_\xi$ and $T_\sigma$ concurrently by minimizing total loss $\alpha \mathcal{L}_{mse} + \beta \mathcal{L}_{adv} + \kappa \mathcal{L}_{tdr}$.
11: end for
```

Algorithm 2 Online Reinforcement Learning with Intrinsic Rewards

```
Input: pretrained $E_\xi, T_\sigma, D_\omega$, policy $\pi_\theta$, MDP $\mathcal{M}$, intrinsic coefficient $\eta$.
 1: Initialize parametric policy $\pi_\theta$ with random $\theta$ randomly and reset $\mathcal{M}$.
 2: while updating $\pi_\theta$ do                                         ▷ policy improvement
 3:     Execute $\pi_\theta$ and store the resulting $n$ state transitions $\{(s, s')\}_t^{t+n}$.
 4:     Use $E_\xi$ to obtain $n$ real latent transitions $\{(e, e')\}_t^{t+n}$.
 5:     Use $T_\sigma$ to obtain $n$ predicted latent transitions $\{(e, \hat{e}')\}_t^{t+n}$.
 6:     Use $D_\omega$ to calculate intrinsic rewards: $\Delta_t^{t+n} = \{D_\omega(e, \hat{e}')\}_t^{t+n} - \{D_\omega(e, e')\}_t^{t+n}$.
 7:     Perform PPO update to improve $\pi_\theta$ with respect to $r^i = -\eta\Delta$.
 8: end while
```

B Environment Details

B.1 Atari

We directly adopt the official default setting for Atari games. Please refer to https://www.gymlibrary.dev/environments/atari for more details.

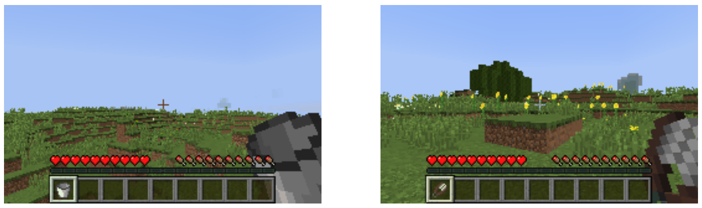
B.2 Minecraft

Environment Settings

Table 1 outlines how we set up and initialize the environment for each harvest task.

Table 1: Environment Setup for Harvest Tasks

Harvest Item	Initialized Tool	Biome
milk	empty bucket	plains
wool	shears	plains
tallgrass	shears	plains
sunflower	diamond shovel	sunflower plains



(a) Plains **(b)** Sunflower Plains

Figure 1: Biomes in Minecraft

Biomes. Our method is tested in two different biomes: plains and sunflower plains. Both the plains and sunflower plains offer a wider field of view. However, resources and targets are situated further away from the agent, which presents unique challenges. Figure 1a and 1b show the biomes of plains and sunflower plains respectively.

Observation Space. Despite MineDojo offering an extensive observation space, encompassing RGB frames, equipment, inventory, life statistics, damage sources, and more, we exclusively rely on the RGB information as our observation input.

Action Space. In Minecraft, the action space is an 8-dimensional multi-discrete space. Table 2 lists the descriptions of action space in the MineDojo simulation platform. At each step, the agent chooses one movement action (index 0 to 4) and one optional functional action (index 5) with corresponding parameters (index 6 and index 7).

Table 2: Action Space of MineDojo Environment

Index	Descriptions	Num of Actions
0	Forward and backward	3
1	Move left and right	3
2	Jump, sneak, and sprint	4
3	Camera delta pitch	25
4	Camera delta yaw	25
5	Functional actions	8
6	Argument for “craft”	244
7	Argument for “equip”, “place”, and “destroy”	36

C Offline Pretraining Details

Hyperparameters. Table 3 outlines the hyperparameters for offline pretraining in the first stage.

Network Structure. Different architectures for feature encoding are designed for different environments. In Atari, we stack four gray-scale images of shape (84,84) to form a 4-channel state and use the feature encoder architecture as shown in Figure 2a. In Minecraft, a 3-channel image of shape (160,256,3) is directly regarded as a single state, which is processed by a feature encoder with more convolutional layers and residual blocks to capture more complex features in the ever-changing Minecraft world. The detailed structure of the feature encoder for Minecraft is illustrated in Figure 2b. All discriminators, taking in two 512-dimension embeddings from the feature encoder, follow the MLP structure of FC(1024,512)→FC(512,256)→FC(256,128)→FC(128,64)→FC(64,32)→FC(32,1) with spectral normalization.

Representation Visualization. We draw inspiration from Grad-CAM [47] to visualize the saliency map of offline-pretrained feature encoder to assess the effectiveness and advantages of the representation of STG. Specifically, we compare the visualization results of STG and ELE in the Atari environment as illustrated in Figure 3. Each figure presents three rows corresponding to the features captured by the three layers of the convolutional layers, respectively. The saliency maps demonstrate that STG exhibits a particular focus more on local entities and dynamic scenarios and effectively ignores extraneous distractions. As a result, compared with ELE, STG shows greater proficiency in

Table 3: Hyperparameters for Offline Pretraining

Hyperparameter	**Value**
STG optimizer	AdamW
Discriminator optimizer	RMSprop
LR	1e-4
GPT block size	128
CSA layer	3
CSA head	4
Embedding dimension	512
Batch size	16
MSE coefficient	0.5
Adversarial coefficient	0.3
TDR coefficient	0.1
WGAN clip range	[-0.01,0.01]
Type of GPUs	A100, or Nvidia RTX 4090 Ti

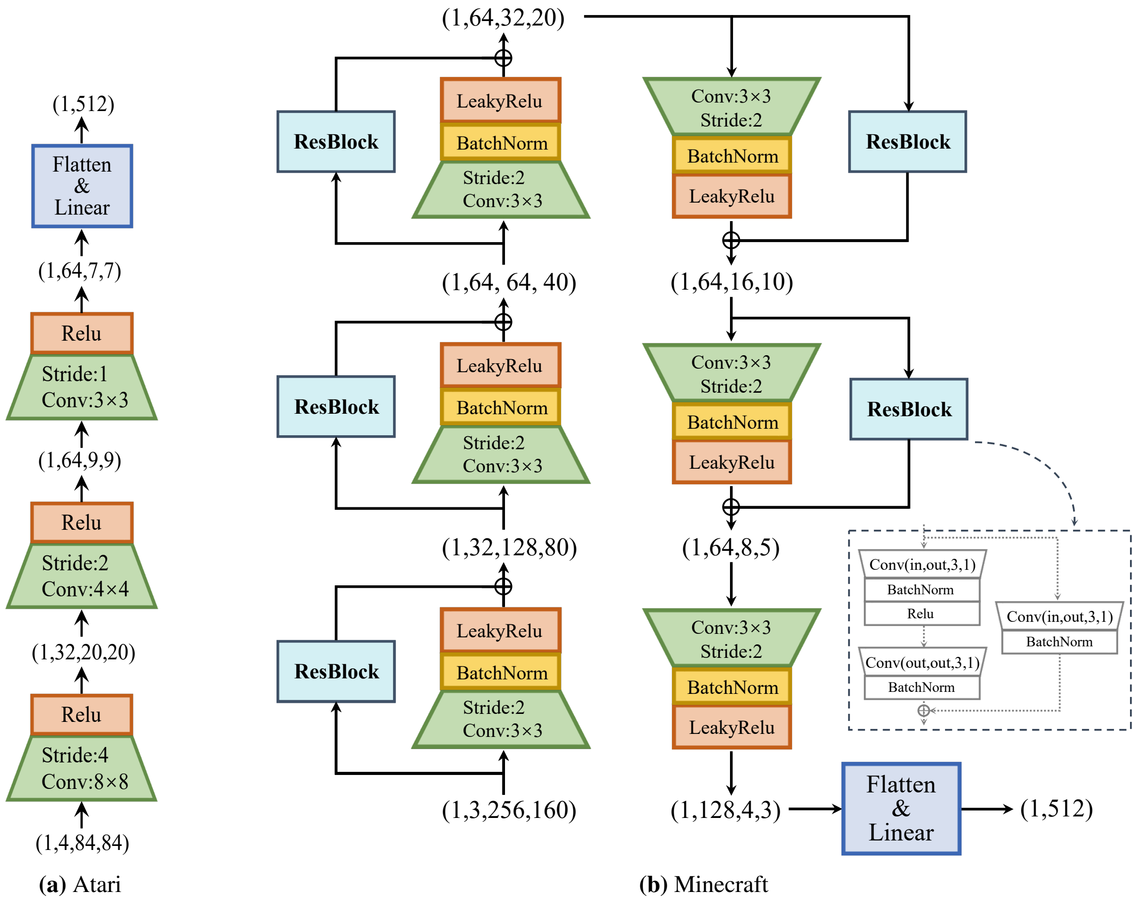
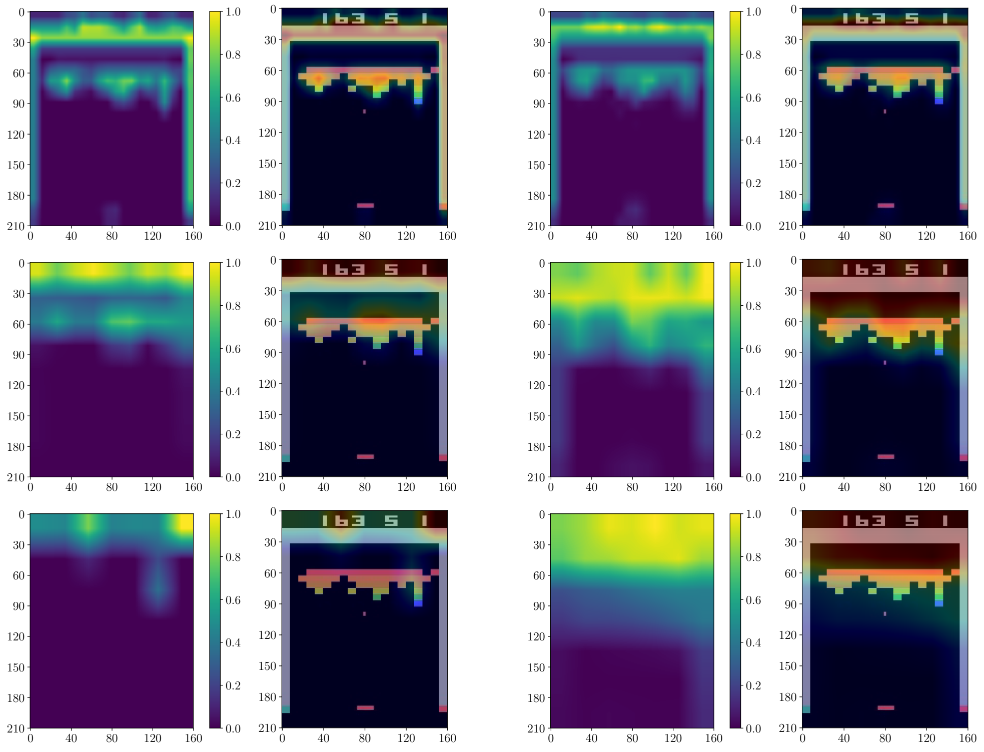
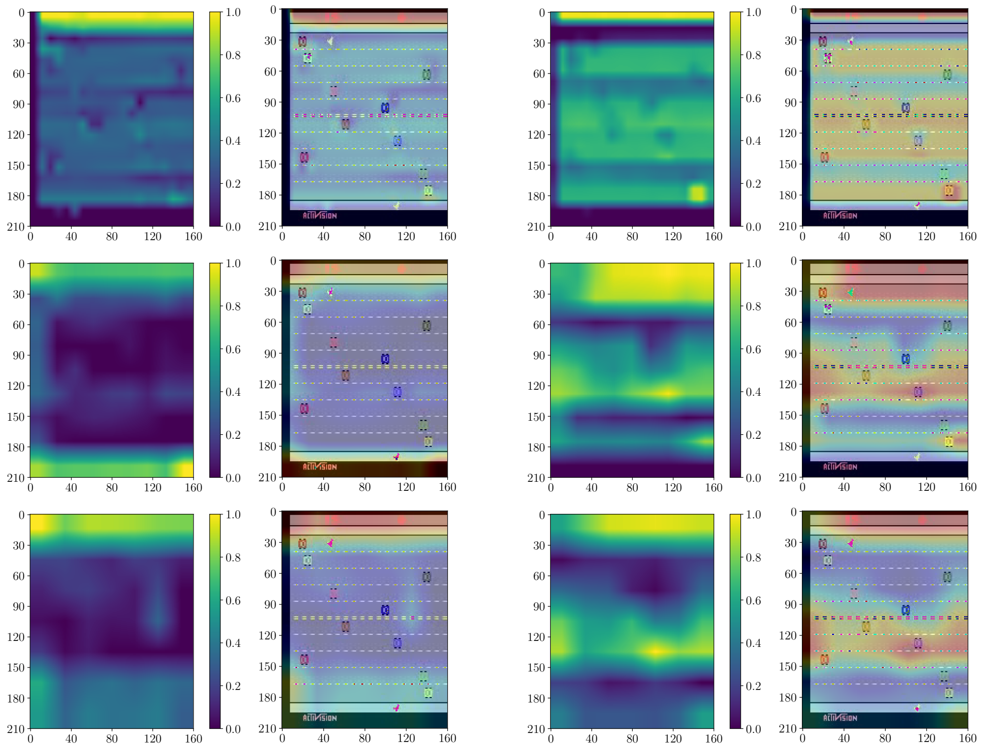
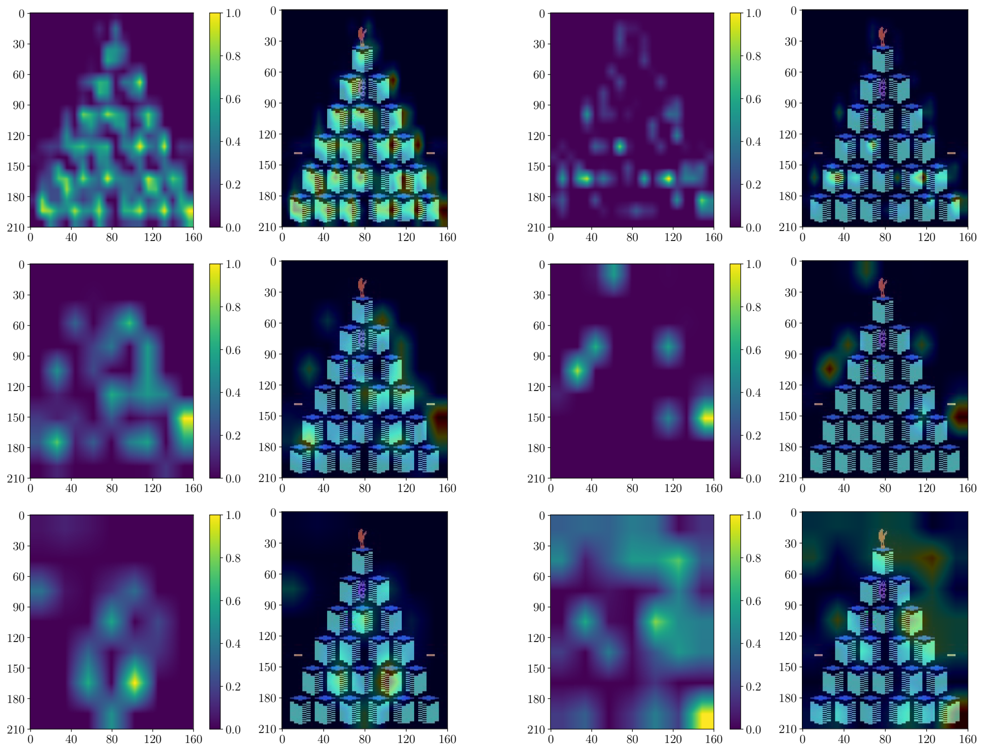


Figure 2: Feature encoder structure for Atari and Minecraft

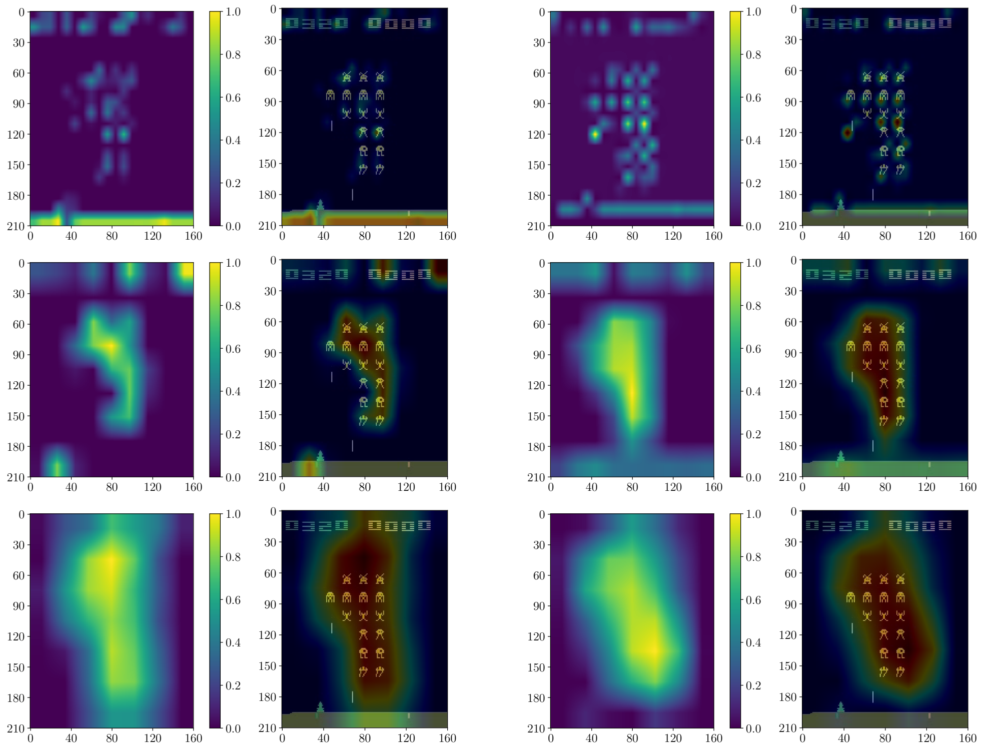
identifying information strongly correlated with state transitions, thereby generating higher-quality rewards for downstream reinforcement learning tasks.


(a) Saliency maps of different CNN layers in Breakout


(b) Saliency maps of different CNN layers in Freeway



(c) Saliency maps of different CNN layers in Qbert



(d) Saliency maps of different CNN layers in Space Invaders

Figure 3: Saliency maps (SM) of different CNN layers in Atari tasks. The first two columns display the normalized saliency maps and corresponding observations of STG and the last two columns represent SM and corresponding observations of ELE. Through comparison, STG is better at capturing fine-grained features which are strongly correlated with transitions.

D RL Training Details

The general training hyperparameters of PPO for downstream RL tasks in the second stage are listed in Table 4.

Table 4: General Hyperparameters for PPO

Hyperparameter	Value
Optimizer	Adam
Learning rate	2.5e-4
RL discount factor	0.99
Number of workers (CPU)	1
Parallel GPUs	1
Type of GPUs	A100, or Nvidia RTX 4090 Ti
Minecraft image shape	(160,256,3)
Atari stacked state shape	(84,84,4)
Clip ratio	0.1
PPO update frequency	0.1
Entropy coefficient	0.01

Neither the discriminative reward from STG nor the progression reward from ELE is bounded. Therefore, it is reasonable to adjust certain hyperparameters to bring out the best performance of each algorithm in each task. In Table 5, the coefficient of intrinsic reward $\eta(\eta > 0)$ for different baselines is tuned to balance the value scale and GAE $\lambda(0 < \lambda < 1)$ is tuned to adjust the impact of intrinsic rewards in different tasks.

Table 5: Specific Hyperparameters for Different Tasks

Task	η_{STG}	η_{ELE}	η_{GAIfO}	λ_{GAE}
Breakout	0.6	1.0	2.0	0.1
Freeway	2.0	0.1	1.0	0.15
Qbert	5.0	0.05	2.0	0.95
Space Invaders	6.0	0.1	2.0	0.95
Milk a Cow	1.0	0.5	-	0.8
Gather Wool	10.0	0.1	-	0.8
Harvest Tallgrass	1.0	0.1	-	0.95
Pick a Flower	1.0	0.1	-	0.95

The coefficients of STG* (noted as $\eta r^i + \nu r^*$) in four Atari tasks are reported in Table 6.

Table 6: Coefficients for STG* in Atari Tasks

Task	η	ν
Breakout	0.6	0.01
Freeway	2.0	0.1
Qbert	5.0	0.03
Space Invaders	6.0	0.01

Training Details. For Minecraft tasks, we adopt a hybrid approach utilizing both PPO [40] and Self-Imitation Learning (SIL) [48]. Specifically, we store trajectories with high intrinsic rewards in a buffer and alternate between PPO and SIL gradient steps during the training process. This approach allows us to leverage the unique strengths of both methods and achieve superior performance compared to utilizing either method alone [46].

E Further Experiments

Intrinsic Reward Design. In Equation (9), we define our intrinsic reward r^i as the difference between r^{guide} and r^{base}:

$$r_t^i = D_\omega\big(E_\xi\left(s_t\right), E_\xi\left(s_{t+1}\right)\big) - D_\omega\big(E_\xi\left(s_t\right), T_\sigma\left(E_\xi\left(s_t\right)\right)\big) = r_t^{guide} - r_t^{base}. \tag{10}$$

On the one hand, pretrained D_ω clearly provides informative judgment r^{guide} of transition quality during online interaction. On the other hand, the baseline reward r^{base}, solely relying on the current state s_t, serves as a baseline to normalize r^i to a relatively lower level. In this section, we aim to investigate the necessity of incorporating r^{base}.

To assess the significance of r^{base}, we conducted experiments on the four Atari tasks utilizing only r^{guide} as the intrinsic reward, which is similar to previous adversarial methods like GAIfO [3]. In order to bring the scale of r^{guide} in line with r^i, we employ running normalization and bound the values within the range of $[-1, 1]$ to mitigate the negative influence of outliers. All other settings remain unchanged. We denote this ablation baseline as STG'.

As illustrated in Figure 4, r^{guide} yields comparable final performance in Breakout and Space Invaders while failing to achieve satisfactory performance in Freeway and Qbert. In contrast, by leveraging r^{base}, which provides a unique reference from expert datasets for each individual s_t, we observe reduced variance and improved numerical stability compared to the running normalization trick that calculates batch means for normalization.

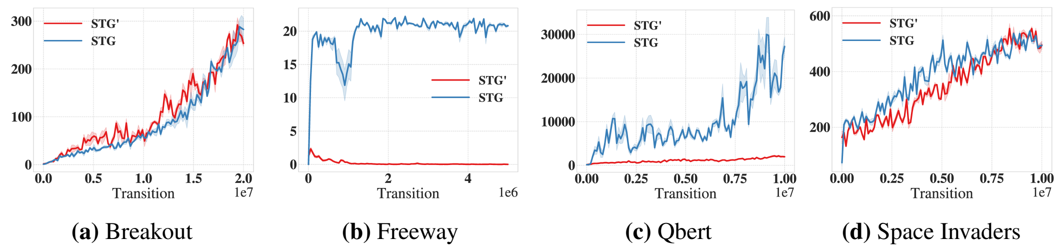


Figure 4: Atari experiments comparing using r^{guide} (STG') and r^i (STG) as intrinsic reward.

F Additional Ablations

Multi-Task STG. Given the four times larger datasets, we enlarge the size of the STG Transformer by increasing the number of heads (24) and layers (16) within the multi-head causal self-attention modules, augmenting the model capacity by about four times. Here we further assess the efficacy of multi-task adaptation of the STG Transformer in Minecraft. We tune the intrinsic coefficient η to be 5 for Breakout, Qbert, SpaceInvaders, "milk a cow", "harvest tallgras", "pick a flower", and 10 for Freeway and "gather wool". STG-Multi results in Minecraft are illustrated in Figure 5. As all four tasks share a similar biome in MineDojo, STG-Multi may provide less clear guidance for the agent than STG in downstream tasks, which may cause their discrepancy in performance.

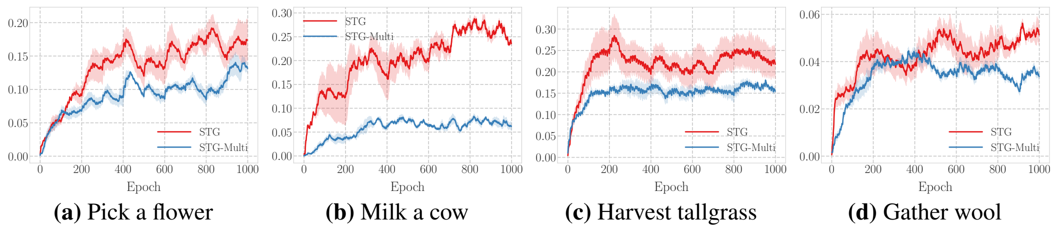
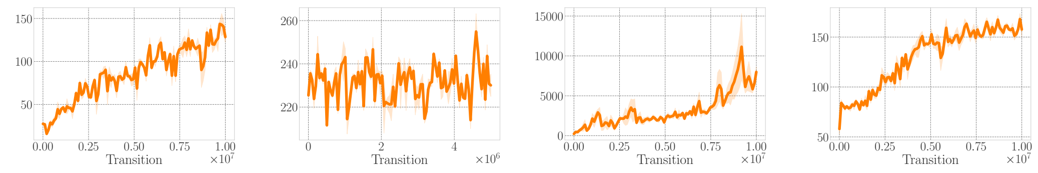


Figure 5: Multi-task STG (STG-Multi) is pretrained on the whole Minecraft datasets to guide RL training in downstream tasks..

G Additional Visualizations

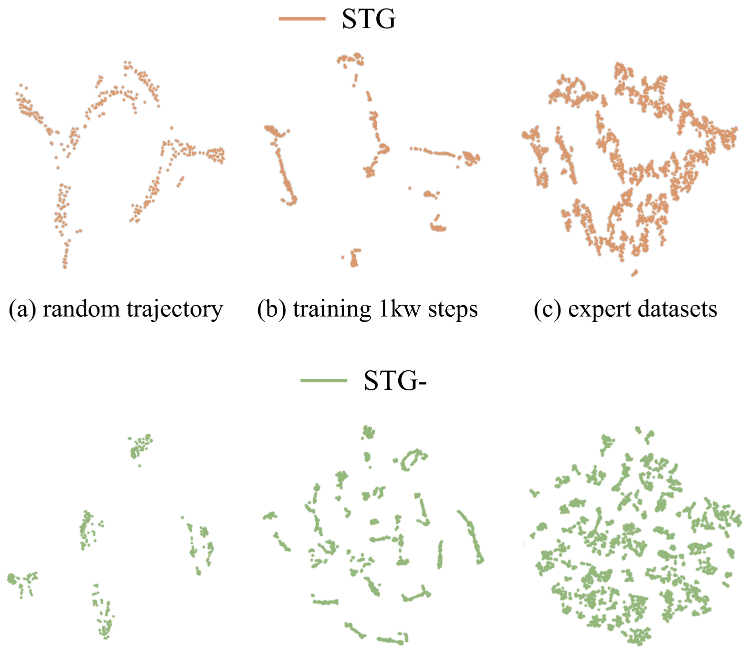
Intrinsic Return. As no environmental rewards participate in updating policy, the ultimate objective of LfVO is to maximize the expectation of cumulative dense intrinsic rewards, namely intrinsic return. Figure 6 shows the learning curves of smoothed intrinsic return in Atari and Minecraft. The rising trend proves that online collected observation distribution is getting closer to expert observation distribution during training, indicating the effectiveness of the offline-pretrained Modules.



(a) Breakout **(b)** Freeway **(c)** Qbert **(d)** Space Invaders

Figure 6: The ultimate objective of LfVO is to maximize the expectation of cumulative dense intrinsic rewards, namely intrinsic return. The rising trend proves that online collected observation distribution is getting closer to expert observation distribution.

Multi-Trajectories Visualization. As an extension to increase diversity, we additionally visualize more trajectories in SpaceInvaders in Figure 7. We randomly sample five trajectories in the expert dataset and use t-SNE to visualize the embedding sequences encoded by STG and STG- respectively. Expert trajectories of STG exhibit more continuity in adjacent states compared with STG-. This is consistent with the visualization results in Figure 6. Furthermore, it can be observed that after 10M-step of training, the observation distribution is getting closer to expert, and the patterns of STG are closer to exert in comparison with STG-. This reflects that our WGAN-style training indeed generates meaningful reward signal to imitate the behavior of experts and TDR module accelerate the process.



(a) random trajectory (b) training 1kw steps (c) expert datasets

(a) random trajectory (b) training 1kw steps (c) expert datasets

Figure 7: T-SNE visualization of embeddings of a random trajectory, a rollout trajectory and expert trajectories in SpaceInvaders.