Building Visual Semantic Bias in Curious Exploration during Free Play

Thesis
submitted in partial fulfilment of the requirements for the degree

Master of Science

Graduate School of Neural Information Processing

Faculty of Science
Faculty of Medicine

Eberhard-Karls-Universität Tübingen

Presented by

Pulkit Goyal

from *Dewas, India*

Tübingen; May 15, 2024

Thesis Advisor: Prof. Dr. Georg Martius
Department of Computer Science

Second Reader: Dr. Charley Wu
Tübingen AI Center

Disclosures:

- I affirm that I have written the dissertation myself and have not used any sources and aids other than those indicated.
- I affirm that I have not included data generated in one of my laboratory rotations and already presented in the respective laboratory report.

Date / Signature: May 15, 2024

Contents

List of Figures

List of Tables

Abstract

Humans have a propensity to explore their environments in expressive manners that manifest their previous knowledge and understanding of the world. Particularly, we tend to exhibit a semantics bias when there is no definite goal at hand, i.e. during free play. For example, when playing with building blocks or doodling, we tend to create patterns and structures that are objectively meaningful such as a house, a car, or a tree. In this study, we tried to realize this semantically expressive behavior in artificial agents. We formulated an entropy-based intrinsic reward for self-supervised exploration during free play using large vision language models (VLMs). This was used in a model-based planning paradigm in special environments with building blocks as their basic elements, which enabled rich creative expression. With the right parameters, an agent optimizing its actions for this reward was able to create a diverse set of abstract creative structures reminiscent of real objects. To improve our results, we further experimented with different regularization methods and investigated if a complementary reward for regularity and compression rooted in a similar bias towards symmetry promoted semantic expression. This work provides a novel perspective on imbuing creativity in artificial agents and furthers the use of VLMs as a source of semantic rewards in robotics and AI.

Chapter 1

Introduction

Humans are capable of exploring and learning about the world in a self-supervised free-form manner with no explicit predefined goal. Children spend substantial time fiddling with toys, scribbling, or playing with building blocks, eventually creating patterns and structures that are meaningful to them. This *free play* has been established as a crucial component for the development of cognitive skills, acquisition of knowledge about the world, and adaptation to new environments (Gibson, 1988; Chu and Schulz, 2020).

As we constantly seek to understand the meaning (semantics) of the objects and scenes we encounter, free play in humans is deeply intertwined with visual perception, and our inherent preferences for visual semantics play a pivotal role in guiding our curiosity-driven exploration. In a study on free play conducted by Diggs-Galligan et al. (2021), participants predominantly showed a preference for semantic expression in the form of regular and symmetric patterns. This suggests that humans have an intrinsic bias or motivation towards visual semantics, i.e. a propensity to explore their environments in expressive styles that manifest their previous knowledge and understanding of the world.

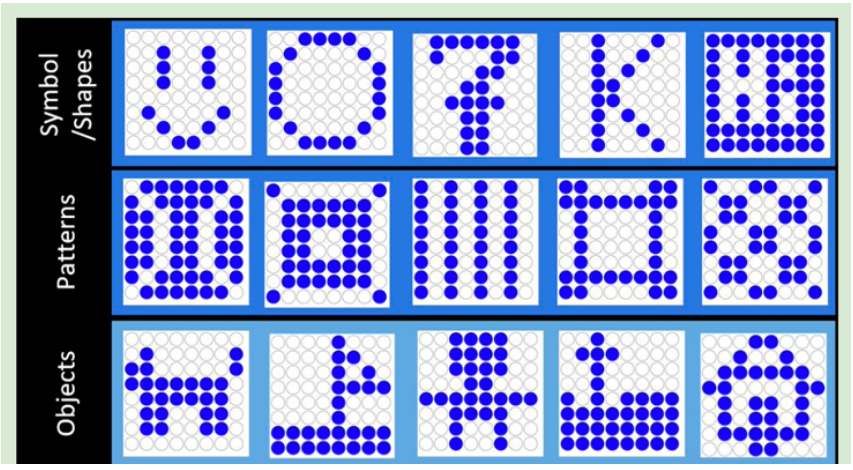


Figure 1.1: Some creations from the free-play study on humans by Diggs-Galligan et al. (2021). (Taken from the original publication.)

In the context of artificial intelligence, self-supervised learning has been a topic of interest for researchers in the fields of robotics and machine learning (ML), and the ability to efficiently and sufficiently explore remains a fundamental challenge. In recent years, reinforcement learning (RL) has emerged as a powerful framework for training agents to learn complex behaviors through trial and error. However, most RL algorithms are designed to solve specific tasks and are not well-suited for free-form exploration. Instead, complementary formulations of novelty-based or uncertainty-based exploration methods have been proposed to encourage agents to explore their environment in a self-supervised manner (Burda et al., 2018;

Pathak et al., 2017, 2019; Ladosz et al., 2022). Yet, these methods are often unable to generate diverse and creative behaviors that are characteristic of human exploration.

This thesis focuses on the role of visual semantic bias in shaping these choices, proposing that our innate visual preferences act as a compass, directing our attention and interactions during free play. Our work tried to imbue artificial agents with a visual semantics bias, akin to that in humans. We do this by leveraging large vision language models (VLMs). Also termed "foundational models", these deep neural networks are trained on massive generic text and image datasets using the recent advances in self-supervised learning. The abstractions of natural language, which reflect the semantics of the world, allow them to learn efficient representations that essentially encapsulate human visual understanding.

VLMs are capable of successfully transferring to a diverse range of downstream applications, such as visual detection, classification, and question-answering. However, the use of foundation models for control and embodied intelligence is relatively new and under-explored. Recent work (Cui et al., 2022; Baumli et al., 2023; Rocamonde et al., 2023; Adeniji et al., 2023) has shown that they can be used as effective abstractions to generate zero-shot rewards for language-guided goal-conditioned tasks. There are also other studies on using VLMs for exploration (Mu et al., 2022; Tam et al., 2022) that use them for refining intrinsic reward signals by abstracting away pseudo-novelty.

Yet this is fundamentally different from our goal as we are specifically interested in creative semantic expression. Moreover, the existing studies do not consider the free-form exploration paradigm that is of interest to us; they require a specific goal to be achieved, either self-generated or manually defined. We instead use VLMs to generate exploratory intrinsic rewards that incentivize the agent to play with the environment and build something meaningful. This reward is based on minimizing the entropy of a VLM's predictions over a set of creative possibilities that the environment offers. Without any explicit goal specification, our controller exploits this reward to guide the agent to semantically expressive states, i.e. to automatically converge to a state that the VLM finds confidently meaningful.

Furthermore, since there is an implicit structure in meaningful creations, and given the compositional strategies for creative expression learned by humans during development (Zingrone, 2014; Golomb, 1987) that favor symmetry and uniformity, we hypothesized that a complementary reward for this regularity (Sancaktar et al., 2023) could promote semantic expression.

We tested our formulations in two rich creative environments: the puzzle Tangram and a pixel grid, where we were able to achieve the desired semantically expressive behavior in our planning agent.

This thesis provides a novel perspective on imbuing free-form creativity in artificial agents and furthers the use of VLMs as a source of rewards in robotics and AI.

Chapter 2

Methods

We use a family of VLMs called CLIP (Radford et al., 2021) for semantic inference and a model-based planning controller using an improved Cross-Entropy Method (iCEM) (Pinneri et al., 2021) as our artificial agent. This chapter describes the components of our approach, including the mathematical setup, the CLIP model, the intrinsic semantics rewards, and the iCEM controller.

2.1 Preliminaries

We formulate the problem as a *fully observable* Markov Decision Process (MDP) given by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \theta, f, r, \mathbf{s}_0, \mathbf{o}_0)$, with state-space $\mathcal{S} \in \mathbb{R}^{n_s}$, action-space $\mathcal{A} \in \mathbb{R}^{n_a}$, observation-space (image-space) $\mathcal{O} \in \mathbb{R}^{n_o}$, rendering function $\theta : \mathcal{S} \mapsto \mathcal{O}$, transition kernel (environment model) $f : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, reward function $r : \mathcal{S} \times \mathcal{O} \mapsto \mathbb{R}$, initial state $\mathbf{s}_0 \in \mathcal{S}$, and its corresponding initial image observation $\mathbf{o}_0 = \theta(\mathbf{s}_0), \mathbf{o}_0 \in \mathcal{O}$.

A trajectory $\boldsymbol{\tau}_t = \{(\mathbf{s}_t, \mathbf{o}_t, R_t, \mathbf{a}_t), (\mathbf{s}_{t+1}, \mathbf{o}_{t+1}, R_{t+1}, \mathbf{a}_{t+1}), \ldots\}$ at time t is a sequence of observation-action pairs starting from the current state $\mathbf{s}_t$, where $\mathbf{s}_k \in \mathcal{S}$, $\mathbf{s}_k \in \mathcal{O}$, $R_k \in \mathbb{R}$, and $\mathbf{a}_k \in \mathcal{A}$.

In a typical RL setting, the returns of such a trajectory $\boldsymbol{\tau}_t$ are the discounted sum of rewards given by,

$$g(\boldsymbol{\tau}_t) = \sum_{k=0} \gamma^k R_{t+k}, \tag{2.1}$$

where R_t is the expected reward at time t and $\gamma \in [0, 1]$ is the discount factor. The agent's goal is to learn an action policy that maximizes its expected reward;

$$\pi_t^* = \underset{\pi}{\operatorname{argmax}} \; E_{\boldsymbol{\tau}_t \sim \pi} \left[g(\boldsymbol{\tau}_t) \right], \tag{2.2}$$

where the expectation is under trajectories *rolled-out* with $\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t)$, $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$, $\mathbf{o}_t = \theta(\mathbf{s}_t)$, and $R_t = r(\mathbf{s}_t, \mathbf{o}_t)$.

In our formulations, we consider a finite-horizon problem, where the agent is tasked with maximizing the expected return over a limited planning horizon h.

We use boldface to denote

2.2 Vision-Language Models

We define vision-language models (VLMs; Zhang et al. (2024)) as functions capable of processing both language inputs $\boldsymbol{l} \in \mathcal{L}$ and vision inputs $\boldsymbol{i} \in \mathcal{I}$, where $\mathcal{L}$ is a set of natural language strings/prompts and $\mathcal{I} \subseteq \mathcal{O}$ is the space of 2D RGB images.

2.2.1 CLIP

CLIP (Contrastive Language Image Pretraining; Radford et al. (2021)) is a family of vision-language models that have two main components – an image encoder $\psi_I : \mathcal{I} \mapsto \mathcal{V}$ and a language encoder $\psi_L : \mathcal{L} \mapsto \mathcal{V}$, which generate unit magnitude vector embeddings for images and text, respectively, in the same high-dimensional latent space $\mathcal{V} \in \mathbb{R}^{n_v}$, which can thus be compared to each other using a vector similarity metric $\zeta : \mathcal{I} \times \mathcal{L} \mapsto \mathbb{R}$, e.g. cosine similarity (S_c, essentially dot product).

For a label $\mathbf{l} \in \mathcal{L}$ and an image $\mathbf{i} \in \mathcal{I}$, the similarity is given by,

$$\zeta(\mathbf{i}, \mathbf{l}) = S_c(\psi_I(\mathbf{i}), \psi_L(\mathbf{l})) = \frac{\psi_I(\mathbf{i}) \cdot \psi_L(\mathbf{l})}{\|\psi_I(\mathbf{i})\| \|\psi_L(\mathbf{l})\|} = \psi_I(\mathbf{i}) \cdot \psi_L(\mathbf{l}). \tag{2.3}$$

Eponymously, the CLIP image and text encoders are jointly trained using a contrastive loss which directs the model to semantically align the correct image and text pairs (in the latent space) by increasing their similarity while simultaneously pushing apart the embeddings of the mismatched pairs by decreasing their similarity.

Given a batch of N image-caption pairs; images $\boldsymbol{i}^{(N)} = \{\mathbf{i}_1, \ldots, \mathbf{i}_N\}$ and captions $\boldsymbol{l}^{(N)} = \{\mathbf{l}_1, \ldots, \mathbf{l}_N\}$, such that the corresponding entries are paired, CLIP first computes the matrix of cosine similarities, $\zeta^{(N)}(\boldsymbol{i}, \boldsymbol{l}) \in \mathbb{R}^{N \times N}$, where each item $\zeta^{(N)}_{j,k}$ computes the cosine similarity between the image $\mathbf{i}_j$ and caption $\mathbf{l}_k$. The training batch loss is then computed as the difference between the similarities corresponding to the correct image-caption pairs and the mismatched caption-image pairs, which is then minimized.

There are several variants of CLIP, with different sizes and different architectures for its image encoder (vision transformer *ViT*; Dosovitskiy et al. (2020), or residual network *ResNet*; He et al. (2016)). We used the vision transformer variant (`ViT-L/14`) for most of our experiments as it gave a good balance of performance and computational efficiency. It takes image inputs of size 224×224 and tokenizes them in patch sizes of 14×14 for the vision transformer. The comparison of the different models from the original paper for our use case is presented in the appendix Section D.

CLIP as Zero-Shot Image Classifier

CLIP can be used as a zero-shot classifier by comparing the embeddings of a given image and a set of prompts to predict the likelihood of the image belonging to all of the classes.

Given an image $\mathbf{i}$ and a set of text prompts $\boldsymbol{l}$, the likelihood of the image belonging to the class described by a prompt $\mathbf{l}_k \in \boldsymbol{l}$ is given by the softmax function over the similarity scores of the image and the prompt embeddings,

$$p(\mathbf{l}_k; \mathbf{i}, \boldsymbol{l}, T) = \frac{\exp(\zeta(\mathbf{i}, \mathbf{l}_k)/T)}{\sum_{\mathbf{l}_j \in \boldsymbol{l}} \exp(\zeta(\mathbf{i}, \mathbf{l}_j)/T)}, \tag{2.4}$$

where $T \in \mathbb{R}^+$ is the temperature hyperparameter that controls the flatness of the distribution. This is an important parameter as it directly controls the quality of our entropy reward. We usually use the temperature 0.01 as suggested in the original paper for our experiments, which we also found to work much better than other values. See Section 3.5.2 for details.

Prompts for CLIP

Ideally, CLIP should abstract away frivolous phrasing to focus on the semantics of a prompt, but in practice, due to inherent ambiguity in natural language, the exact phrasing can still influence its inference (Roth et al., 2023). We experimented with this in our investigations.

For objective analysis, we break the structure of a language prompt $\mathbf{c} \in \mathcal{L}$ representing a creative possibility c in an environment as "{prefix}{c}{suffix}", where "prefix" and "suffix" are optional context phrases. For example, if $c =$ "apple", then the prompt could be "picture of an apple, on a white background", where "picture of an " is the prefix and ", on a white background" is the suffix.

2.3 Semantic Rewards for Free Play

This section describes the mathematical formulation of the intrinsic semantic reward we developed to encourage the agent to explore its environment in a semantically expressive manner, akin to free play in humans. It also gives the necessary background on the additional intrinsic regularity reward that complements it.

2.3.1 Semantics Entropy Reward

Mathematically, in its simplest form, the semantics reward $r_{\text{semantics-entropy}} : \mathcal{O} \mapsto \mathbb{R}$ is formulated as the negative entropy of the likelihood of the predictions of the VLM for an image observation $\mathbf{o} \in \mathcal{O}$ over a set of n_c creative possibilities $\boldsymbol{c} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{n_c}\}$; We call this the *semantics-entropy reward*,

$$r_{\text{semantics-entropy}}(\mathbf{o}; \boldsymbol{c}, T) := -\mathcal{H}(p(\boldsymbol{c} \mid \mathbf{o}; T)) = \sum_{\mathbf{c}_k \in \boldsymbol{c}} p(\mathbf{c}_k \mid \mathbf{o}; \boldsymbol{c}, T) \log p(\mathbf{c}_k \mid \mathbf{o}; \boldsymbol{c}, T). \tag{2.5}$$

Goal-Baseline Regularization

To improve the reward quality, we further extend this formulation by adopting a combination of suggestions from other studies on VLMs as a source of goal-conditioned rewards.

The most trivial way to use CLIP as a goal-conditioned reward is to directly use the similarity metric from eq. (2.3) as the reward function. Given a goal description $\mathbf{g} \in \mathcal{L}$, the reward for an image observation $\mathbf{o} \in \mathcal{O}$ is given by,

$$r_{\text{clip}}(\mathbf{o}; \mathbf{g}) = S_c(\psi_I(\mathbf{o}), \psi_L(\mathbf{g})). \tag{2.6}$$

This captures the projection of the image observation embedding $\mathbf{o}$ onto the direction spanned by the goal embedding $\mathbf{g}$.

Cui et al. (2022) developed an alternative method called *ZeST* (zero-shot task specification) which employs *delta embeddings* that use the difference in embeddings between the desired and initial/*baseline* configurations, for both image and text inputs, to directionally remove irrelevant parts of the CLIP representation. The goal-baseline regularized reward function in this case is the similarity between the delta embeddings,

$$r_{\text{zest}}(\mathbf{o}; \mathbf{o}_b, \mathbf{g}, \mathbf{g}_b) = S_c((\psi_I(\mathbf{o}) - \psi_I(\mathbf{o}_b)), (\psi_L(\mathbf{g}) - \psi_L(\mathbf{g}_b))). \tag{2.7}$$

where $\mathbf{g}_b \in \mathcal{L}$ is a natural language description of the initial/baseline environment image observation $\mathbf{o}_b = \mathbf{o}_0$.

Intuitively, proving a baseline sets the context of the setup, and the direction from a baseline $\mathbf{g}_b$ to goal $\mathbf{g}$ captures the desired change from the environment's image baseline $\mathbf{o}_b$ to the target state. As an example, consider the case where the goal is to draw a circle starting from a blank canvas i.e. the goal $\mathbf{g}$ is something like "a circle on a white background". Then the baseline $\mathbf{g}_b$ could be "a blank canvas" and the image observation baseline $\mathbf{o}_b$ would be the starting blank canvas image.

Baumli et al. (2023) further proposed a negative prompt formulation that used the negations of goal states, instead of descriptions of the initial states, as goal baselines. In the example, this would be "not a circle on a white background".

These delta embeddings considerably improved performance in language-guided goal-conditioned tasks by providing a more focused reward signal. However, it is not certain that the direction from the baseline to the goal captures all the relevant information. Therefore, instead of using one or the other, Rocamonde et al. (2023) further made a trade-off between the two projections in their *VLM-RM* method and proposed a goal-baseline regularization technique that considered both – the similarity of the image observation to the target and also the direction from the baseline towards the goal in the CLIP embedding space.

VLM-RM does not use delta embeddings or baselines for the image input, but only for the text input. Their formulation is given by,

$$r_{\text{goal-reg}}(\mathbf{o}; \mathbf{g}, \mathbf{g}_b, \gamma) = 1 - \frac{1}{2} \left\| \gamma \, \text{proj}_V \psi_I(\mathbf{o}) + (1-\gamma) \, \psi_I(\mathbf{o}) - \psi_L(\mathbf{g}) \right\|_2^2, \tag{2.8}$$

where $V := \frac{\psi_I(\mathbf{g}) - \psi_L(\mathbf{g}_b)}{\|\psi_I(\mathbf{g}) - \psi_L(\mathbf{g}_b)\|}$ is the direction spanned by the vector between the embeddings of $\mathbf{g}$ and $\mathbf{g}_b$, $\text{proj}_V \psi_I(\mathbf{o}) := (\mathbf{o} \cdot V)\, V$ is the projection of the image observation embedding $\mathbf{o}$ onto this direction, and $\gamma \in [0, 1]$ is a hyperparameter to control the trade-off/regularization strength. Note that the magnitude of the projection $\|\text{proj}_V \psi_I(\mathbf{o})\|_2$ is equal to the cosine similarity $\cos(\vartheta) := S_c(\psi_I(\mathbf{o}), (\psi_L(\mathbf{g}) - \psi_L(\mathbf{g}_b)))$, and $\sin(\vartheta)$ is the projection of $\psi_I(\mathbf{o})$ perpendicular to this direction. The authors found this method to be relatively robust to changes in γ with most intermediate values being better than 0 or 1.

For $\gamma = 0$, the original CLIP similarity metric from eq. (2.3) is recovered as an unregularized goal-conditioned reward function,

$$r_{\text{goal-reg}}(\mathbf{o}; \mathbf{g}, \mathbf{g}_b, 0) = 1 - \frac{1}{2} \left\| \psi_I(\mathbf{o}) - \psi_L(\mathbf{g}) \right\|_2^2 = S_c(\psi_I(\mathbf{o}), \psi_L(\mathbf{g})), \tag{2.9}$$

since $\|\psi_I(\mathbf{g}_b)\|_2 = \|\psi_L(\mathbf{g})\|_2 = 1$. For $\gamma = 1$, there is a trade-off of components of $\mathbf{o}$ orthogonal and parallel to "$\mathbf{g} - \mathbf{g}_b$".

$$\begin{aligned} r_{\text{goal-reg}}(\mathbf{o}; \mathbf{g}, \mathbf{g}_b, 1) &= 1 - \frac{1}{2} \left\| \text{proj}_V \psi_I(\mathbf{o}) - \psi_L(\mathbf{g}) \right\|_2^2 && (2.10) \\ &= 1 - \frac{\|\text{proj}_V \psi_I(\mathbf{o})\|_2^2 + 1 - 2 \, \text{proj}_V \psi_I(\mathbf{o}) \cdot \psi_L(\mathbf{g})}{2} && (2.11) \\ &= \underbrace{\frac{1 - \|\text{proj}_V \psi_I(\mathbf{o})\|_2^2}{2}}_{\propto \sin^2(\vartheta)} + \underbrace{\frac{\psi_I(\mathbf{o}) \cdot (\psi_L(\mathbf{g}) - \psi_L(\mathbf{g}_b))}{2}}_{\propto \cos(\vartheta)}. && (2.12) \end{aligned}$$

Regularized Semantics Entropy Reward

We combine these ideas to derive a *baseline regularized semantics-entropy reward*. Let $\boldsymbol{p} = \text{proj}_V \psi_I(\mathbf{o})$, $\boldsymbol{o} = \psi_I(\mathbf{o})$, $\boldsymbol{g} = \psi_L(\mathbf{g})$, $\boldsymbol{g}_b = \psi_L(\mathbf{g}_b)$ for brevity, then using the property $\|\boldsymbol{o}\|_2 = \|\boldsymbol{g}\|_2 = \|\boldsymbol{g}_b\|_2 = 1$ and the results $\boldsymbol{p} \cdot \boldsymbol{o} = \|p\|_2^2$ and $\boldsymbol{p} \cdot \boldsymbol{g} = (\boldsymbol{g} - \boldsymbol{g}_b) \cdot \boldsymbol{o} \;/\; 2$, the VLM-RM reward can be simplified as,

$$
\begin{aligned}
& 1 - \frac{1}{2}\|\gamma\, \boldsymbol{p} + (1-\gamma)\, \boldsymbol{o} - \boldsymbol{g}\|_2^2 && (r_{\text{goal-reg}}, \text{from 2.8}) \\
&= 1 - \frac{1}{2}\Big[\|\gamma\, \boldsymbol{p} + (1-\gamma)\, \boldsymbol{o}\|_2^2 + \cancelto{1}{\|\boldsymbol{g}\|_2^2} && - 2\,(\gamma\, \boldsymbol{p} + (1-\gamma)\, \boldsymbol{o}) \cdot \boldsymbol{g}\Big] && (2.13) \\
&= \frac{1 - \|\gamma\, \boldsymbol{p} + (1-\gamma)\, \boldsymbol{o}\|_2^2}{2} && + \gamma\, \boldsymbol{p} \cdot \boldsymbol{g} + (1-\gamma)\, \boldsymbol{o} \cdot \boldsymbol{g} && (2.14) \\
&= \frac{1 - \|\gamma\, (\boldsymbol{p} - \boldsymbol{o}) + \boldsymbol{o}\|_2^2}{2} && + \frac{\gamma}{2}\, (\boldsymbol{g} - \boldsymbol{g}_b) \cdot \boldsymbol{o} + (1-\gamma)\, \boldsymbol{g} \cdot \boldsymbol{o} && (2.15) \\
&= \frac{1 - \gamma^2\, \|\boldsymbol{p} - \boldsymbol{o}\|_2^2 - 2\gamma\, (\boldsymbol{p} - \boldsymbol{o}) \cdot \boldsymbol{o} - \cancelto{1}{\|\boldsymbol{o}\|_2^2}}{2} && + \Big(\frac{\gamma}{2}\, (\boldsymbol{g} - \boldsymbol{g}_b) + (1-\gamma)\, \boldsymbol{g}\Big) \cdot \boldsymbol{o} && (2.16) \\
&= \frac{-\gamma^2\, \big(\|\boldsymbol{p}\|_2^2 + \|\boldsymbol{o}\|_2^2 - 2\, \boldsymbol{p} \cdot \boldsymbol{o}\big) - 2\gamma\, \big(\boldsymbol{p} \cdot \boldsymbol{o} - \|\boldsymbol{o}\|_2^2\big)}{2} && + \Big(\frac{\gamma}{2}\, \boldsymbol{g} - \frac{\gamma}{2}\, \boldsymbol{g}_b + \boldsymbol{g} - \gamma\, \boldsymbol{g}\Big) \cdot \boldsymbol{o} && (2.17) \\
&= \frac{-\gamma^2\, \big(\|\boldsymbol{p}\|_2^2 + 1 - 2\, \|\boldsymbol{p}\|_2^2\big) - 2\gamma\, \big(\|\boldsymbol{p}\|_2^2 - 1\big)}{2} && + \Big(\boldsymbol{g} - \frac{\gamma}{2}\, \boldsymbol{g} - \frac{\gamma}{2}\, \boldsymbol{g}_b\Big) \cdot \boldsymbol{o} && (2.18) \\
&= \frac{-\gamma^2 + 2\gamma}{2} + \frac{\gamma^2\, \|\boldsymbol{p}\|_2^2 - 2\gamma\, \|\boldsymbol{p}\|_2^2}{2} && + \Big(\Big(1 - \frac{\gamma}{2}\Big)\, \boldsymbol{g} - \frac{\gamma}{2}\, \boldsymbol{g}_b\Big) \cdot \boldsymbol{o} && (2.19) \\
&= \gamma\, \Big(1 - \frac{\gamma}{2}\Big)\, \big(1 - \|\boldsymbol{p}\|_2^2\big) && + \Big(\Big(1 - \frac{\gamma}{2}\Big)\, \boldsymbol{g} - \frac{\gamma}{2}\, \boldsymbol{g}_b\Big) \cdot \boldsymbol{o} && (2.20) \\
&= \underbrace{\gamma\, \Big(1 - \frac{\gamma}{2}\Big)\, (\sin(\vartheta))}_{\text{Term \#1 } (\perp)} && + \underbrace{\Big(\Big(1 - \frac{\gamma}{2}\Big)\, \boldsymbol{g} - \frac{\gamma}{2}\, \boldsymbol{g}_b\Big) \cdot \boldsymbol{o}}_{\text{Term \#2 } (\parallel)}. && (2.21)
\end{aligned}
$$

Term #1 captures the projection perpendicular to the direction from the baseline to the goal and Term #2 captures the trade-off of projection along this direction and the goal. To simplify, we consider only the latter in our formulation as it contains the desired information about the goal. We adopt this term as the basis to add baseline regularization to the semantics-entropy reward and consider similar delta embeddings for the image observation space as well.

First, let us formulate a regularized similarity metric for an image observation $\mathbf{i} \in \mathcal{I}$ with baseline image $\mathbf{i}_b \in \mathcal{I}$ and target label $\mathbf{l} \in \mathcal{L}$ with baseline $\mathbf{l}_b \in \mathcal{L}$;

$$\hat{\zeta}(\mathbf{i}, \mathbf{l}; \mathbf{i}_b, \mathbf{l}_b, \alpha, \beta) = S_c(((1-\beta)\, \psi_I(\mathbf{i}) - \beta\, \psi_I(\mathbf{i}_b)), ((1-\alpha)\, \psi_I(\mathbf{l}) - \alpha\, \psi_I(\mathbf{l}_b))) \tag{2.22}$$

where $\alpha, \beta \in [0, 0.5]$ are the regularization hyperparameters homologous to $\gamma/2$ in eq. (2.21), for text and image inputs respectively, and $T \in \mathbb{R}^+$ is the temperature hyperparameter.

The likelihood distribution $\hat{p}$ of the predictions of the VLM for the image observation $\mathbf{o}$ over the creative possibilities $\boldsymbol{c}$. For $\mathbf{c}_k \in \boldsymbol{c}$ is then given by,

$$\hat{p}(\mathbf{c}_k \mid \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha, \beta, T) = \frac{\exp(\hat{\zeta}(\mathbf{o}, \mathbf{c}_k; \mathbf{o}_b, \mathbf{c}_b, \alpha, \beta)/T)}{\sum_{\mathbf{c}_j \in \boldsymbol{c}} \exp(\hat{\zeta}(\mathbf{o}, \mathbf{c}_k; \mathbf{o}_b, \mathbf{c}_b, \alpha, \beta)/T)}, \tag{2.23}$$

The regularized semantics-entropy reward is thus defined as,

$$
\begin{aligned}
r_{\text{semantics-entropy-reg}}(\mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha, \beta, T) &:= -\mathcal{H}(\hat{p}(\boldsymbol{c} \mid \mathbf{o}; \mathbf{o}_b, \mathbf{c}_b, \alpha, \beta, T)) && (2.24) \\
&= \sum_{\mathbf{c}_k \in \boldsymbol{c}} \hat{p}(\mathbf{c}_k \mid \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha, \beta, T) \log \hat{p}(\mathbf{c}_k \mid \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha, \beta, T).
\end{aligned}
$$

The text baseline could either be the initial state description or the negation of the goal.

2.3.2 Regularity Reward

In addition to the entropy reward, the semantic reward also consists of regularity as an intrinsic reward (Sancaktar et al., 2023), which encourages the agent to create regular and symmetric patterns in its creations, which could promote semantic expression. The authors described the regularity reward ***RaIR*** as "(the) redundancy in the description of the situation, to measure the degree of sub-structure recurrence".

Mathematically, it is considered that the state space can be factorized into a composition of the different entities in the environment and the configuration of the agent, $\mathcal{S} = (\mathcal{S}_{\text{obj}})^N \times \mathcal{S}_{\text{agent}}$, where $\mathcal{S}_{\text{obj}} \in \mathbb{R}^d$, such that $n_s = N \cdot d$. This allows the***RaIR*** reward $r_{\text{RaIR}} : \mathcal{S} \mapsto \mathbb{R}$ to be formulated in terms of multiplicities $m : \mathcal{X} \mapsto \mathbb{N}$ (counts of occurrence) of the sub-structures $\mathcal{X}$ in the arrangement of the different entities of the environment in the state $\mathbf{s}$,

$$r_{\text{RaIR}}(\mathbf{s}; h) := -\mathcal{H}(\phi(\mathbf{s}; h)) = \sum_{\mathcal{X} \in \phi(\mathbf{s})} p(\mathcal{X}; h) \log p(\mathcal{X}; h), \tag{2.25}$$

where ϕ is a factorization function that decomposes a state $\mathbf{s}$ into a multiset of its unique sub-structures given the hyperparameters h (resolution, bidirectionality),

$$\phi : (\mathcal{S}_{obj})^N \mapsto \{\mathcal{X}_1^{m(\mathcal{X}_1)}, \mathcal{X}_2^{m(\mathcal{X}_2)}, \ldots\}, \tag{2.26}$$

and $p(\mathcal{X}; h)$ is the probability of a sub-structure $\mathcal{X} \in \phi(\mathbf{s}; h)$ in the state $\mathbf{s}$ calculated as,

$$p(\mathcal{X}; h) = \frac{m(\mathcal{X})}{\sum_{\mathcal{X}' \in \phi(\mathbf{s}; h)} m(\mathcal{X}')}. \tag{2.27}$$

For our purposes, following insights from Zingrone (2014) and Golomb (1987), we use the relational formulation of ***RaIR*** which captures the pairwise relationships between the entities of the environment in the state $\mathbf{s}$ by calculating the distances between them in the discretized coordinate space;

$$\phi(\mathbf{s}) = \{\mathcal{X}^{m(\mathcal{X})} : \mathcal{X} = \lfloor s^{(j)} - s^{(k)} \rceil \mid s^{(j)}, s^{(k)} \in \mathbf{s}\}, \tag{2.28}$$

where the $\lfloor \cdot \rceil$ operator groups the distances into discrete bins based on the configured resolution. For more details on the parameters and formulation, please refer to Section C.1.2.

2.3.3 Complete Intrinsic Semantics Reward

The complete intrinsic semantics reward $r_{\text{semantics}}$ is then given as the weighted sum of the regularity reward r_{RaIR} and the regularized semantics-entropy reward $r_{\text{semantics-entropy-reg}}$;

$$r_{\text{semantics}}(\mathbf{s}, \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha, \beta, T) = r_{\text{semantics-entropy-reg}}(\mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha, \beta, T) + \lambda\, r_{\text{RaIR}}(\mathbf{s}), \tag{2.29}$$

where λ is a hyperparameter that controls the trade-off between the two rewards.

To be able to properly compare the two rewards, we normalize the regularity reward by the maximum possible entropy of the multiset $\log\left(\binom{N}{2}\right)$, which is equal to the entropy of a uniform distribution over the $\binom{N}{2}$ possible pairwise distances in the state space. This constrains it to the range $[0, 1]$.

Similarly, the semantics-entropy reward is normalized by the maximum possible entropy of the likelihood distribution over the creative possibilities, which is $\log(n_c)$, where n_c is the number of creative possibilities.

2.4 iCEM

In the Model-Based Reinforcement Learning (MBRL) paradigm, the agent learns a model f of the MDP dynamics from interactions with the environment and uses it to predict the consequences of its actions which enables it to plan for the best policies. Such policies can be learned with universal function approximators such as neural networks using RL algorithms or planning methods.

The improved Cross-Entropy Method (iCEM) is one such planning method. It is a zeroth-order Monte-Carlo trajectory optimizer that works in a Model Predictive Control (MPC) loop to solve finite-horizon planning problems, i.e. it iteratively re-plans after every step in the environment by (1) coming up with several action sequences (by sampling a multi-variate distribution) to *think* a few steps ahead, (2) imagining/predicting the future states resulting from these actions (using the model of the environment) and assessing their expected returns (using the reward function), (3) picking out the best action sequences, (4) improving the *thinking* method based on these elite sequences, and (5) taking the first step of the best action sequence.

Precisely, given a finite planning horizon h, iCEM uses a clipped colored-noise Gaussian distribution of dimension $n_a \times h$ over action sequences ($\boldsymbol{a}^{(h)} = \{\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{h-1}\}$), given by,

$$p(\boldsymbol{a}^{(h)}) = \text{clip}(\boldsymbol{\mu}_t + \mathcal{C}^{\beta}(n_a, h) \odot \boldsymbol{\sigma}_t^2), \tag{2.30}$$

where $\boldsymbol{\mu}_{t,\boldsymbol{\sigma}_t} \in \mathbb{R}^{n_a \times h}$ are the mean and standard deviation of the sampling distribution at time t respectively, and $\mathcal{C}^{\beta}(n_a, h)$ is a sampling function that returns n_a (one for each action dimension) sequences of length h sampled from colored noise normal distribution with exponent β ($= 1$ in our experiments; pink noise), zero mean, and unit variance.

At every timestep t, there are multiple *inner* iterations of the following calculations. At inner iteration i, (1) starting with a sampling distribution $p(\boldsymbol{\mu}_t^{(i-1)}, \boldsymbol{\sigma}_t^{(i-1)})$, a fixed-length ($n$) set of action sequences ($\mathcal{A}_t = \{\boldsymbol{a}_1^{(h)}, \boldsymbol{a}_2^{(h)}, \ldots, \boldsymbol{a}_n^{(h)}\}$) is sampled from this distribution. These samples are augmented with a fraction of the best action sequences (called the *elite-set* $\mathcal{E}_t^{(i-1)}$) from the previous inner iteration (called *keep-elites*) or the previous timestep (called *shift-elites*) if $i = 0$. Additionally, to ensure that the mean action of the distribution is accessible to the controller, in the last inner iteration m, the mean of the distribution $\boldsymbol{\mu}_t$ is added to the elite-set $\mathcal{A}^+$;

$$\mathcal{A}_t^+ = \mathcal{A}_t \cup \begin{cases} \mathcal{E}_{t-1}^{(m)} & \text{if } i = 0, \\ \mathcal{E}_t^{(m-1)} \cup \{\boldsymbol{\mu}_t^{(m)}\} & \text{if } i = m, \\ \mathcal{E}_t^{(i-1)} & \text{otherwise.} \end{cases} \tag{2.31}$$

(2) Every action sequence in this augmented set $\mathcal{A}_t^+$ is then rolled out (*in imagination*) using the environment model f and rendering function θ from the current state $\mathbf{s}_t$ to predict their corresponding expected states and image observations respectively,

$$\mathcal{S}_t = \{\{\mathbf{s}_{t+1}, \mathbf{s}_{t+2}, \ldots, \mathbf{s}_{t+h}\}\}_{j=1}^{n}, \tag{2.32}$$

$$\mathcal{O}_t = \{\{\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \ldots, \mathbf{o}_{t+h}\}\}_{j=1}^{n}, \tag{2.33}$$

where $\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{a}_{t-1})$ and $\mathbf{o}_t = \theta(\mathbf{s}_t)$. The predicted states are evaluated according to the reward function r to get their corresponding rewards,

$$\mathcal{R}_t = \{\{R_{t+1}, R_{t+2}, \ldots, R_{t+h}\}\}_{j=1}^{n}, \tag{2.34}$$

where $R_t = r(\mathbf{s}_t, \mathbf{o}_t) = \sum_k \lambda_k r_k(\mathbf{s}_t, \mathbf{o}_t)$ is the weighted sum of all the rewards. The corresponding trajectories are,

$$\mathcal{T}_t^{(n \times h)} = \{\{(\mathbf{s}_t, \mathbf{o}_t, R_t, \mathbf{a}_t), \ldots, (\mathbf{s}_{t+h-1}, \mathbf{o}_{t+h-1}, R_{t+h-1}, \mathbf{a}_{t+h-1}), (\mathbf{s}_{t+h}, \mathbf{o}_{t+h}, R_{t+h}, \varnothing)\}\}_{j=1}^{n}. \tag{2.35}$$

(3) Subsequently, a new elite-set of actions $\mathcal{E}_t^{(i)}$ of size K is determined; it consists of the best K action sequence samples that maximize an aggregation of the returns from their expected states according to a reward aggregation function g;

$$\mathcal{E}_t^{(i)} = \{\boldsymbol{a}_k^{(h)} \in \mathcal{A}_t^+ : | \; \boldsymbol{a}_j^{(h)} \in \mathcal{A}_t^+ : g(\boldsymbol{\tau}_k^{(h)}) \leq g(\boldsymbol{\tau}_j^{(h)}) \; | \leq K\}. \tag{2.36}$$

Best K action sequences in the augmented set of samples $\mathcal{A}_t^+$

More information about the reward aggregation strategies used in our experiments is provided in subsection 2.4.1.

(4) These new elite action sequences are finally used to refine the distribution to maximize the expected return of its samples.

$$\boldsymbol{\mu}_t^{(i)} = \eta \; \boldsymbol{\mu}_t^{(i-1)} + (1-\eta)\boldsymbol{\mu}^{\mathcal{E}_t^{(i)}}, \tag{2.37}$$

$$(\boldsymbol{\sigma}_t^{(i)})^2 = \eta(\boldsymbol{\sigma}_t^{(i-1)})^2 + (1-\eta)(\boldsymbol{\sigma}^{\mathcal{E}_t^{(i)}})^2, \tag{2.38}$$

where $\eta \in [0, 1]$ is a *momentum* factor.

The above steps are repeated for m inner iterations at every timestep. At the end of this process, the distribution concentrates on action sequences with high returns, and the best action sequence converges to an optimum;

$$\boldsymbol{a}^{(h)*} = \underset{\boldsymbol{a}_k^{(h)} \in \mathcal{A}^+}{\operatorname{argmax}} \; g(\boldsymbol{\tau}_k^{(h)}). \tag{2.39}$$

(5) Finally, the first action of the best elite sequence of actions $\boldsymbol{a}_t^{(h)*}$ is executed in the environment. The process is repeated for a specified number of rollout steps or until a terminal state or reward/cost threshold is reached.

Since iCEM is a zero-order trajectory optimizer with a limited sample budget and finite-horizon planning, we do not necessarily converge to the global optima. Although this does not solve the full reinforcement learning problem (infinite horizon and stochastic environments), it is very powerful in optimizing for tasks ad-lib without further adaptation, which makes it suitable for optimizing changing exploration targets, as in our case (because of noisy CLIP rewards; discussed later in Section 3.3.2).

The colored noise of the iCEM distribution leads to temporally correlated action sequences over the horizon for smoother trajectories, which helps mitigate the problems due to the sparse nature of our semantics entropy reward. Furthermore, the addition of memory gives it good sample efficiency, which was essential for our case due to the computational bottleneck because of the long inference times for CLIP. These reasons made iCEM a suitable choice for our experiments.

For this to be successful, a good transition model f needs to be learned, and the reward function needs to be known or discovered. In our case, the reward function is well-defined, and to evaluate the efficacy of our entropy minimization objective in achieving good semantic expression, we do planning using ground truth (GT) models, i.e. with access to the true simulator of the environment itself for planning. Thus, we can perform multi-horizon planning without accumulating any model errors, which allows us to better investigate the global/local optima of our semantics reward.

2.4.1 Reward Aggregation Strategies

There are many different ways to aggregate the rewards of the trajectory samples over the horizon. This is treated as one of the hyperparameters in our experiments.

A trivial reward aggregation function is the sum of the rewards over the horizon (`sum`),

$$\texttt{sum}(\boldsymbol{\tau}^{(h)}) = \sum_{k=1}^{h} R_k. \tag{2.40}$$

We further experimented with a discount factor $\gamma \in \mathbb{R}^+$ over the horizon, but usually set it to 1.0;

$$\texttt{sum-}\gamma(\boldsymbol{\tau}^{(h)}) = \sum_{k=1}^{h} \gamma^{k-1} R_k. \tag{2.41}$$

However, this type of consolidating aggregation strategy is not suitable for cases where an increase in return can be preceded by an initial decrease (as a consequence of leaving local optima). So we also considered the `best` method, which uses the best return over the planning horizon;

$$\texttt{best}(\boldsymbol{\tau}^{(h)}) = \max_{k=1}^{h} R_k. \tag{2.42}$$

Additionally, to combine the better qualities of both the `sum` and `best` methods; we formulated a dynamic reward aggregation method that ranks trajectories based on the highest sum of consecutive rewards in a window of a predefined hyperparamterized length l, `best-`l;

$$\texttt{best-}l(\boldsymbol{\tau}^{(h)}) = \max_{k=1}^{h-l+1} \sum_{j=k}^{k+l-1} R_{k+j}. \tag{2.43}$$

Another strategy is to assess the trajectories based on their last states (`last`), i.e. to simply consider only what the agent could potentially achieve at the end of the horizon. This is particularly useful when inferring the reward is expensive, as in our case, because only one final state needs to be evaluated;

$$\texttt{last}(\boldsymbol{\tau}^{(h)}) = R_h. \tag{2.44}$$

Finally, we also experimented with a combination of the `sum` and `last` methods with `last-`l, which is the sum of the returns of the last l steps of the trajectory;

$$\texttt{last-}l(\boldsymbol{\tau}^{(h)}) = \sum_{k=h-l+1}^{h} R_k. \tag{2.45}$$

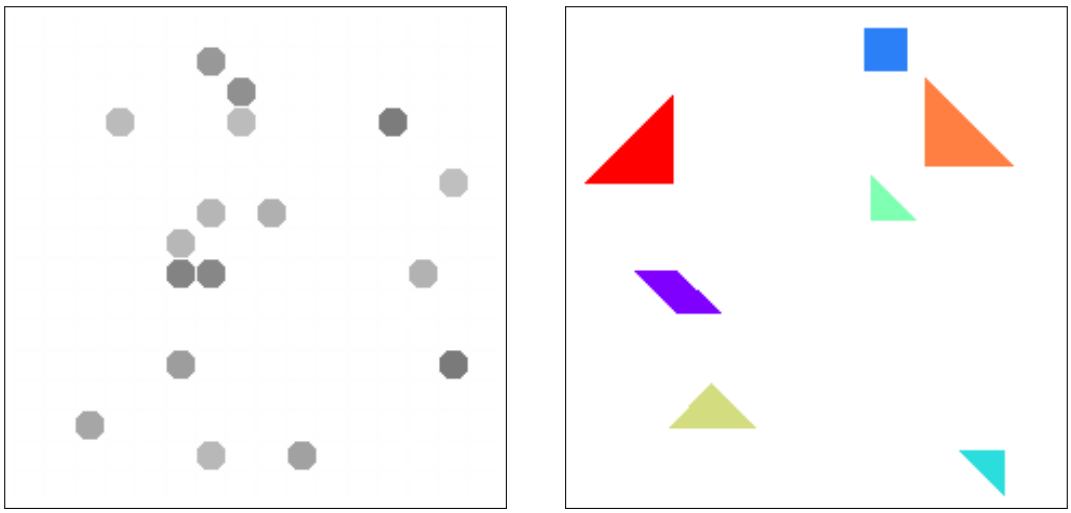
Chapter 3

Experiments

We perform these experiments to showcase that we can indeed get semantically expressive creations with our proposed formulation.

3.1 Environments

We use two custom environments in our experiments which allow for rich creative expression.



(a) ShapeGridWorld (b) Tangram

Figure 3.1: The environments.

3.1.1 ShapeGridWorld

ShapeGridWorld is a pixel grid environment where the agent can draw on the grid by moving pixels around, either one or all of them at the same time. This is reminiscent of the pin-board platform used in the free play study conducted by Diggs-Galligan et al. (2021) (Fig. 1.1). By setting the pixels in a specific layout, the agent can draw a variety of shapes on the grid. This environment was adapted from the pixel grid environment used in the work by Sancaktar et al. (2023) with some modifications summarised in appendix A.1.

ShapeGridWorld was mostly used in the initial experiments but it eventually proved to promote noise in CLIP inferences which led to underwhelming results. To work around the problems, the Tangram environment was developed.

3.1.2 Tangram

Tangram is a traditional puzzle game consisting of a canvas and seven geometric shapes – five right isosceles triangles (two large, one medium, and two small), one square, and one parallelogram, which conventionally have distinct colors. These shapes can be rotated, flipped, and translated on the canvas, without overlapping. Even though this is a very simple setup, these pieces can be arranged in very different configurations to create expressive abstract patterns.

The Tangram virtual environment was developed specifically for our experiments as it mitigated the problems discussed in sections Section 3.3.1 and Section 3.3.2, and is a major contribution of our work.

We mostly configured the environment to correspond with the traditional Tangram game. The shapes involved were the same, no overlap was allowed, and the shapes could be rotated, flipped, and translated. Yet, we did experiments with colors to improve the performance of CLIP and also gave the agent the freedom to use a subset of the shapes to make its creations. Please refer to appendix A.2 for more technical details about this environment.

3.2 Initial Tests with CLIP

CLIP was trained on a large dataset of captioned natural images foraged from the internet. While it generalizes well to many natural-image distributions and has proved to be a powerful zero-shot model for various tasks, it was not clear how well it would perform on sketches and abstract images.

CLIP has been shown to not perform much better than chance on many tasks which were not well represented in its pre-training dataset. The authors of CLIP also noted that it is particularly poor on tasks for fine-grained classification such as differentiating models of cars, species of flowers, and variants of aircraft. It also struggles with more abstract and systematic tasks such as counting the number of objects in an image.

Moreover, Rocamonde et al. (2023), from their experiments with CLIP as a source of goal-conditioned rewards, reported that CLIP rewards are only meaningful and well-shaped for environments that are photorealistic enough for the CLIP visual encoder to interpret correctly. They found that it is crucial to add textures and shading to the images to make them more realistic for CLIP to perform well.

Influenced by these observations and caveats, we suspected that our environments, since they are not photorealistic, might also be out-of-distribution. Yet, we expected them to be simple enough for CLIP to interpret correctly.

Thus, before testing our reward function, we conducted a series of simple experiments on the inference capabilities of CLIP on ShapeGridWorld and Tangram. At the same time, there were several hyperparameters in the environments and CLIP into which we took insights with these initial tests.

3.2.1 CLIP on Sketches and ShapeGridWorld

We first tested CLIP on sketches of simple shapes from the ImageNet-Sketch dataset (Wang et al., 2019).

We also compared these results with renderings of these sketches on ShapeGridWorld grids of different resolutions by registering them to a grid (see Section A.1.1 for more details).

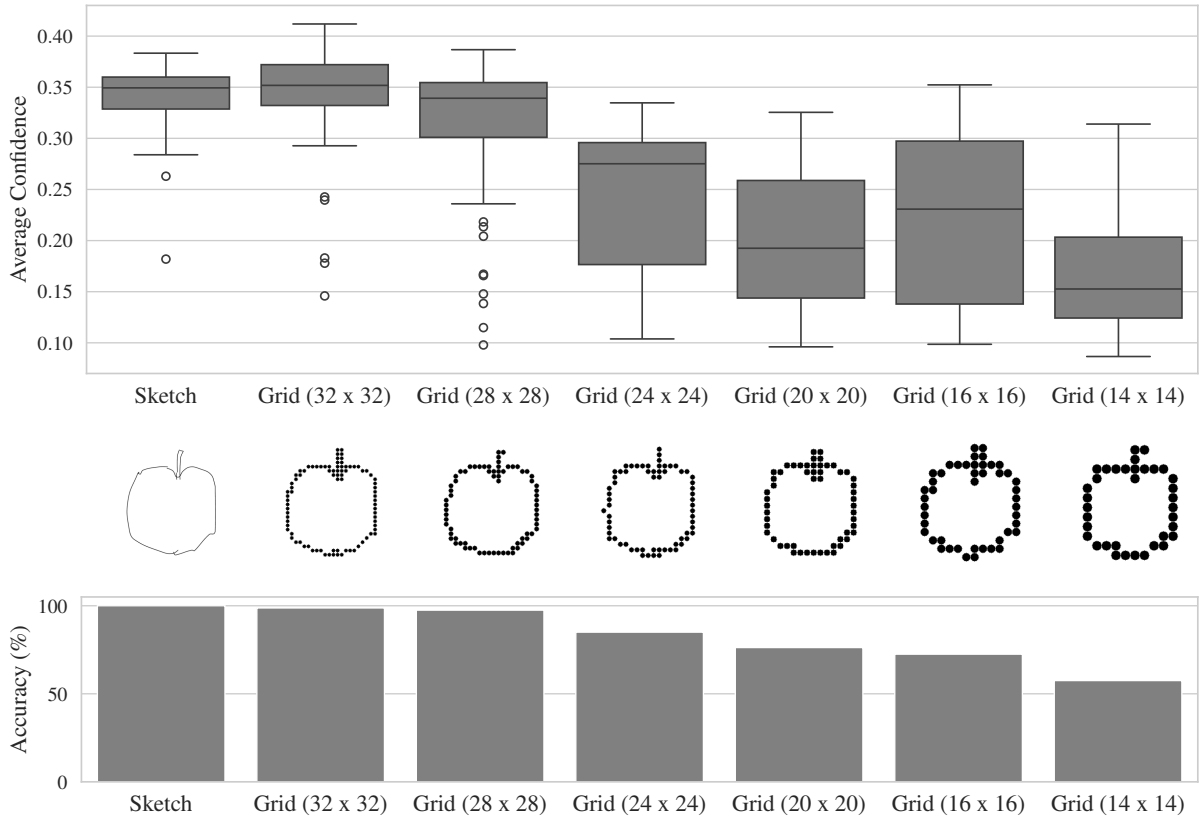


Figure 3.2: CLIP on sketches and ShapeGridWorld grids of different resolutions.

As an example, Fig. 3.2 shows the comparative similarity of the embeddings of 80 sketches of apples (and their grid counterparts) to the label "sketch of an apple" using the CLIP variant `ViT-L/14`, as a probability. The probabilities are calculated using the equation (2.4), with temperature $T = 1$, to show the trend. The categories for this example are "apple", "chair", "car", "flower", "pencil", "house", "tree", "fish", "star", and "bird". The middle row shows a sample of the images for each of the cases. The bottom row shows the accuracy (percentage of correct predictions) for each of the cases.

We found that CLIP can recognize these sketches quite well. As the grid became coarser, the confidence and accuracy of CLIP decreased significantly. For very coarse grids, the accuracy is almost as bad as random guessing. We also experimented with inverting the images and found that results were consistently slightly better with black-on-white sketches (shown here) than with white-on-black sketches. Yet, we did not notice any difference in performance with the addition of grayscale values in the pixel blocks. In most cases, we also noticed that using specific labels like "apple" for a picture of an apple had better accuracy than using hypernyms like "fruit" (Fig. 3.3).

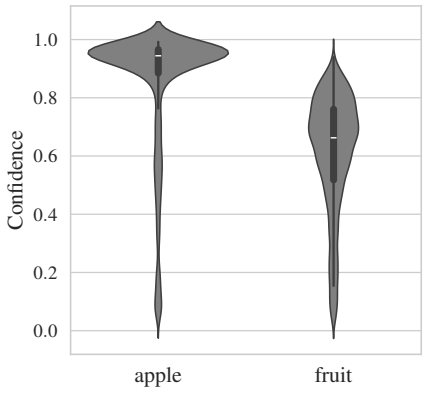


Figure 3.3: CLIP on different specificity of labels.

Additionally, we found that the bigger CLIP models performed better on these tasks than the smaller ones which is in line with observations from other studies on VLMs as a source of rewards. Furthermore, we tried the same experiments with different sets of categories, prefixes, and suffixes, but the results followed similar trends.

3.2.2 CLIP on Tangram

Tangram has lesser degrees of freedom than ShapeGridWorld and it is more abstract but it still allows for rich creative expression. To test if it would be a feasible environment for our study, we used some images of creations on Tangram from the internet and used CLIP as a zero-shot classifier as above to test its performance. We again found it to be quite good at recognizing these creations with very high confidence. Fig. 3.4 shows a few examples.

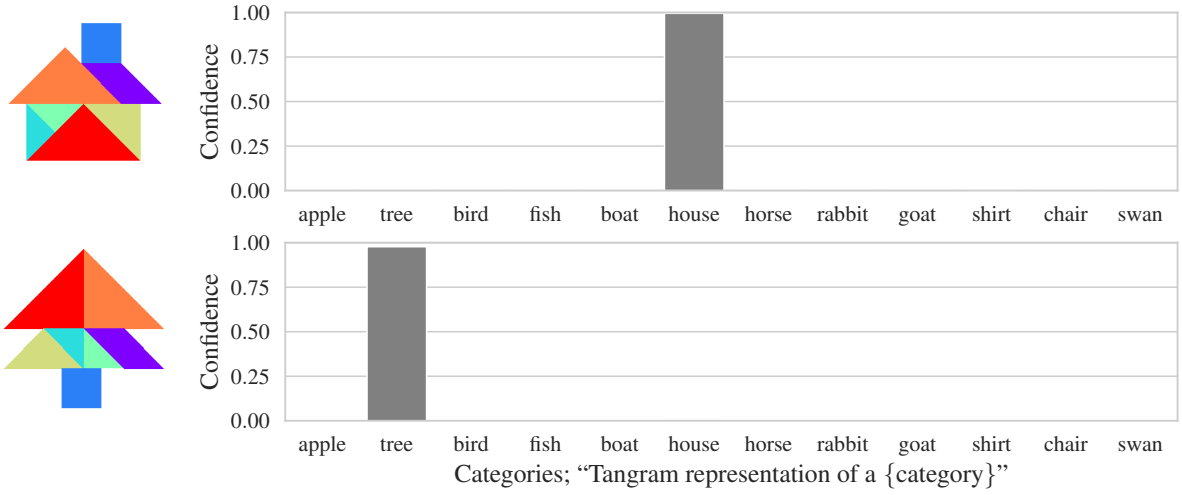


Figure 3.4: CLIP on a few simple Tangram creations.

We did not find a significant trend in the effect of colors on the performance of CLIP, but inversions seemed to slightly affect the distribution of inferences in certain cases (one such example is given in the appendix Fig. A.2, where the distribution of CLIP was skewed in grayscale and white-on-black cases). Consequently, we used colored or black-on-white binary renderings for our experiments.

More experiments on these hyperparameters are discussed in the later sections.

3.3 Trajectory Analysis with Random Rollouts

Although we found that CLIP is quite good at recognizing these creations, we had to ensure that it would be a good source of rewards for our environments, or that the controller would be able to exploit it well to reach a sufficiently good local optimum. To study the semantics entropy reward landscape, we conducted a series of experiments with random rollouts in the environments, i.e. starting from a meaningful creation, we perturbed the creation with randomly sampled actions and observed the effects on the reward.

These experiments showed that CLIP is quite susceptible to noise in its inferences, i.e. its probability distribution responds abruptly to small adjustments in the input image. As a consequence of this, the reward is quite sparse, and more critically, there is a large semantic bias in not-so-meaningful images. The sections below discuss these problems in detail.

3.3.1 Sparse Rewards

We observed that the inference of CLIP breaks suddenly with small changes in the image, i.e. its confidence in its classification is very sensitive. This results in a sparse semantics entropy reward landscape. Fig. 3.5 illustrates this problem.

Sparse rewards are a problem in reinforcement learning in general because underly a lack of continuous feedback. With only sporadic rewards, it becomes exceedingly difficult for an agent to gauge the values of actions to lead it to eventual rewards, i.e. credit assignment becomes difficult, which in turn leads to difficult and inefficient learning of action policies.

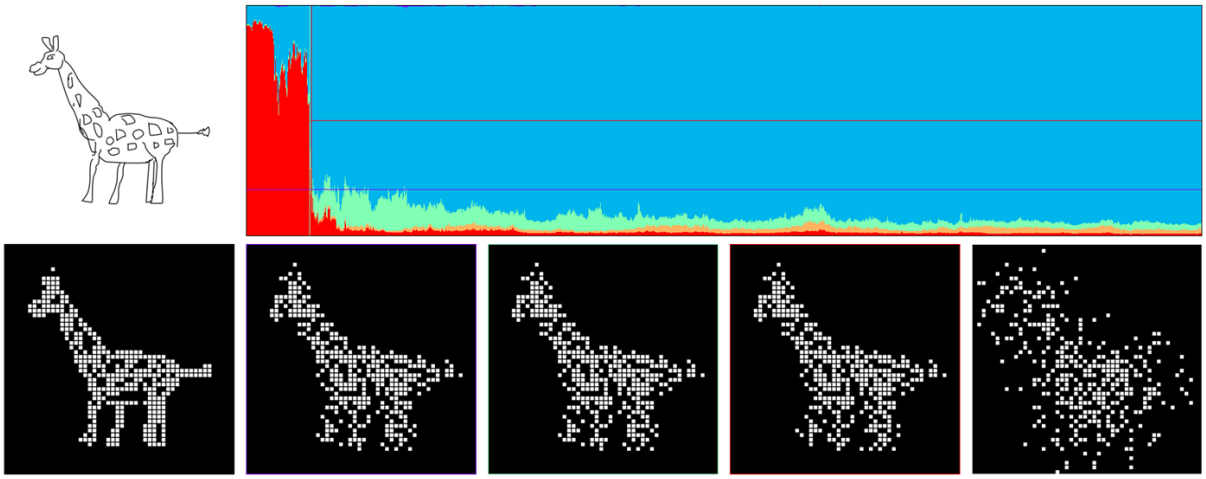


Figure 3.5: Random rollouts in ShapeGridWorld. Going from left to right in time, we perturb the grid pixel-by-pixel and observe CLIP's confidence change. The top row shows the distribution of CLIP's inferences over the labels "giraffe", "number", and "random" in red, green, and blue respectively. The first and last images in the bottom row are the initial and final images respectively. The middle three images in the bottom row show the images where the distribution changed its trend, at the last time when "giraffe" was the most confident category, at the time when the distribution was equally split between the categories, and the first time when "random" was the confident category. They are all just a few pixels away from each other but essentially still the same, yet they are much different for CLIP.

Thus, a smooth and continuous reward is essential for effective learning. Since we used the iCEM controller which is a gradient-free optimization algorithm and has proved to work well in sparse reward settings, it helped to mitigate this problem.

3.3.2 Semantic Bias in Random Images

The other consequence of the noise in CLIP inference is that can sometimes confidently classify random images to a class, i.e. false positives, instead of having a flat distribution. Fig. 3.6 shows some examples of Tangram creations in which CLIP displayed high confidence.

This can also be interpreted as a jagged reward landscape with many local optima. This noisy reward signal can lead the agent to be stuck in local optima (with a rather meaningless creation that it erroneously finds confident), or inhibit it from reaching a good optimum.

The two problems, sparse rewards and semantic bias in random images are not independent. They have the same underlying cause – noisy CLIP inference, yet there's a non-trivial anticorrelation between reward sparseness and noise. That is, increasing reward density involves increasing confidence in imperfect/partial creations which also increases noise.

Class Preference in CLIP

Another complementary issue we faced related to the problem of semantic bias in random images was that of class imbalance. Simply put, there were some classes in which CLIP was consistently more confident than others and leaned towards them over others when unsure. This included classes that signified broad concepts like "animal", "fruit", "number", "letter", or "object".

This is a fundamental nature of the CLIP model and we could only mitigate it by avoiding certain categories and choosing enough categories for the semantics entropy reward such that CLIP's preferences balanced out. More discussion on tackling this problem in particular is given in section Section 3.5.1.

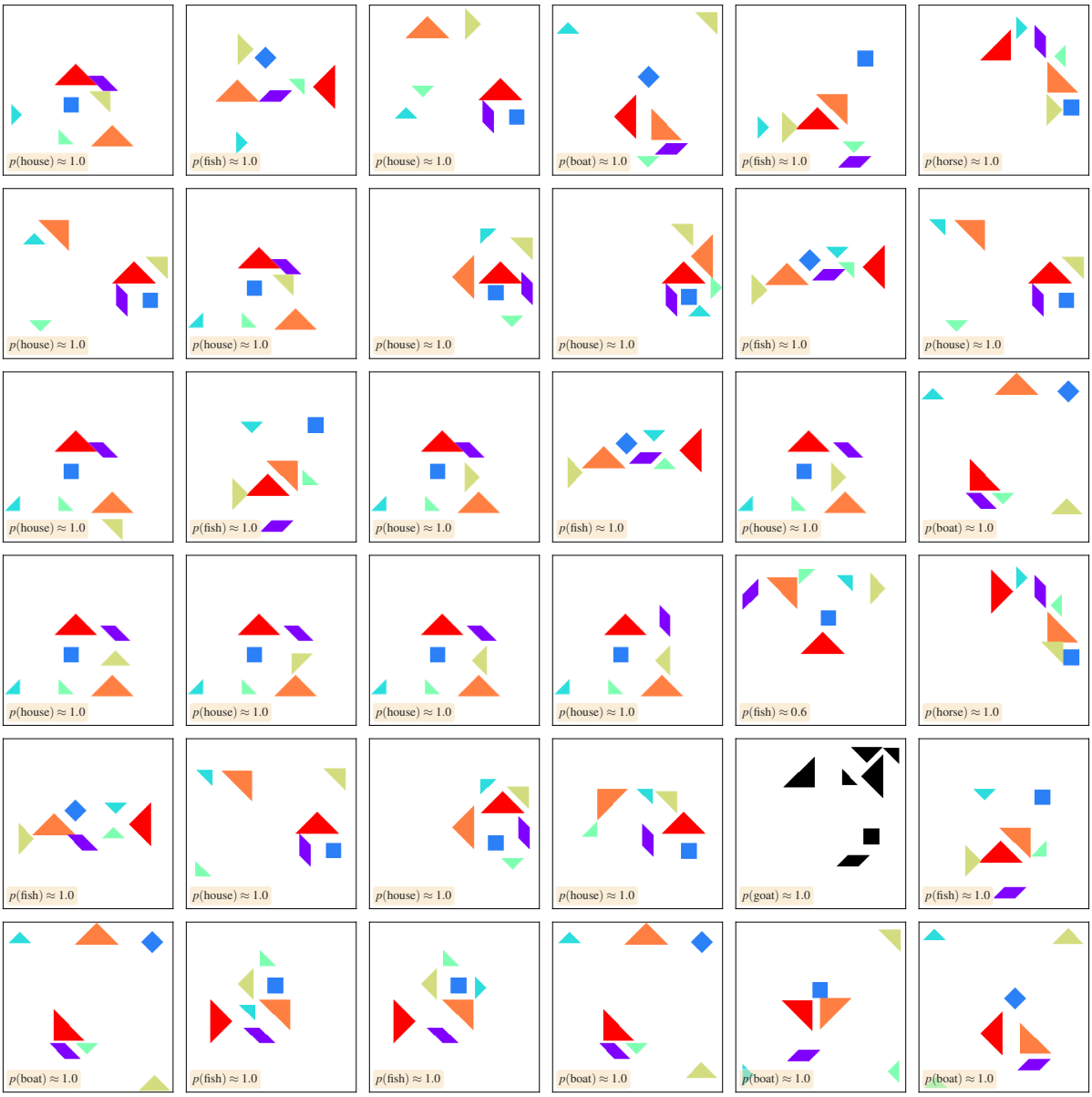


Figure 3.6: Large semantic bias in random images.

3.4 Trajectory Analysis with Closeness Costs

Before we started improving and running simulations using inferences from CLIP, we needed to ensure that the controller was able to reach a reward-conditioned goal in our environments, i.e. it was *solvable*. Especially, it was crucial to establish this solvability under sparse rewards.

To ensure that our controller did this efficiently and robustly, a good choice of the iCEM controller hyperparameters and the environment configuration was essential. These hyperparameters and configurations were dependent on each other so we needed to optimize them together.

For example, for both environments, the size of the grid together with the step size affects the minimum required planning horizon for the controller. For a small discrete grid or a large enough step size, the planning horizon does not need to be very high, because the controller can potentially reach the goal in a few steps and does not need to plan far ahead. If the step size is too small for the grid size, a longer planning horizon is required so that the agent can discover the reward by random sampling.

In a similar vein, the step size is also related to the *object persistency* used in the environments. This

parameter governs how many steps a single object (pixel block in ShapeGridWorld and a polygon in Tangram) is in focus for the actions of the controller before it cycles to the next object in the sequence. For a high object persistency, the step size could be lower, but for a low object persistency, the step size needs to be higher.

These interdependencies made the hyperparameter optimization problem quite complex and the sheer number of these parameters made this task very expensive. Given the high computation resources and time required to run simulations with the semantics entropy reward using CLIP, it was infeasible to do a proper search over all the hyperparameters. Thus, we instead used an alternative reward function to analyze the effect of the hyperparameters. We called this ersatz reward function the *closeness reward*.

A closeness reward is defined in the context of a fixed target creation in the environment, and its function is given as the negative of a distance function between the current creation and the target creation in the state space of the environment.

We used two different formulations of the distance function, which we called the *dense closeness cost* and *sparse incremental closeness cost*. The dense closeness cost is formulated as the L^2 distance between the current and target creations and the sparse incremental closeness cost is formulated as the dimension-wise thresholded L^1 distance between the current and target creations, given by,

$$d(\mathbf{i}, \mathbf{t}) = \sum_{k \in n_s} \mathbf{1}_\varepsilon(\mathbf{i}_k - \mathbf{t}_k), \tag{3.1}$$

where $\mathbf{i}, \mathbf{t} \in \mathcal{S}$ are the current and target creations respectively, ε is a small threshold, and $\mathbf{1}_\varepsilon$ is the indicator function such that $\mathbf{1}_\varepsilon(x) = 1$ if $|x| > \varepsilon$ and 0 otherwise.

Using the closeness reward/cost formulations, we performed an extensive grid search over the many hyperparameters of the iCEM controller and the environment together to find the best combinations and study their effects. To assess and compare the results in these experiments, we compared their cumulative rewards over the rollouts.

The controller was easily able to achieve the goal under dense closeness rewards for a wide range of hyperparameters and other than a few parameters such as the granularity of the grid, the step size, and the object persistency, we did not notice much difference in the performance of the controller over the other hyperparameters. Fig. 3.7 shows a sample closeness reward run for the dense closeness cost on the Tangram environment.

The tuning was more difficult under sparse incremental rewards, which required a higher planning horizon and more sampled trajectories to reach the goal. This directly corresponds to a higher computational cost.

The cost aggregation functions `best` and `sum` were comparable in performance and performed better than `best-1`, *last*, and *last*-1. Although the greedier `best` aggregation function was more efficient in reaching the goal, for our semantics reward simulations, we mainly considered the `sum` aggregation function as it's more robust to noise due to its summation.

The comparison results are omitted here for brevity. Please refer to the appendix sections B and A for more details about these hyperparameters of the iCEM controller and the environments respectively.

From this analysis, we were able to gain an intuition of the minimal required set of hyperparameters that allowed the controller to solve the environment. These minimal parameters were then used as a starting point for the simulations with the semantics entropy reward.

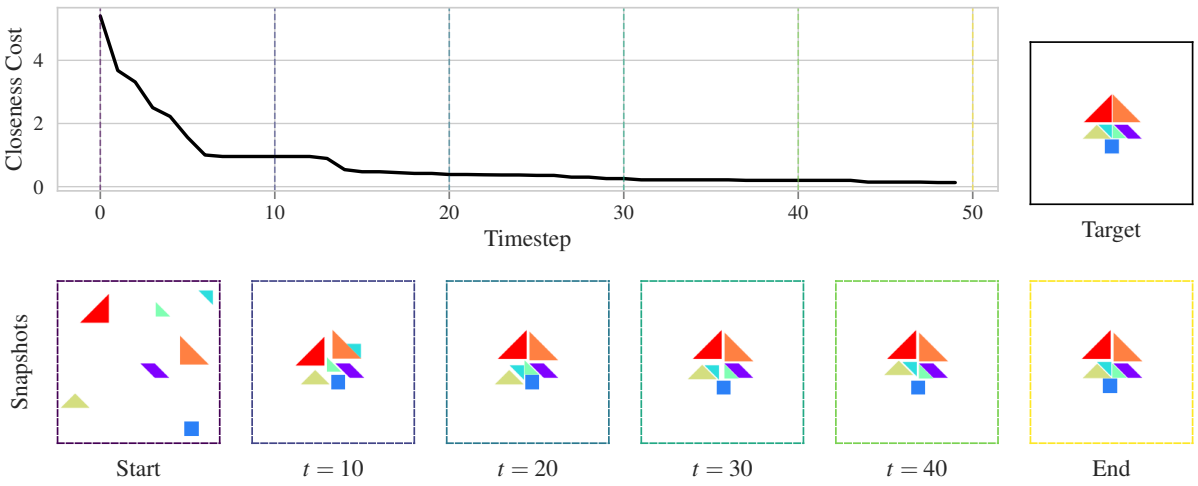


Figure 3.7: A successful closeness cost rollout in Tangram.

3.5 Improving CLIP Rewards

We experimented with several methods to reduce the noise from CLIP. The regularized semantics reward also has many hyperparameters that can be tuned to make the reward landscape smoother. Namely, the temperature of the softmax function (T), the target baseline regularization strength (α), the image baseline regularization strength (β), the use of negative embeddings, and tweaking the rendering function (adding texturing or modifying the images with other operations).

The complex nature of the reward landscape and the high dimensionality of the state space made it difficult to analyze the effects of these hyperparameters. To gain an intuition over these parameters and reduce the search space before we ran simulations using them in the full semantics reward, we instead used the best of the previously rolled-out trajectories with the closeness reward from Section 3.4 and ran post hoc inference on the resulting sequence of image observations using CLIP and calculated the trajectories of the resulting semantics reward.

Subsequently, we performed an ablation study on these hyperparameters to gain insights into their effects[1].

Additionally, the performance can highly depend on the choice of prefix and suffix (Fig. 3.8).

To get the underlying trends in these parameters and find robust values that were agnostic to prompt engineering, i.e. the choice of prefix and suffix in formulating the text input, we average over our results across multiple choices and combinations of these text parameters.

We observed clear trends in the effects of these hyperparameters on the reward landscape. The following sections on the specific hyperparameters discuss the results of these analyses. In the figures shown in these sections, we choose the best-performing other hyperparameters. These results can be compared to the corresponding adversarial study results discussed in Section 3.5.7.

3.5.1 Effect of the Number of Creative Possibilities

To alleviate the problem of class preference in CLIP, we experimented with different numbers of categories/creative possibilities (c) for the semantics entropy reward. Fig. 3.9 shows the effect of the number of categories on the entropy of CLIP inferences over a random image.

[1]The myriad of combinations due to the high dimensional space makes it infeasible to show all the subtleties of the interplay between these hyperparameters in a few figures. Thus, we have additionally made the analysis available for the reader as an interactive notebook at https://colab.research.google.com/drive/1UzKb5t5PDRO05GbSKpgzWd3DELfAzDxJ or https://t.ly/j84on, where one can pick a combination of different hyperparameters to see their effects

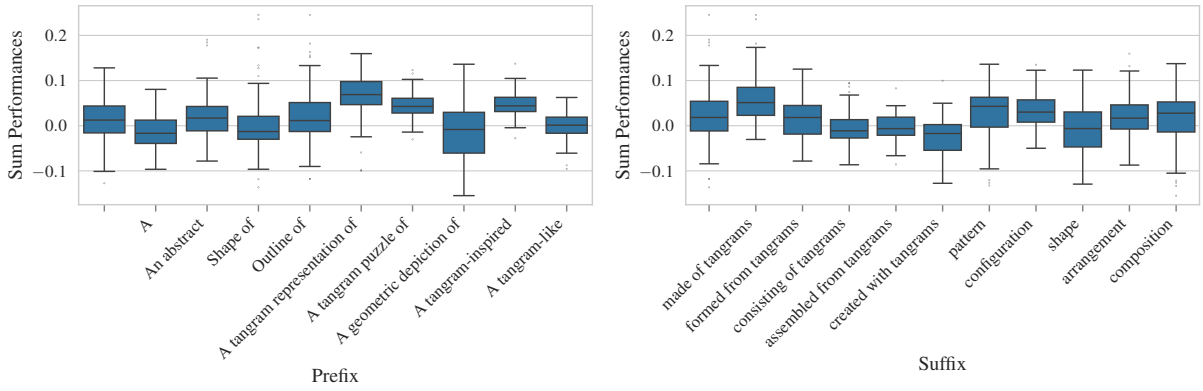


Figure 3.8: Effect of different prefixes and suffixes on semantics entropy reward trajectories.

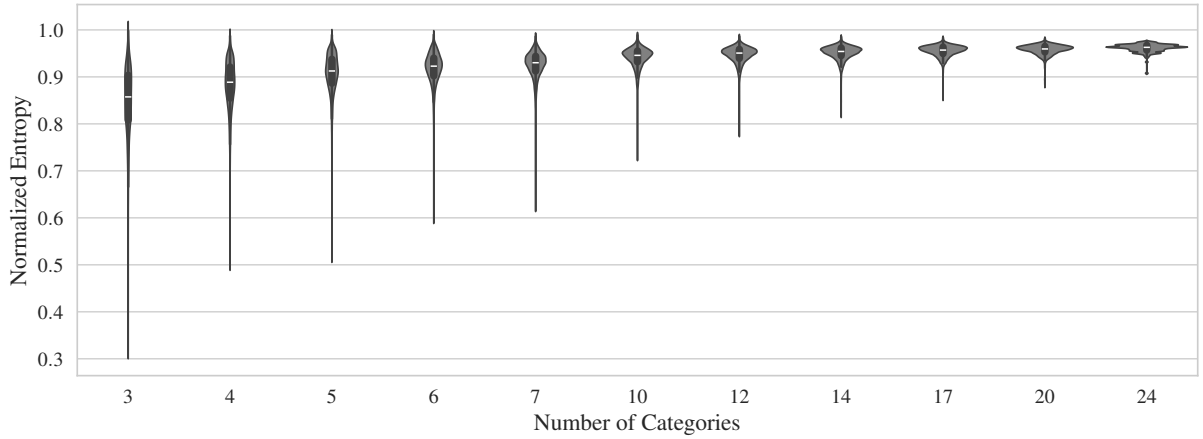


Figure 3.9: Effect of the number of categories on CLIP inference distribution over a random image.

Too few categories can promote semantic bias and class preference in random images, but too many categories can exacerbate the problem of noisy rewards since there are potentially more misleading distractions for the agent.

It was important to choose categories that were semantically distinct and not too broad which was non-trivial. Especially for the Tangram environment, in which the creations can be quite abstract, there was only a limited set of feasible categories. This suggested that a moderate number of categories would be optimal.

For our further simulations, we chose a set of $10 \sim 20$ categories.

3.5.2 Effect of Temperature

Temperature is a crucial hyperparameter for the softmax function in the semantics entropy reward as it directly controller the entropy of the CLIP distribution. The original CLIP publication recommends a temperature of 0.01 for most use cases, which we also found to most often work best. Temperatures up to 0.02 were good as well. Any temperature below this range performed significantly worse as it exacerbated the noise of CLIP, and any temperature above this range made the reward landscape too smooth and the reward trajectories too flat.

Fig. 3.10 shows the effect of temperature on the semantics entropy reward trajectories, averaged over multiple choice of categories. This is a post hoc CLIP inference on trajectory samples from the closeness reward rollout experiments.

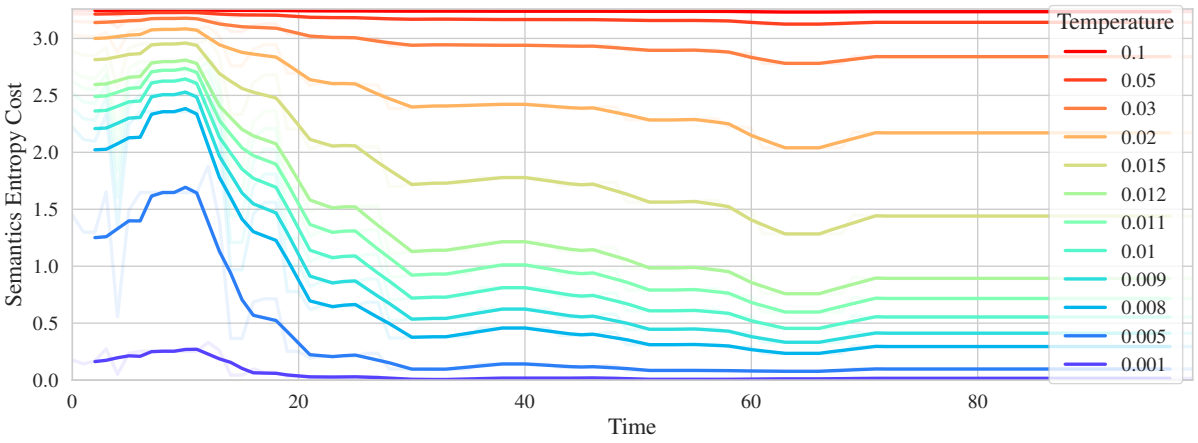


Figure 3.10: Effect of temperature on semantics entropy reward trajectories.

3.5.3 Effect of Baseline Regularization

The regularization strength of the target baseline (α) and the image baseline (β) were arguably the most important hyperparameters for the regularized semantics entropy reward. Baseline regularization helps with the problem of sparse rewards by providing better directional guidance in the CLIP embedding space.

The motivating study of *VLM-RM* noted the best value of the target baseline regularization strength in goal-regularized settings to be somewhere in the middle of the two extremes. We found a similar trend in our experiments, with a slight preference for the additional image baseline regularization strength to be higher than the text baseline regularization strength.

In our analysis, we compared the cumulative rewards of the post hoc inferences on closeness reward trajectories with different values of α and β to find the optimal values. Fig. 3.11 shows these results, averaged over multiple choice of prefixes and suffixes.

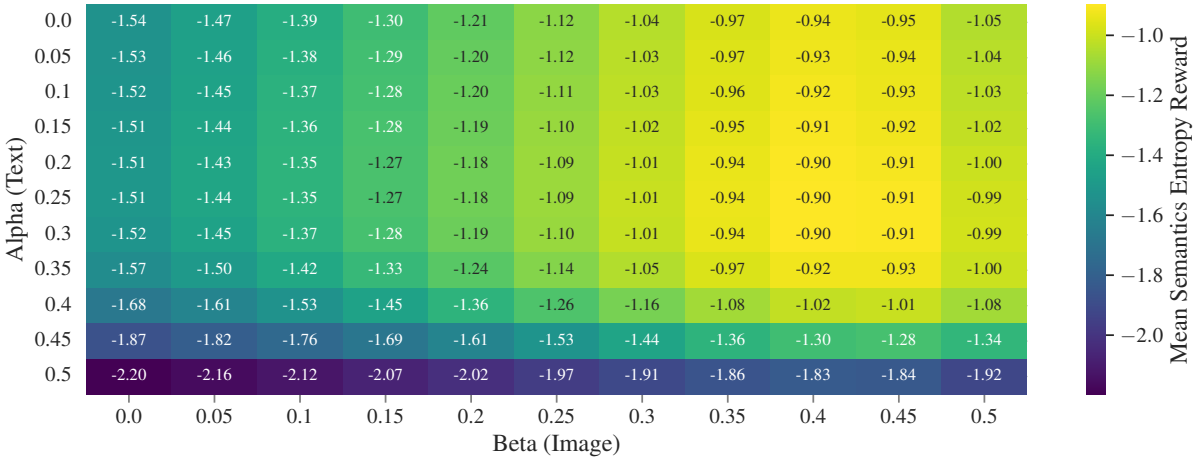


Figure 3.11: Effect of regularization strength on semantics entropy reward trajectories.

3.5.4 Effect of Negative Embeddings

Baumli et al. (2023) used negative embeddings as target baselines in their goal-conditioned CLIP reward function to make the reward landscape smoother. We experimented with this as well, and instead of using the initial description of the environment as the text baseline, we used a negative embedding of the target creation.

We did not find this to be helpful in our experiments, instead it seemed to flatten the reward landscape. We think this is a consequence of CLIP's language encoder being a bag-of-words model, and the negative embeddings being essentially close to the target embeddings, which effectively zeros out the target embeddings.

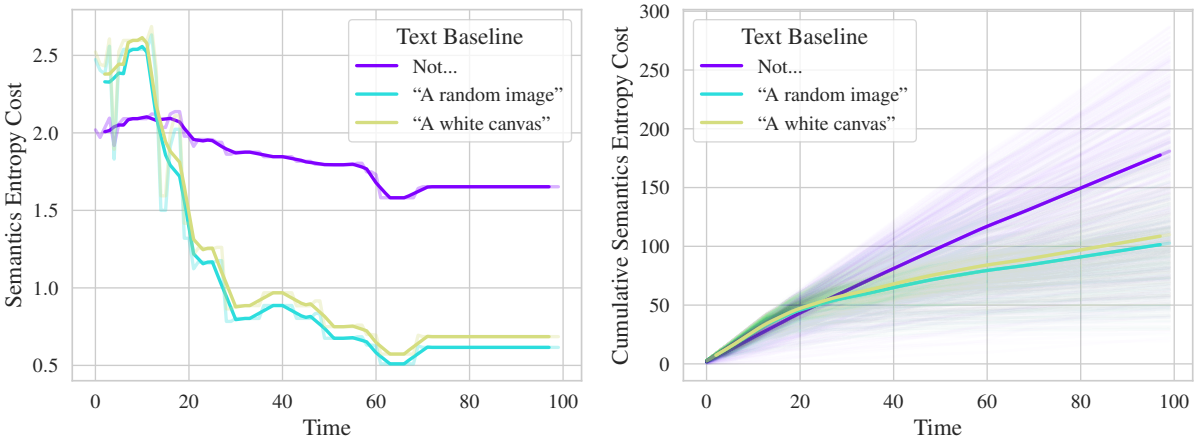


Figure 3.12: Effect of different text baselines on semantics entropy reward trajectories.

3.5.5 Effect of Adding Post-Suffix

Roth et al. (2023) found that adding a *post*-suffix to the label in the text input to CLIP improved the quality of the inferences. This post-suffix can surprisingly even be a random string of characters.

Following these insights, we also experimented with adding a random string of jibberish to the end of the label as a post-suffix, but we did not find it to affect the reward. Fig. 3.13 shows the effect of different post-suffixes on the semantics entropy reward trajectories, averaged over combinations of prefixes and suffixes.

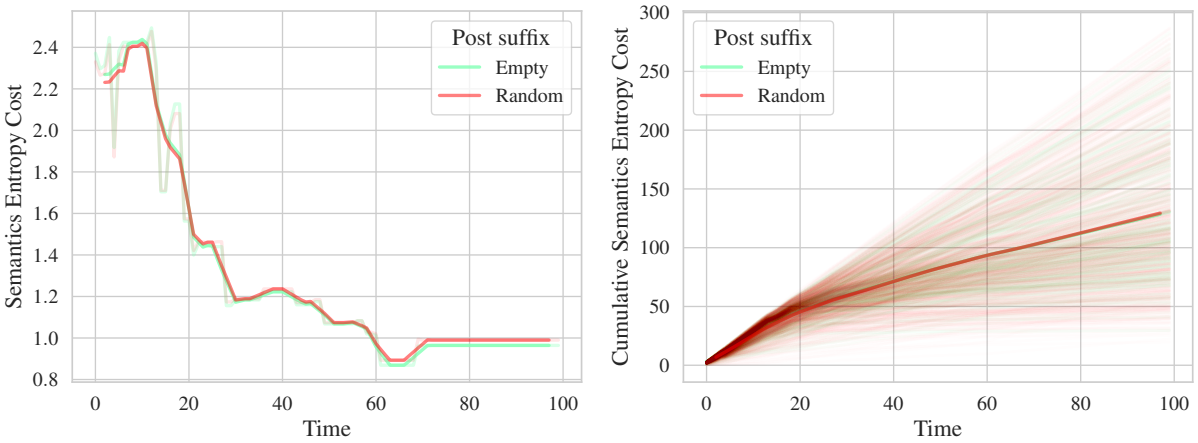


Figure 3.13: Effect of different post-suffixes on semantics entropy reward trajectories.

3.5.6 Entropy Regularization

Another promising way to make the reward landscape of CLIP smoother is to fine-tune it with an additional entropy regularization loss over its output. This is given by,

$$L(\boldsymbol{i}, \boldsymbol{l}) = L_\iota(\boldsymbol{i}, \boldsymbol{l}) \underbrace{- \sum_{\mathbf{i}_k \in \boldsymbol{i}} \sum_{\mathbf{l}_j \in \boldsymbol{l}} p(\mathbf{l}_j; \mathbf{i}_k, \boldsymbol{l}) \log p(\mathbf{l}_j; \mathbf{i}_k, \boldsymbol{l})}_{\text{Entropy Regularization}}, \tag{3.2}$$

where L_ι is the contrastive cosine-similarity loss used to train CLIP and $p(\mathbf{i}_k, \boldsymbol{l})$ is the classification probability distribution of image $\mathbf{i}_k$ over the labels $\boldsymbol{l}$ predicted by CLIP.

We experimented with this regularization method on a toy convolutional neural network (CNN) for classifying handwritten digits, which we called *Flatnet*. We found it to be quite effective in smoothing the reward landscape; it seemed to relieve both of the problems from Section 3.3 – the reward trajectory was smoother and we observed less semantic bias in random images.

Yet, we did not use it to fine-tune CLIP to limit the scope of the project given the limited time. More information of this analysis can be found in appendix E.

3.5.7 Adversarial Performance

To tackle the problem of semantic bias in random images, we collected samples of the false positive image observations from our rollouts, called *adversarial observations* (Fig. 3.6), by filtering out the image observations with low entropy from all our runs and then manually removing the ones that were true positives. Then, we searched for the semantics reward regularization hyperparameters that make CLIP less confident in the adversarial images while maintaining its inference for the truly semantically expressive images, i.e. decrease its specificity while maintaining its sensitivity.

We used the difference in mean entropy of distributions for the false positive and true positive images as a metric to gauge the performance of CLIP. Fig. 3.14 to 3.17 show the effect of temperature, regularization, negative embeddings, and post-suffixes on discerning true positives from false positives.

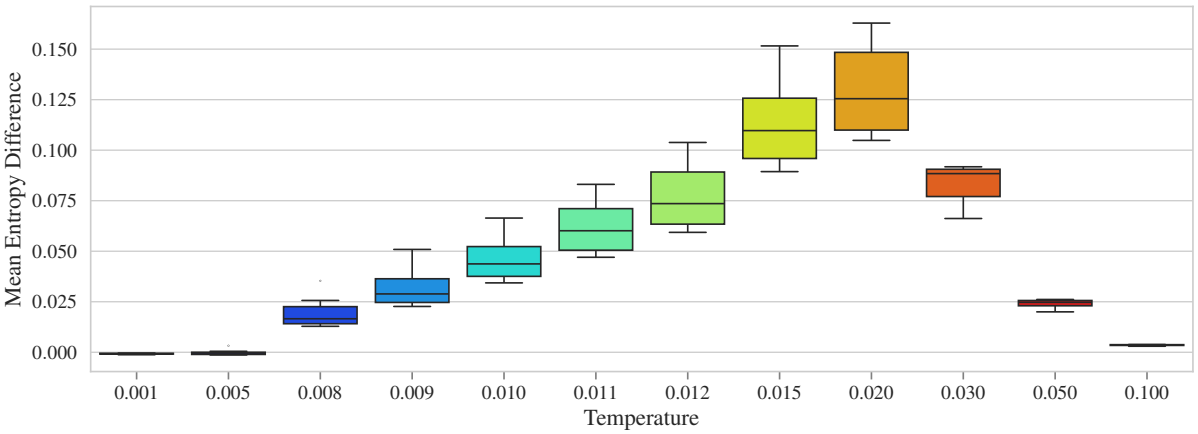


Figure 3.14: Effect of temperature on discerning true positives from false positives.

Higher temperatures of 2 were better at smoothing the reward landscape just enough to improve its adversarial performance.

Interestingly, the trends in the regularization strength of the text baseline (α) and the image baseline (β) shown in Fig. 3.15 were essentially opposite from the ones in Fig. 3.11 that showed the effect on the reward landscape. An optimal choice of baseline strength for reducing the sparsity of the goal-based reward landscape essentially improves the path to this goal by improving the gradients to reach this desired reward. This means that it changes the reward at near-goal status in such a way that it is better

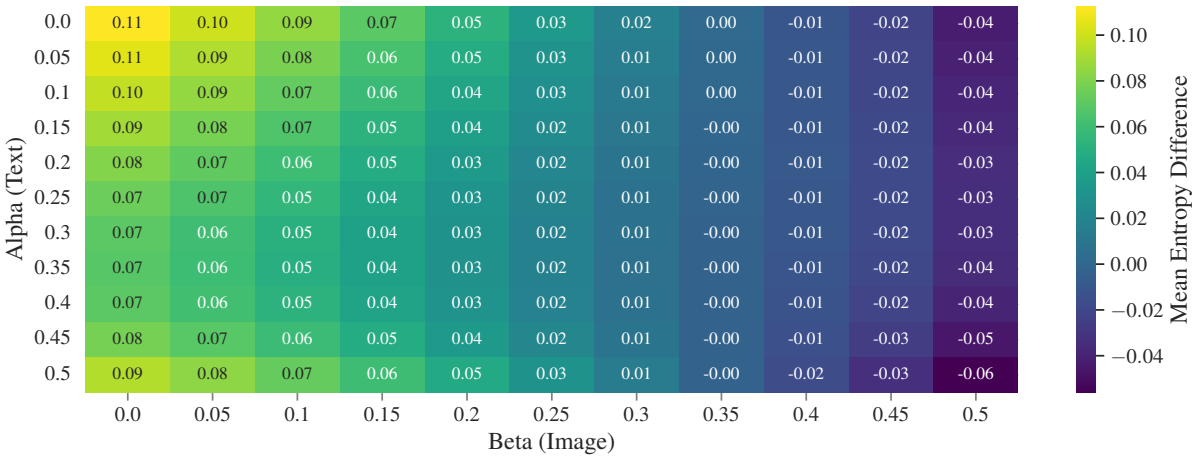


Figure 3.15: Effect of regularization strength on discerning true positives from false positives.

and hinting the agent if it's close to its ultimate goal, i.e. it increases the similarity or correspondance to this state to the goal state. This is in contrast to the adversarial performance where we desire to make this difference more pronounced. Thus, these results are not at odds with each other, but rather complementary.

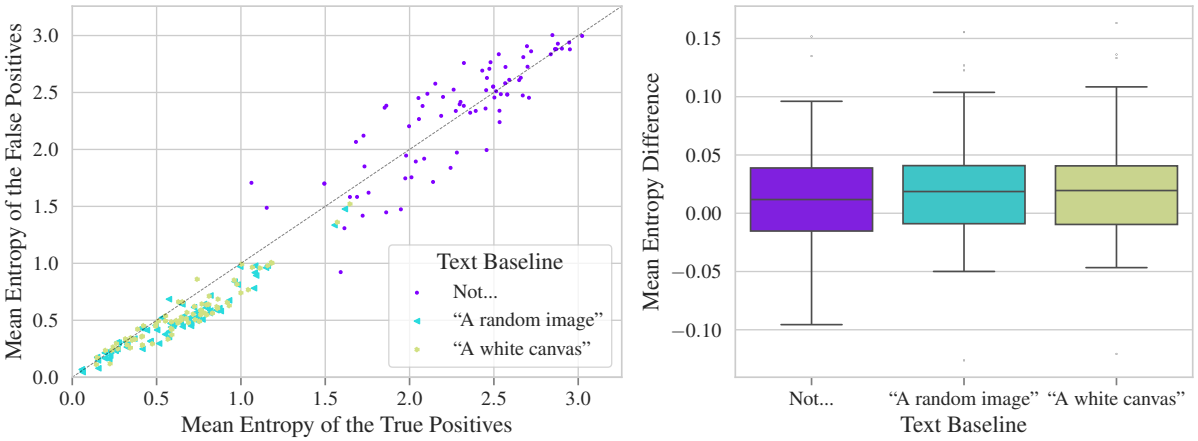


Figure 3.16: Effect of different text baselines on discerning true positives from false positives.

Fig. 3.16 left subfigure shows the adversarial performance of negative embeddings in a 2D space. Each of the points is a different combination prefix and suffix for the text input. Ideally, we would like the points to live above the diagonal, i.e. the entropy of the false positives should be higher than the true positives.

Negative embeddings as we have seen before, flatten the reward landscape uniformly and as a consequence increase the entropy of the true positives as well. This results in no performance improvement in the adversarial case.

Fig. 3.17 shows the effect of different post-suffixes on the adversarial performance. Since we were not sure of the exact mechanism for choosing a post-suffix that would work well in general, for this analysis, to ensure more exhaustive and rigorous tests, we also used different formulations of the post-suffix and considered cases such as having the same random post-suffix across all the categories for every prefix-suffix combination or having a random one for each of the categories and combinations. As before, we see no difference in adding or omitting the post-suffix.

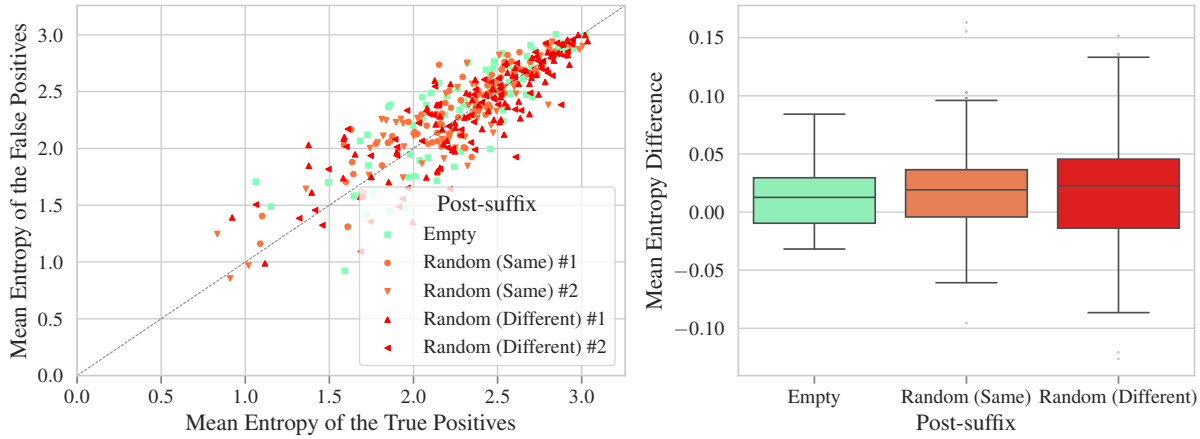


Figure 3.17: Effect of different post-suffixes on discerning true positives from false positives.

3.5.8 Effect of Image Texturing

Following the observations from Rocamonde et al. (2023), to make the image inputs to CLIP more photorealistic to ensure in-distribution performance, we also experimented with adding textures to the images to make them more realistic for CLIP in hopes of improving its quality. This texturing was done by adding a constant shading to the Tangram polygons. Fig. 3.18 show the effects of this texturing on the reward landscape. We did not find a significant improvement in the reward landscape with this texturing.

3.5.9 Effect of Image Operations

We also had the idea to use subtle image operations like shearing before inference. We expected the true positive semantic inferences to be invariant to these operations, but the false positives to be reduced, thus reducing the noise in the reward landscape.

This is compared together with the texturing operations in Fig. 3.18 and Fig. 3.19 to study their effects on the reward landscape and the adversarial performance respectively.

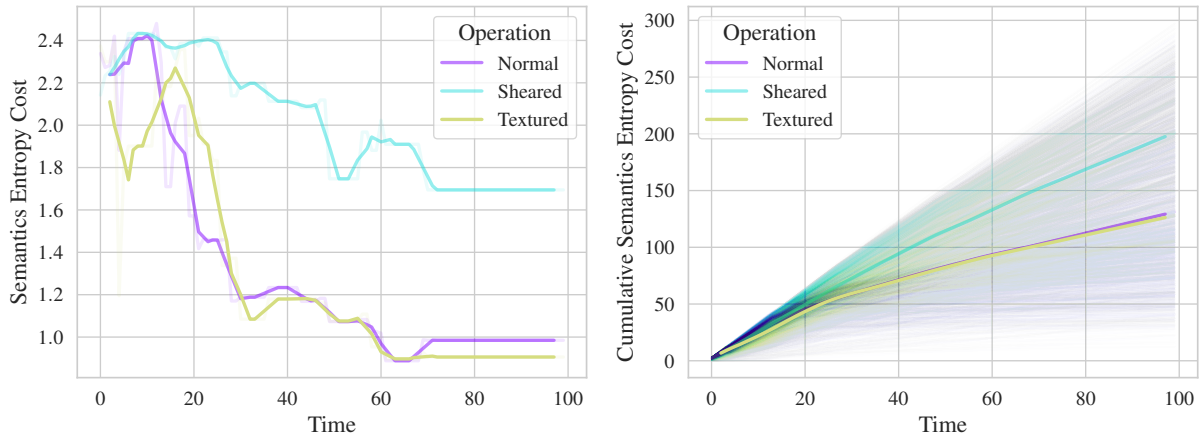


Figure 3.18: Effect of texturing and image operations on semantics entropy reward trajectories.

Shearing the images before inference did improve the quality of the inferences slightly by smoothing the reward landscape. It had a significant performance improvement in the adversarial case as well, but it also affected the confidence in the true positive images, thus leading to lower performance by our cumulative reward metric.

To show the interdependence of the hyperparameters discussed above, Fig. 3.20 and Fig. 3.21 compare

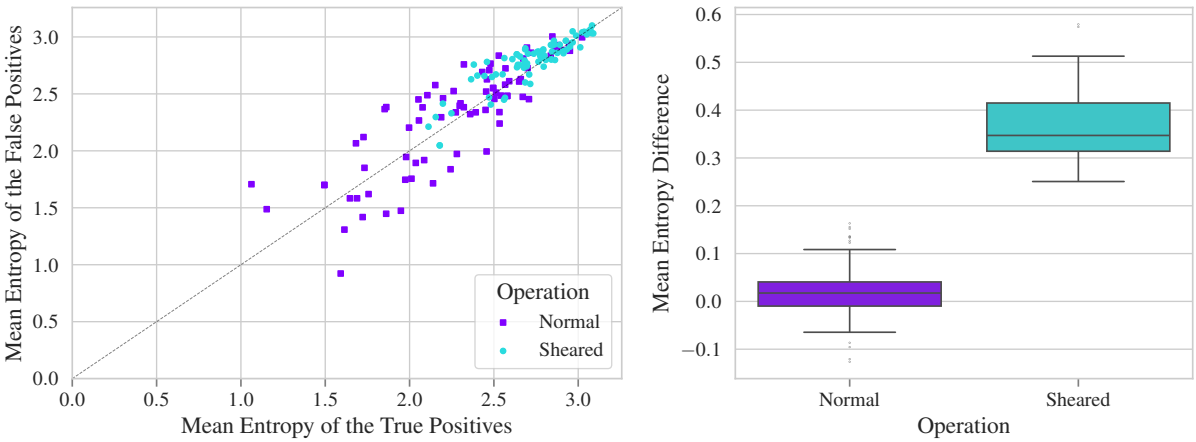


Figure 3.19: Effect of image operations on discerning true positives from false positives.

the adversarial performance of the sheared images across negative embeddings and different post-suffix patterns respectively. We observe that the shearing operation when used with baselines based on initial states is very helpful in improving the adversarial performance of CLIP. Although this is helpful, in the following sections, we did not use this method, because of the associated computational costs, and because a similar effect could be achieved by tuning the regularization strengths and the temperature.

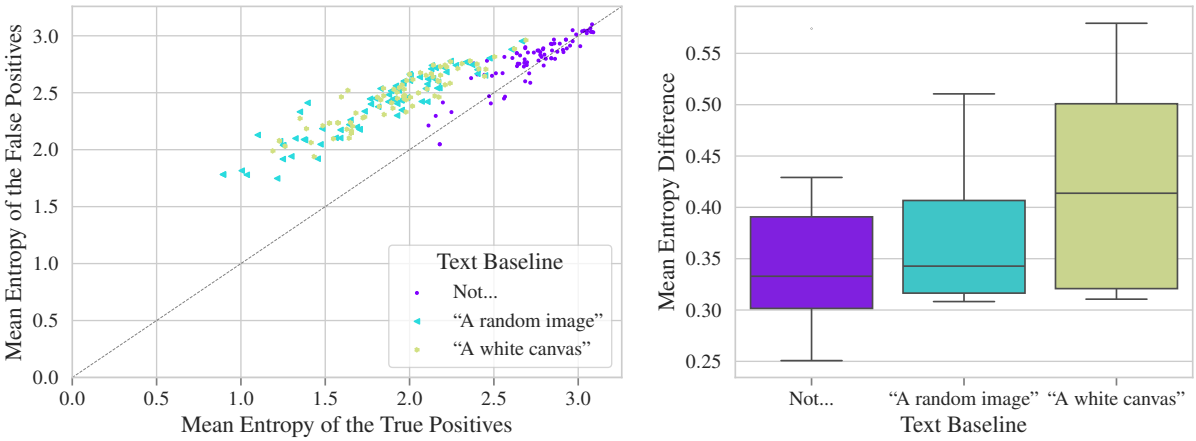


Figure 3.20: Effect of different text baselines on discerning true positives from false positives with sheared images.

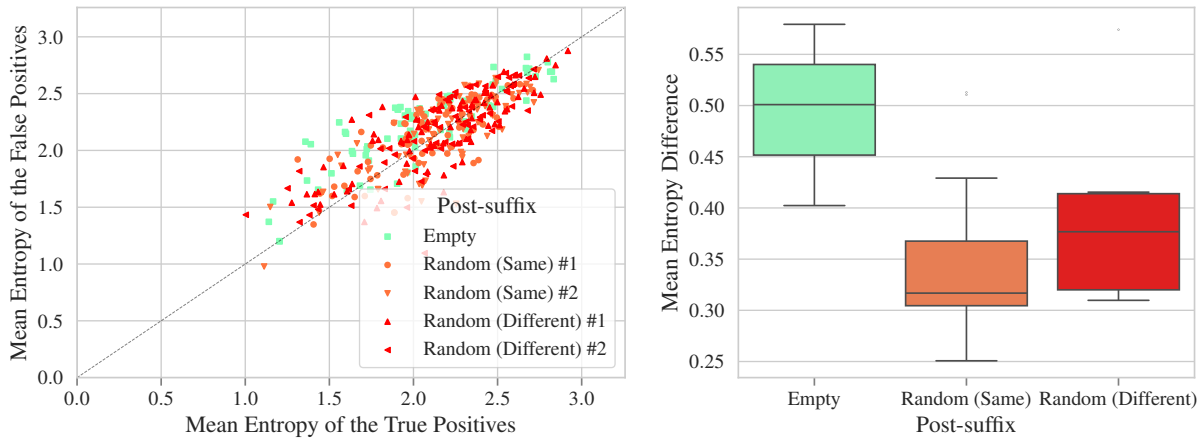


Figure 3.21: Effect of different post-suffixes on discerning true positives from false positives with the sheared image. Note that the temperature here is lower, $T = 0.012$, the points gather in the upper right corner at higher temperatures.

3.6 Simulations

Using the suitable combinations of the controller and environment hyperparameters from Section 3.4 and intuition of the regularized semantics entropy reward hyperparameters Section 3.5, we ran simulations with the semantics reward to generate creations in the environments. We were generally able to generate meaningful creations in both environments, but the quality of the creations varied significantly.

We further investigated the effects of a few hyperparameters on the quality of the creations. To evaluate these creations, in addition to the cumulative reward, we also asked three human evaluators to manually rank the final creations on a scale of 1 to 5 based on their apparent semantic expressiveness.

These simulations were computationally expensive and required a lot of time to run. We were limited in our ability to fine-tune all the hyperparameters to the fullest extent, so we focused on the main few – the regularization strengths and the effect of additional regularity reward RaIR. We showcase the results of these analyses in the following sections.

While we ran many additional simulations to confirm our intuitions about the hyperparameters, only a few of our analyses used enough restarts with different random seeds to have enough statistical power to make justified claims. Interesting creations from these simulations are shown in Section I.

3.6.1 Effect of Regularization Strenths

We ran simulations with different regularization strengths to reconcile their effects with their results from the previous sections.

Fig. 3.22 shows the effect of regularization strengths on the semantics entropy reward trajectories.

We find that the mid-values of the regularization strengths are the best for the performance of the controller, which is in line with the results from the ablation study in Section 3.5.3. Fig. 3.23 shows the heatmap of the mean cumulative rewards for different combinations of the regularization strengths and Fig. 3.24 shows some samples of the creations with different regularization strengths. For standard deviations, please see Fig. H.1.

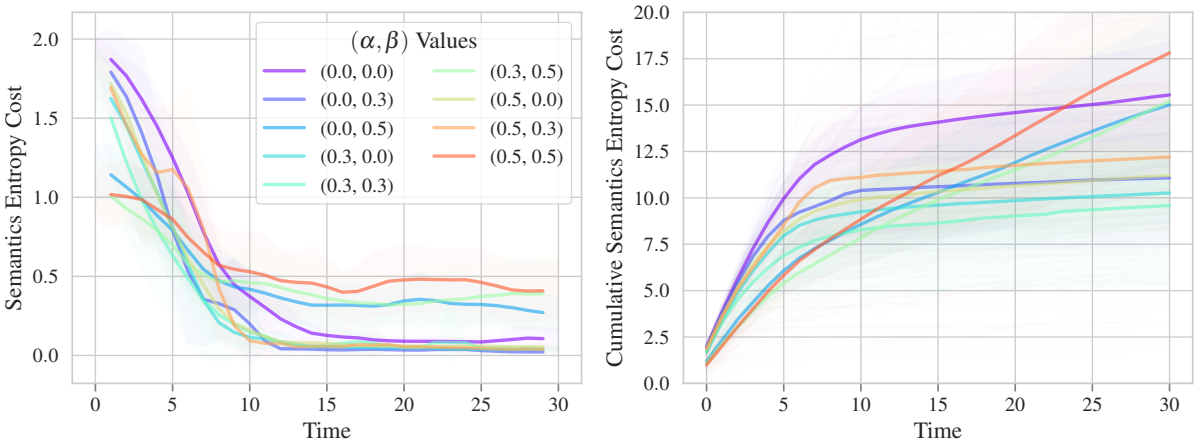


Figure 3.22: Effect of regularization strengths on semantics entropy reward trajectories.

3.6.2 Effect of RaIR

Throughout our simulations, we noticed a high degree of regularity in semantic creations and there was always a high correlation between the RaIR and the semantics entropy reward. Indeed, adding the structural bias from RaIR helped with grounding and coalescing the creations in more human-recognizable forms and patterns. This was particularly helpful in the beginning of the planning as the agent starts building regular structures to bring it closer to a semantically meaningful state which is then further refined by the semantics entropy reward. We found that creations simulated with RaIR were much more recognizable than without it. Fig. 3.25 shows the effect and Fig. 3.26 shows some samples of the creations with and without RaIR.

We also tested this in ShapeGridWorld, where the effect of RaIR was only marginal and only affected when the pixels had grayscale values.

Fig. 3.27 and Fig. 3.28 show the effect of RaIR on ShapeGridWorld with grayscale pixels.

Results for additional simulations on ShapeGridWorld are given in Section G.

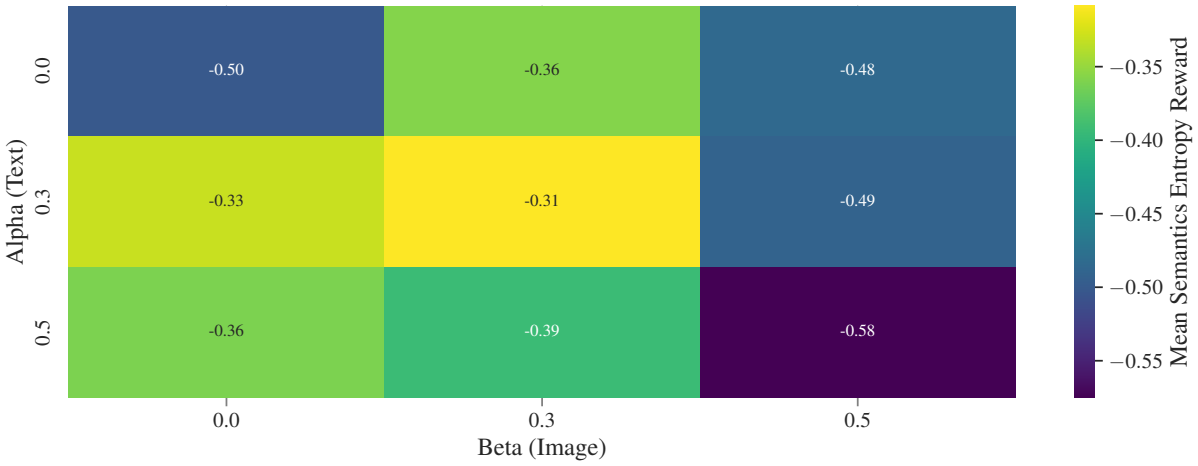


Figure 3.23: Performance of regularization strengths on semantics reward. 10 seeds were used.

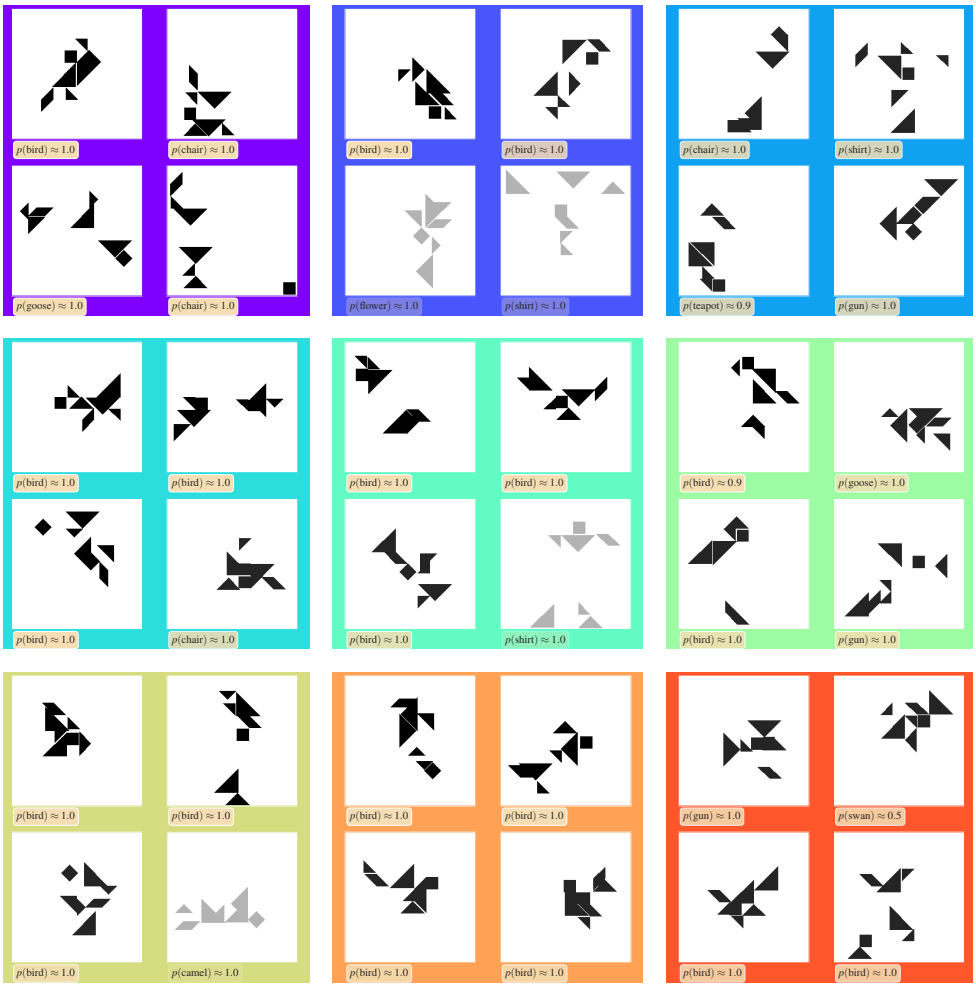


Figure 3.24: Samples of creations with different regularization strengths.

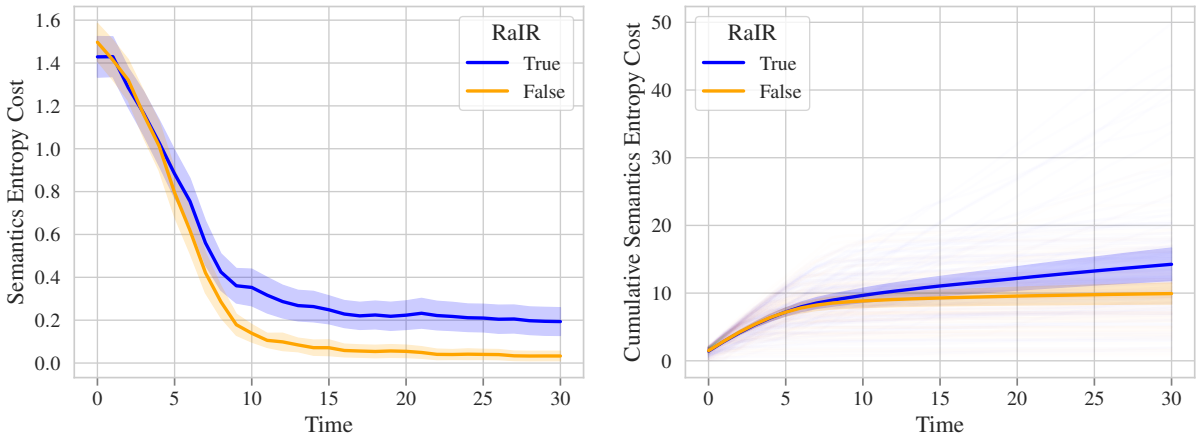


Figure 3.25: Effect of RaIR on semantics entropy reward in Tangram.

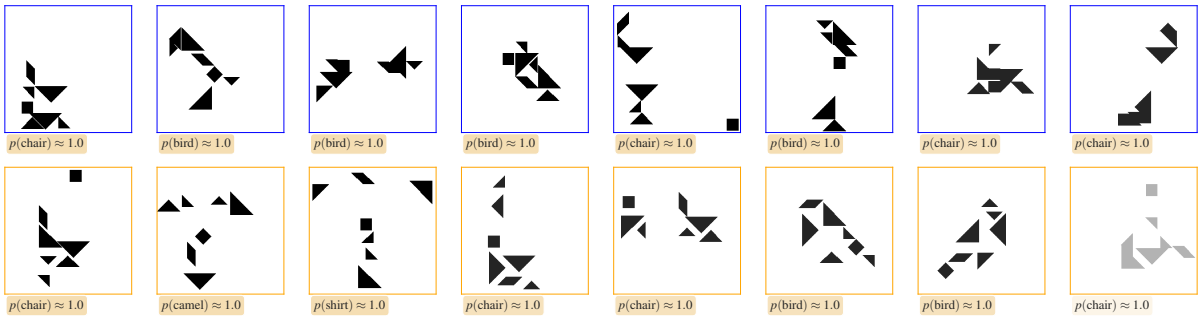


Figure 3.26: Samples of creations with and without RaIR.

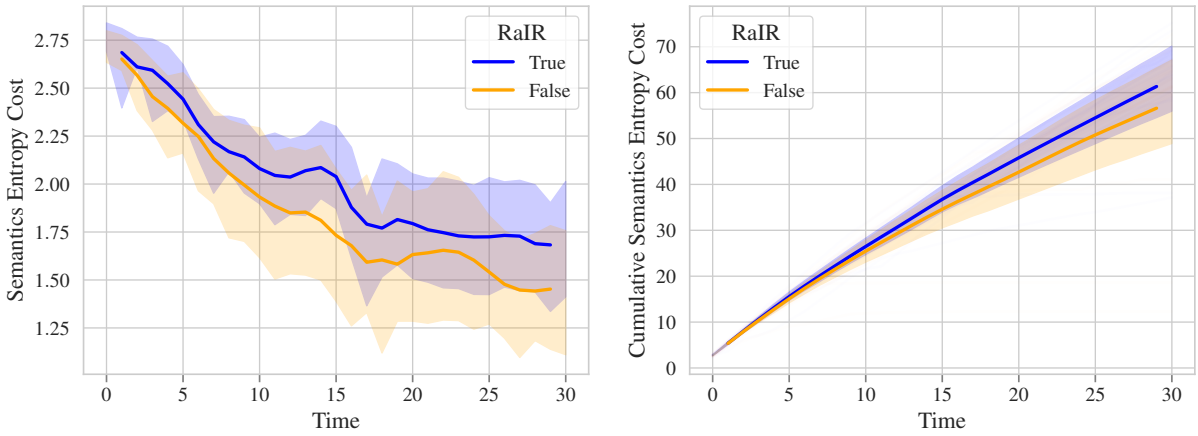


Figure 3.27: Effect of RaIR on ShapeGridWorld with grayscale pixels.

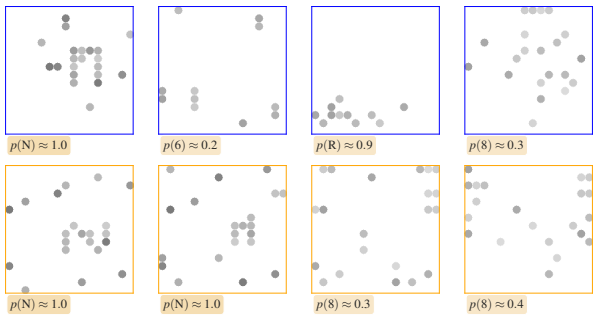


Figure 3.28: Samples of creations with and without RaIR in ShapeGridWorld with grayscale pixels.

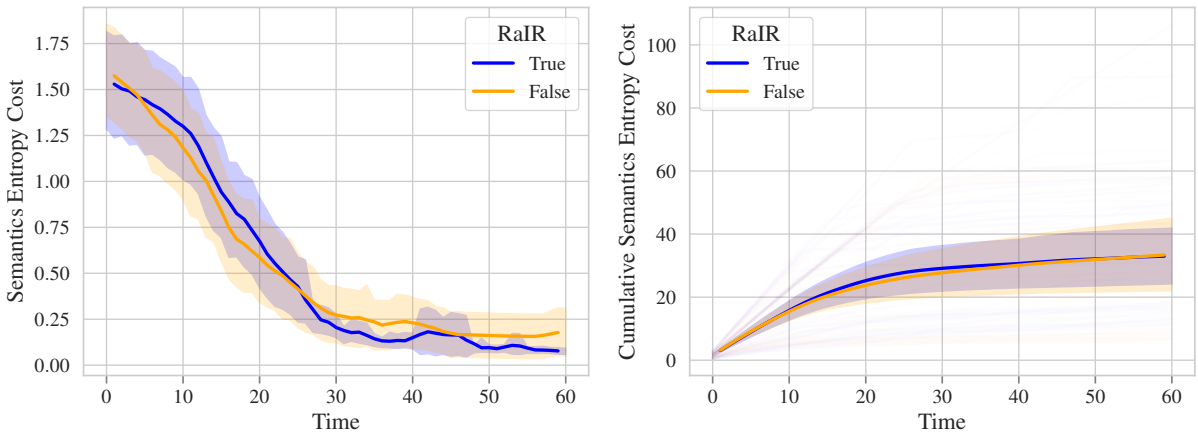


Figure 3.29: Effect of RaIR on ShapeGridWorld.

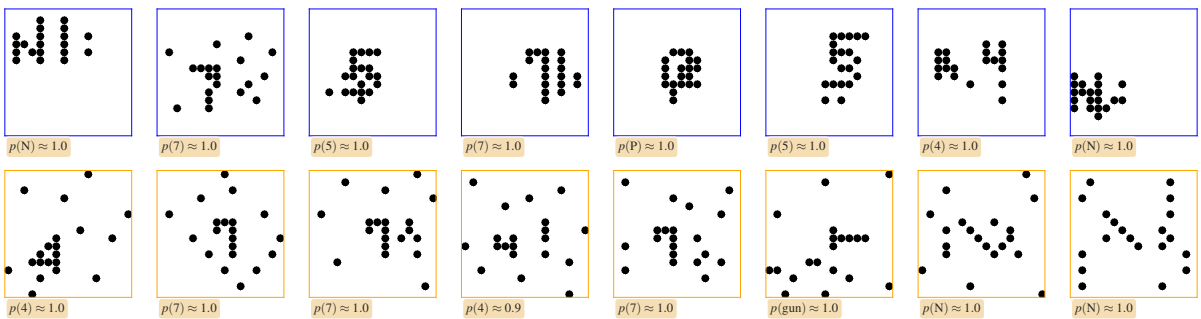


Figure 3.30: Samples of creations with and without RaIR in ShapeGridWorld.

Chapter 4

Discussion

Semantic expression is an innate nature of human exploration during creative free-play. Motivated by these observations from experiments in cognitive science and developmental psychology, in this thesis, we tried to leverage multimodal foundational models to realize this behavior in artificial agents. We formulated the semantics entropy reward to give the agent a freedom akin to free-play and enable it to reach semantically meaningful states.

To test this reward formulation in creative settings, we extended the ShapeGridWorld environment and created the Tangram environment. Subsequently, we showed the performance of the popular vision language model CLIP on these abstract settings and developed methods to improve its use for our purposes. We demonstrate that CLIP is noisy and consequently has problems such as sparse rewards and semantic biases in random images.

Thus we built upon recent work on using VLMs as a source of goal-conditioned rewards to formulate the regularized semantics entropy reward. We further showed how the different components of the revised reward formulation affect the reward trajectories with our initial experiments using the ersatz closeness reward trajectories in a post hoc simulation and adversarial analysis. Subsequently, we used insights from these experiments to guide the hyperparameter choice and search in the main experiments using the full reward.

We found that tweaking the regularization strengths with the temperature was enough to achieve good reward signals. Their effects are complementary but not quite the same. Regularization improved reward trajectories by aligning the reward function to semantically meaningful states in the CLIP embedding space which makes the reward signals more dense and a better guide to semantic expression. The temperature helps smoothen the reward trajectories and reduce the noise of CLIP in imperfect creations, thus reducing the semantic bias in random images. Moderate values of the regularization strengths $\sim (0.3, 0.3)$ and temperature somewhere between $0.01 - 0.02$ were found to be the best for the reward signal to guide the agent to semantically meaningful states, but slight variations in these values did not have a significant effect on the reward signal.

Other hyperparameters such as the use of negative embeddings, and tweaking the rendering function (adding texturing or modifying the images with other operations) had different effects on the reward signal, but we found them to be less important than the regularization strengths and temperature. Additionally, we show that the complementary RaIR reward helped improve the quality of the final creations in most conditions.

We used our insights into these method to successfully create a range of semantically expressive creations in both environments.

Although we were not able to overcome the class preference of CLIP and we got some categories such as "house" in Tangram much more than others. Even removing these categories just gave rise to other

categories that were preferred by CLIP. We think that these failures are related to capability limitations in the current VLM models. Perhaps using newer models from OpenCLIP such as `ViT-bigG-14` (Cherti et al., 2023) which are trained on the bigger LAION-5B dataset (Schuhmann et al., 2022) might lead to some performance improvements.

We also explored the possibility of fine-tuning CLIP with entropy regularization and re-training with random images to decrease its noise with our toy models, but did not scale it to the full CLIP model. We got some promising results with the toy models, and we think that this could be a good direction for future work.

Additionally, introducing a temporal dependency in the strengths of the regularity and semantics reward might lead to better results as RaIR might help start with the creation and bring it to a regime where the regularization methods of the semantics reward can be more effective in guiding it to a semantically meaningful state.

When used as an additional reward to augment traditional novelty-seeking intrinsic rewards $r_{\text{intrinsic}} = r_{\text{semantics}} + r_{\text{novelty}}$, the semantics entropy reward can lead to more human-like exploration i.e. curious behaviors with semantically-sound creative expression.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Cansu Sancaktar for her guidance throughout the project, and to Tankred Saanum for his advise.

I would also like to thank Georg Martius for giving me this opportunity and supporting me, the Max Planck Institute for Intelligent Systems for their resources, and the Graduate Training Center for Neuroscience at the University of Tübingen for their assistance.

Lastly, I thank my family, especially my parents, and my friends, for their emotional and intellectual motivation.

References

Adeniji, A., Xie, A., Sferrazza, C., Seo, Y., James, S., and Abbeel, P. (2023). Language reward modulation for pretraining reinforcement learning. *arXiv preprint arXiv:2308.12270*. 2

Baumli, K., Baveja, S., Behbahani, F., Chan, H., Comanici, G., Flennerhag, S., Gazeau, M., Holsheimer, K., Horgan, D., Laskin, M., et al. (2023). Vision-language models as a source of rewards. *arXiv preprint arXiv:2312.09187*. 2, 5, 21

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*. 1

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. (2023). Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829. 33

Chu, J. and Schulz, L. E. (2020). Play, curiosity, and cognition. *Annual Review of Developmental Psychology*, 2:317–343. 1

Cui, Y., Niekum, S., Gupta, A., Kumar, V., and Rajeswaran, A. (2022). Can foundation models perform zero-shot task specification for robot manipulation? In *Learning for dynamics and control conference*, pages 893–905. 2, 5

Diggs-Galligan, S., Chu, J., Tenenbaum, J., and Schulz, L. (2021). Explore, exploit, create: Inventing goals in play. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 43(43). iii, 1, 12

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 4

Gibson, E. J. (1988). Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. *Annual review of psychology*, 39(1):1–42. 1

Golomb, C. (1987). The development of compositional strategies in children's drawings. *Visual Arts Research*, pages 42–52. 2, 8

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 4

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95. 49

Itseez (2015). Open source computer vision library. https://github.com/itseez/opencv. 49

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. 48

Ladosz, P., Weng, L., Kim, M., and Oh, H. (2022). Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22. 2

Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N., Rocktäschel, T., and Grefenstette, E. (2022). Improving intrinsic exploration with language abstractions. *Advances in Neural Information Processing Systems*, 35:33947–33960. 2

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. 50

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *ICML*. 2

Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR. 2

Pinneri, C., Sawant, S., Blaes, S., Achterhold, J., Stueckler, J., Rolinek, M., and Martius, G. (2021). Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pages 1049–1065. PMLR. 3

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *arXiv (Cornell University)*. 3, 4

Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. (2023). Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*. 2, 6, 13, 25

Roth, K., Kim, J. M., Koepke, A., Vinyals, O., Schmid, C., and Akata, Z. (2023). Waffling around for performance: Visual classification with random words and broad concepts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15746–15757. 5, 22

Sancaktar, C., Piater, J., and Martius, G. (2023). Regularity as intrinsic reward for free play. 2, 8, 12, 38

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294. 33

Tam, A., Rabinowitz, N., Lampinen, A., Roy, N. A., Chan, S., Strouse, D., Wang, J., Banino, A., and Hill, F. (2022). Semantic exploration from language abstractions and pretrained representations. *Advances in neural information processing systems*, 35:25377–25389. 2

Umesh, P. (2012). Image processing in python. *CSI Communications*, 23. 39

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453. 49

Wang, H., Ge, S., Lipton, Z., and Xing, E. P. (2019). Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32. 13

Zhang, J., Huang, J., Jin, S., and Lu, S. (2024). Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 4

Zingrone, W. A. (2014). The construction of symmetry in children and adults. *Journal of genetic psychology/ The Journal of genetic psychology*, 175(2):91–104. 2, 8

Appendices

Appendix A

Environments' Details

The sections of this chapter provide additional details about the custom environments used in our experiments. Both of these environments were developed in Python.

A.1 ShapeGridWorld

Table A.1 lists the parameters of the original ShapeGridWorld environment developed by Sancaktar et al. (2023).

Property	Description
`width`	Width of the discrete grid.
`height`	Height of the discrete grid. Kept same as `width`.
`n_pixels`	Number of "On" pixels.
`shape`	Shape of a pixel block – "circle".
`size`	Size of a pixel block.
`persistency`	Number of time steps an object (pixel block) is moved.

Table A.1: Original ShapeGridWorld parameters.

The state space of this environment is composed of the x and y coordinates of the n block pixels, with the addition of two dimensions – one that specifies which object is currently in focus and another that specifies how many times it has already moved, i.e. $\mathcal{S} \in \mathbb{N}^{2n+2}$. The observation space is a rendering of the grid as an image of shape $\texttt{width} * \texttt{size} \times \texttt{height} * \texttt{size}$.

The action space comprises the action value for each of the directions (x and y) for an object; $\mathcal{A} \in [-1, 1]^{2n}$. For each dimension, the controller samples from a continuous distribution in $[-1, 1]$, which is uniformly mapped to $\{-1, 0, 1\}$ ($[-1, -1/3) \mapsto -1, [-1/3, 1/3] \mapsto 0, (1/3, 1] \mapsto 1$).

We further added more features to this environment for our experiments, which are listed in Table A.2. In particular, the ability to move all objects at once and more than one step in an action step was added. We also developed a controlled reset method with the added feature to freeze sections of the grid, with `control` and `control_boundaries`. This additionally enabled us to allow the controller only partial access to the environment. Furthermore, the rendering function was reimplemented using faster methods from *OpenCV*; see Section F.1 for more details.

If all objects are moved at once, the state space of this environment is composed of only the x and y coordinates of the block pixels, i.e. $\mathcal{S} \in \mathbb{N}^{2n}$. The corresponding action space in this case would be $\mathcal{A} \in [-1, 1]^{2n}$. For each dimension of the action space, the controller samples from a continuous distribution in $[-1, 1]$, which is uniformly mapped to integers $[-l, l]$, where $l \in \{\texttt{step\_x}, \texttt{step\_y}\}$ is the step size of the dimension.

Property	Description
`step_x`	Maximum number of steps an object can be translated in the x-direction in one action.
`step_y`	Maximum number of steps an object can be translated in the y-direction in one action. Kept same as `step_x`.
`persistency*`	Added the ability to move all objects at once.
`color`	A flag that indicates if pixels should have a grayscale value.
`invert`	A flag to control the inversion of the rendered images.
`control_boundaries`	If specified, objects inside these limits *initially* are marked.
`control`	A flag that indicates whether the controller can move all the objects or those marked initially by the control boundaries.
`max_reset_dist`	Maximum distance an object can be moved from its original position on reset. Only the objects marked initially are reset (default: 5).

Table A.2: Additional ShapeGridWorld parameters.

A.1.1 ShapeGridWorld Image Registration Technique

To test CLIP inference on ShapeGridWorld and simulate the controller on partial drawings, without having to draw these drawing samples manually, a registration method for images was developed that reads a given image to generate a corresponding ShapeGridWorld of given dimensions.

This is done using a circular convolution kernel over the image to find the corresponding grid pixel values. Optionally, it makes the lines in the image thinner by finding its skeleton using morphological operations before the convolution. See Fig. A.1 for a demonstration.

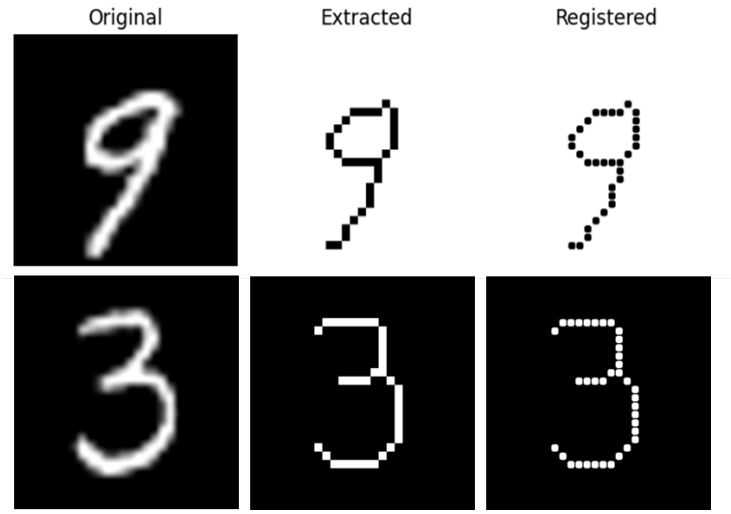


Figure A.1: ShapeGridWorld image registration on images from the MNIST dataset.

We developed a wrapper around the PIL Python library (Umesh, 2012) to conveniently perform these operations and generate ShapeGridWorld environments. The relevant code is hosted on https://github.com/pulkitgoyal56/ImageGrid. Table A.3 lists the relevant parameters of this image registration library.

Property	Description
`mode`	PIL Image mode in which the image is read.
`invert`	A flag to control the inversion of the read image.
`threshold_ratio`	Threshold on the convolution sum, for binary grids with no color.

Table A.3: Image registration parameters.

A.2 Tangram

The Tangram environment is initialized with a list of polygon objects created using the provided `Polygon` class, which defines a polygon's size, shape, and color. The additional parameters of the Tangram environment are tabulated in table Table A.4.

Property	Description
`x_size, y_size`	Span of the discrete grid; number of steps in the x and y directions. If set to 1, grid is continuous in $[0, 1]$ (default: `y_size` = `x_size`).
`x_step, y_step`	Maximum number of steps a polygon can be translated in the x and y directions in one action (default: `y_step` = `x_step`).
`r_size`	Span of the discrete grid; number of rotation steps in $180°$. If set to 1, rotation is continuous in $[0, 180°]$.
`r_step`	Maximum number of steps a polygon can be rotated in one action.
`rotate`	A flag to enable/disable rotation.
`flip`	A flag to enable/disable flipping.
`persistency`	Number of time steps a polygon (pixel block) is moved. If set to 1, all polygons are moved at once.
`control_boundaries`	If specified, polygons inside these limits *initially* are marked (default: NA).
`control_criteria`	The point inside the polygon that determines if the polygon is inside or outside the control boundaries. A Python lambda function or an attribute of `Polygon`, such as "centroid", "center", or "complete" (default: "center").
`control`	A flag that indicates whether the controller can move all the polygons or those marked initially by the control boundaries (default: "all").
`staging_boundaries`	If specified, only polygons inside these limits are rendered to get the state's corresponding image observation (default: NA).
`staging_criteria`	Similar to `control_criteria` but for staging boundaries (default: "complete").
`max_reset_dist`	Maximum distance a polygon can be moved from its original position on reset. Only the polygons marked initially are reset (default: -1; no constraints).

Table A.4: Tangram parameters.

The state space of this environment is composed of the x and y coordinates of the vertices of the $n = 7$ polygons, with the addition of two dimensions – one that specifies which polygon is currently in focus and another that specifies how many times it has already moved, i.e. $\mathcal{S} \in \mathbb{N}^{2+\sum_{i \in n} 2\ n_v(p_i)}$, where $n_v(p_i)$ is the number of vertices of polygon p_i. If all polygons are moved at once, the state space of this environment is composed of only the x and y coordinates of the block pixels, i.e. $\mathcal{S} \in \mathbb{N}^{\sum_{i \in n} 2\ n_v(p_i)}$. The observation space is a rendering of the grid whose resolution is controlled with different parameters.

The action space comprises the action value for each of the degrees of freedom (x, y, rotation, and flip) for an object; $\mathcal{A} \in [-1, 1]^4$. For each dimension of the action space, the controller samples from a continuous distribution in $[-1, 1]$. In the x-, y-, and rotation dimensions, this is uniformly mapped to integers in $[-l, l]$, where $l \in \{\texttt{step\_x}, \texttt{step\_y}, \texttt{step\_r}\}$ is the step size of the dimension, if it is discrete. If the dimension is continuous (because its `size` is set to 1), the action is scaled by its respective step size instead. For example, if `x_size` is 1 and `x_step` is 3, the resulting support of the action distribution for any action sampled for x-dimension will be $[-1/3, 1/3]$. For the flip dimension, the mapping is ($[-1, 0] \mapsto$ `False`, $(0, 1] \mapsto$ `True`).

The developed code to create Tangram-like environments with any number of arbitrary convex polygons is available on https://github.com/pulkitgoyal56/Tangram.

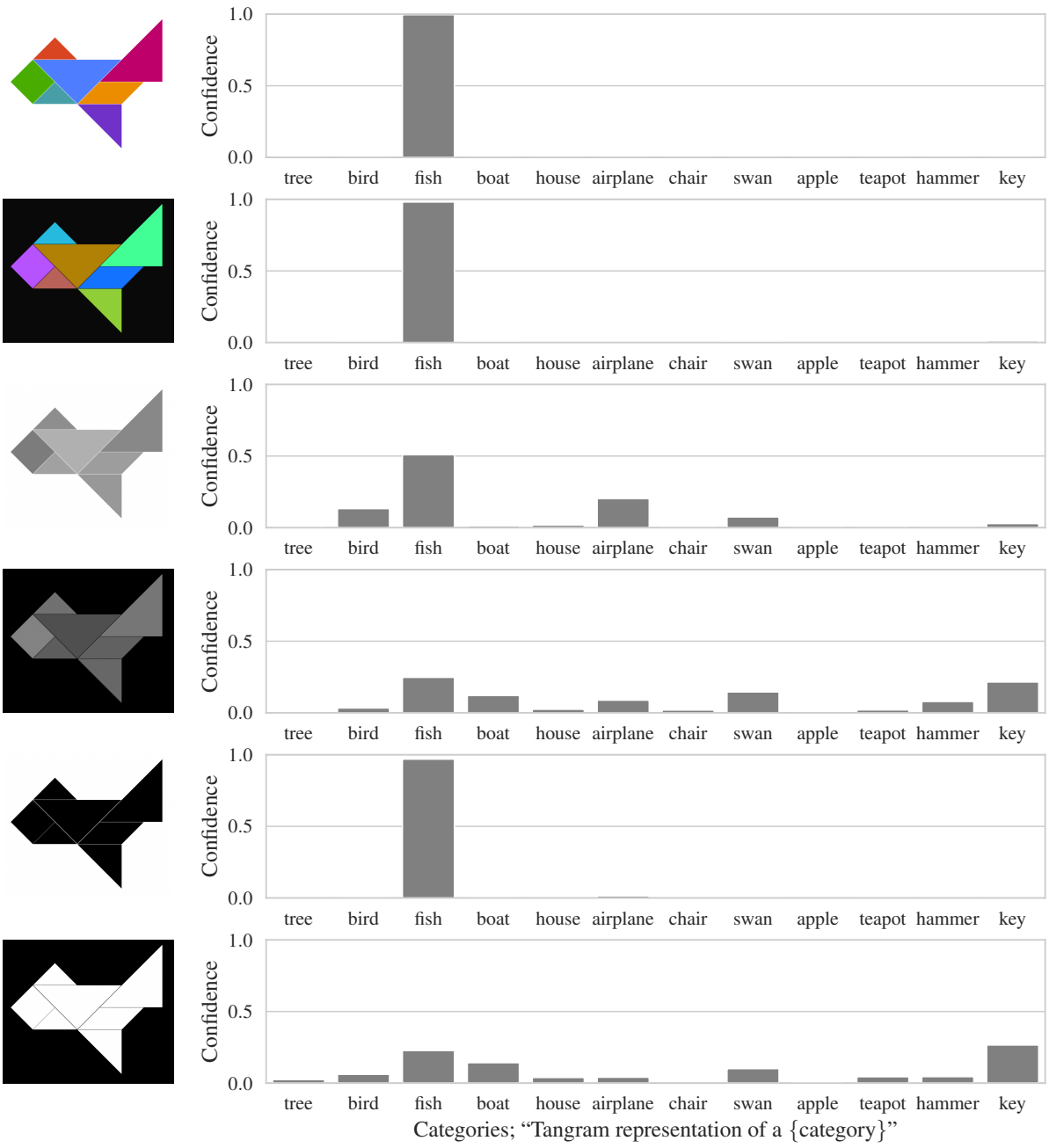


Figure A.2: CLIP on different inversions of the same Tangram creation. $T = 0.1$.

Appendix B

Controller Hyperparameters

The iCEM controller introduced in Section 2.4 has several hyperparameters that need to be tuned for optimal performance. Table B.1 lists the hyperparameters we tweaked in our experiments.

Property	Description
`horizon`	Number of steps in the future to plan, h.
`n_trajectories`	Number of trajectories to sample, n.
`cost_along_trajectory`	Cost/reward aggregation function to evaluate the trajectory, g. See Section 2.4.1 for explanations.
`n_inner_iterations`	Number of iterations for the inner optimization loop, m.
`init_std`	Standard deviation at which the sampling distribution is initialized, $\boldsymbol{\sigma}_0$.
`elites_size`	Size of the elite set, K. See eq. (2.36).
`discount_factor`	Discount factor for returns expected in the future, γ. See eq. (2.41) (default: 1.0; pink noise).

Table B.1: iCEM controller parameters.

The other important hyperparameters that we did not tweak in our experiments are listed in Table B.2.

Property	Description
`factor_decrease_num`	Factor by which population size is decreased over iterations (default: 1).
`alpha`	Momentum term for updating parameters of the sampling distribution.
`keep_previous_elites`	A flag to enable passing elites to the next timestamp (outer iteration).
`shift_elites_over_time`	A flag to enable passing elites over inner iterations.
`fraction_elites_reused`	Fraction of passed elites from the previous inner/outer iteration added to the sampled trajectory set.
`use_mean_actions`	If the mean of the sampling distribution is appended to the elite set at the end of inner iterations.
`noise_beta`	Colored noise exponent of the distribution, β (default: 1).

Table B.2: iCEM controller fixed parameters.

Please refer to the original paper for more details on the parameters of iCEM.

Appendix C

Reward Hyperparameters

This chapter lists the hyperparameters of the reward functions used in our experiments.

C.1 Semantics Reward

C.1.1 Semantics Entropy Reward

Table C.1 lists the parameters of the regularized semantics entropy reward from eq. (2.24).

Property	Description
`categories`	List of creative possibilities, $\boldsymbol{l}$.
`label_prefix`	Prefix to be added to the categories to create text labels.
`label_suffix`	Suffix to be added to the categories to create text labels.
`baseline`	Baseline text input, $\mathbf{l}_b$.
`image_baseline`	Baseline image input, $\mathbf{i}_b$.
`alpha_target`	Regularization parameter for target text, α.
`beta_image`	Regularization parameter for input image observation, β.
`model_version`	Name of the CLIP variant used.
`model_temperature`	Temperature for the softmax function in the semantics entropy reward, T.
`normalize`	If the reward should be normalized by maximum entropy, $\log(\lvert\boldsymbol{l}\rvert)$ (default: True).
`reward_scale`	Factor with which the reward is scaled. Together with the `reward_scale` for the regularity reward, this determines the reward ratio, λ from eq. (2.29) (default: 1).

Table C.1: Semantics entropy reward parameters.

C.1.2 Regularity Reward

In the introduction to the regularity reward in Section 2.3.2, we mentioned the additional hyperparameters h – *resolution* and *bidirectionality*. This section provides more details about these hyperparameters.

The hyperparameter for resolution consists of two parts – precision and granularity. Equation (C.1) extends eq. (2.28) with these hyperparameters to define the $\lfloor\cdot\rceil$ operator.

$$\lfloor s^{(j)} - s^{(k)} \rceil = \begin{cases} \left\lfloor \left| s^{(j)} - s^{(k)} \right| \frac{\text{precision}}{\text{granularity}} \right\rfloor \times \text{granularity} & \text{if bidirectionality} = \text{False} \\ \left\lfloor \left(s^{(j)} - s^{(k)} \right) \frac{\text{precision}}{\text{granularity}} \right\rfloor \times \text{granularity} & \text{if bidirectionality} = \text{True} \end{cases} \tag{C.1}$$

Table C.2 lists the parameters of the regularity reward.

Property	Description
`precision`	Precision of the compression algorithm.
`granularity`	Granularity of the compression algorithm.
`bidirectional`	If the compression should be bidirectional.
`normalize`	If the reward should be normalized by the maximum possible entropy, $\log\left(\binom{N}{2}\right)$ (default: True).
`reward_scale`	Factor with which the reward is scaled. Together with the `reward_scale` for the semantics entropy reward, this determines the reward ratio, λ from eq. (2.29).

Table C.2: Regularity reward parameters.

Please refer to the original paper for more details on the theory and implementation of *RaIR*.

C.2 Closeness Reward

Table C.3 lists the parameters of the closeness reward from Section 3.4.

Property	Description
`reward_type`	Type of closeness reward – *sparse-incremental* or *dense*.
`reward_scale`	Factor with which the reward is scaled.
`reward_threshold`	Threshold, ε, for sparse rewards, eq. (3.1).

Table C.3: Closeness reward parameters.

Appendix D

Comparing CLIP Models

The vision transformer (ViT) models have better accuracy than Resnet models. Resnet models have a gradual slope in rollouts compared to ViT models and flatter distributions on random images.

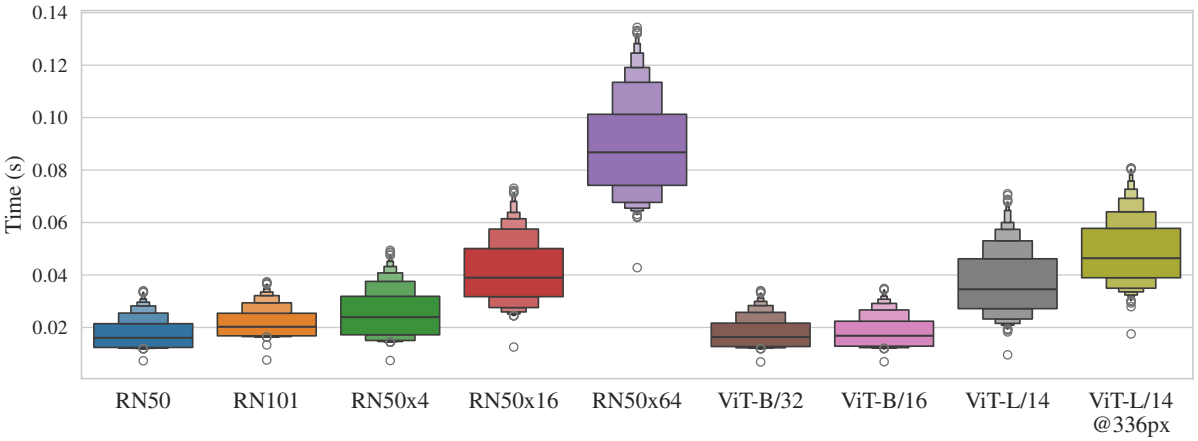


Figure D.1: Comparing times of different original CLIP models for embedding a single image.

Fig. D.2 shows the trajectories of different CLIP models on the Tangram environment.

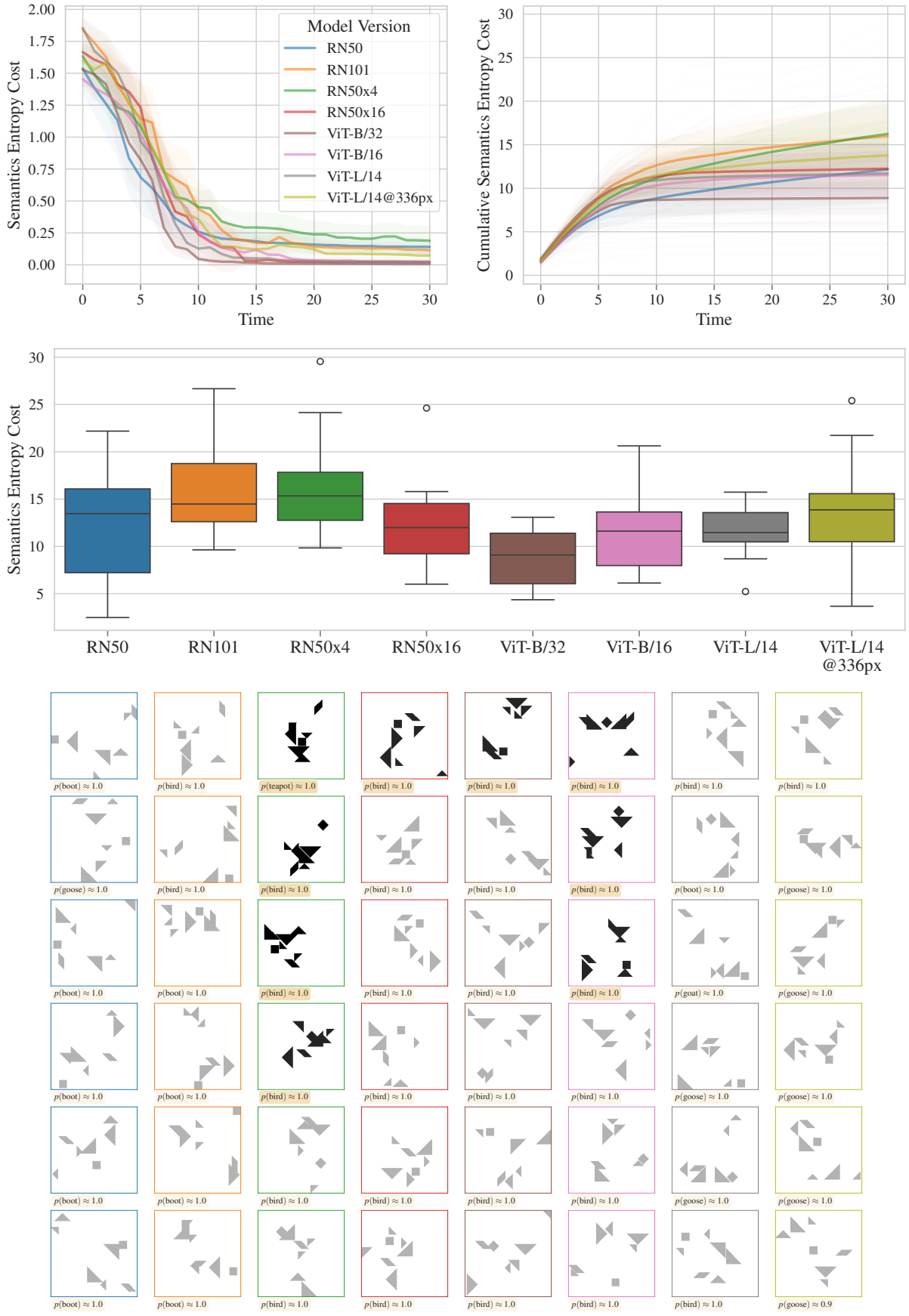


Figure D.2: Comparing reward trajectories of different original CLIP models on black and white Tangram environment. 10 seeds were used for each model.

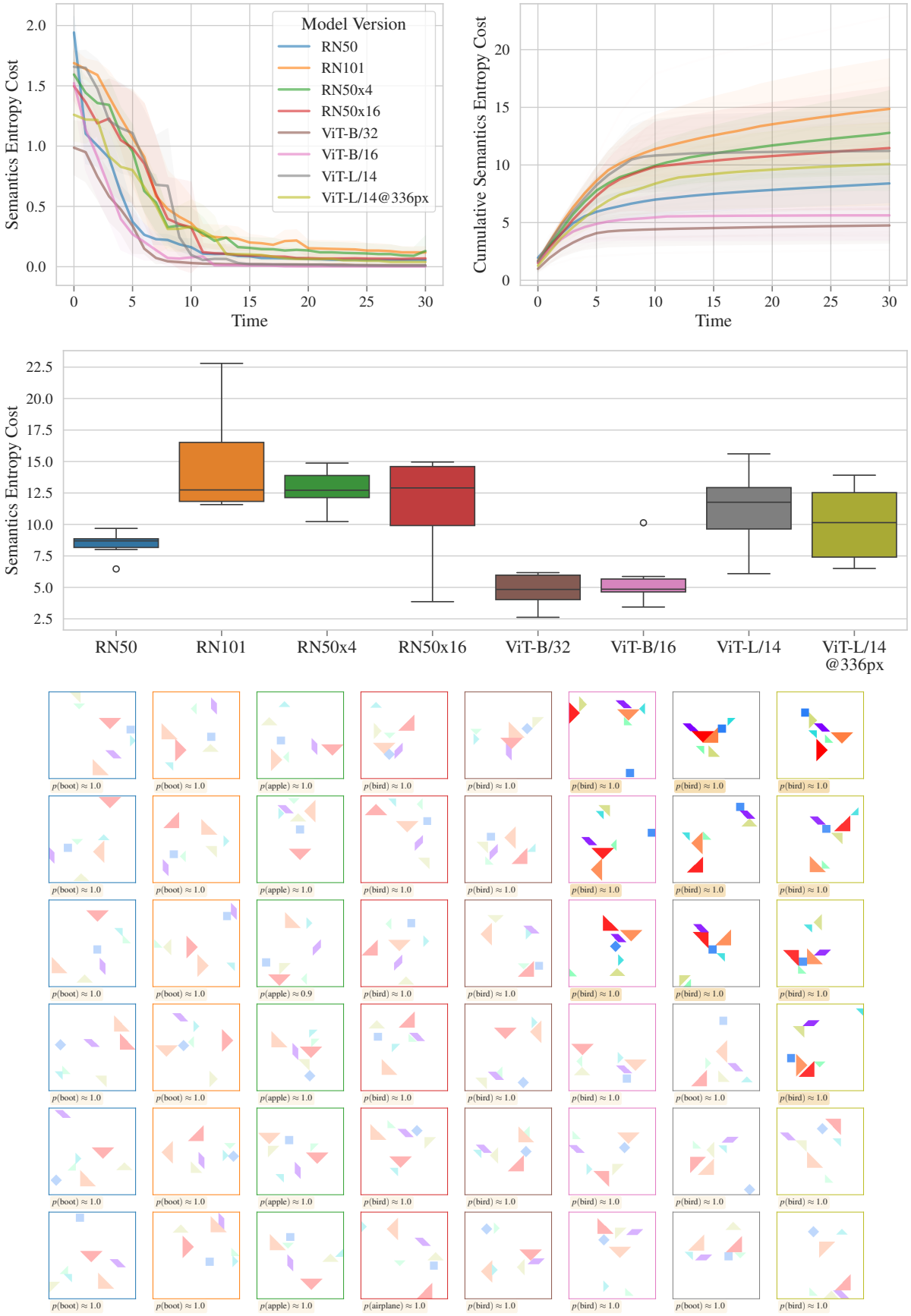


Figure D.3: Comparing reward trajectories of different original CLIP models on a colored Tangram environment. 10 seeds were used for each model.

Appendix E

Flatnet

To test the feasibility and efficacy of fine-tuning CLIP with an entropy regularization technique (eq. 3.2) to reduce its inference noise, we conducted experiments on a smaller toy setup with convolutional neural networks trained as a number classifier on ShapeGridWorld images using the MNIST dataset. We additionally augmented the MNIST training dataset with samples of images with random arrangements of pixels, labeled with a uniform distribution of confidence over the output of the model.

We treated the entropy regularization strength and the addition of random samples as hyperparameters and trained a series of models with different combinations of the two.

The models were trained with the Adam optimizer (Kingma and Ba, 2017) with a learning rate of 0.001 and a batch size of 128. Table E.1 summarizes the models we trained.

Serial	Name	Random Training Data Size	λ
Flatnet 0 (lite)	flatnetlite	0	0.0
Flatnet 1 (lite)	flatnetlite	0	0.2
Flatnet 2	flatnet	0	0.2
Flatnet 3	flatnet	0	0.4
Flatnet 4	flatnet	0	0.6
Flatnet 5	flatnet	0	0.8
Flatnet 6	flatnet	0	1.0
Flatnet 7	flatnet	10000	0.0
Flatnet 8	flatnet	10000	0.3
Flatnet 9	flatnet	10000	0.7
Flatnet 10	flatnet	10000	0.8
Flatnet 11	flatnet	10000	1.0
Flatnet 12	flatnet	0	1.0
Flatnet 13	flatnet	60000	0.0
Flatnet 15	flatnet	0	0.0

Table E.1: Summary of the Flatnet models.

Both, the entropy regularization and the addition of random samples, helped improve the reward trajectories of the models.

Appendix F

Improving Simulation Times

Rendering the environment and using CLIP for planning was a very resource-intensive operation. At every planning step, a total of n_trajectories $\times$ horizon $\times$ n_icem_inner_iterations evaluations needed to be computed. This was further multiplied by the number of steps in the simulation. This total ranged anywhere from $200,000$ to $2,000,000$ in our many experiments which corresponded to about 2 to 20 hours of simulation time if run in a single batch of inference on multiple GPUs.

Reducing simulation time was critical for us to be able to do any hyperparameter analysis. We implemented several optimization techniques to improve this, which mainly involved efficient *vectorization* of all computations, lazy evaluation, caching, parallelization, and cutting down on redundant operations. In particular, two of these, for rendering and inference, are briefly introduced in the following sections.

F.1 Reducing Rendering Time

The rendering time was a major bottleneck for the simulations. To improve this, we experimented with three popular open-source graphics Python libraries; *Matplotlib* (Hunter, 2007), *Scikit-Image* (van der Walt et al., 2014), and *OpenCV* (Itseez, 2015), for both colored and grayscale renderings. We used their functions in different formulations to further optimize their use. The results of the best formulation for each of these libraries are summarized in Table F.1 and Table F.2.

Library	**Main Function/Class**	**Time (μs)***
Matplotlib	`PatchCollection`	14400 ± 153
Scikit-Image	`draw.polygon`	978 ± 10.8
OpenCV	`fillPoly`	66.3 ± 0.85

Table F.1: Colored rendering-time comparison.

Library	**Main Function/Class**	**Time (μs)***
Matplotlib	`PatchCollection`	14200 ± 156
Scikit-Image	`draw.polygon`	978 ± 10.8
OpenCV	`fillConvexPoly`	52.2 ± 1.04

Table F.2: Grayscale rendering-time comparison.

* - Mean $\pm$ standard deviation of 7 runs; $1,000 - 10,000$ loops each.

These results refer to the Tangram environment, but the trend was the same with ShapeGridWorld. The complete analysis is available on https://github.com/pulkitgoyal56/master-thesis-notebooks/blob/main/archive/testbed_rendering.ignore.ipynb.

F.2 Reducing Inference Time

To minimize the inference times with CLIP, all inferences were done in one large batch. This required significant memory, for which we parallelized them on multiple GPUs using Pytorch `DataParallel` (Paszke et al., 2019). Depending on their size, the simulations were run concurrently on multiple NVIDIA V100 and A100 GPUs.

To reduce the inference time, we tweaked the image preprocessing functions of CLIP to be adaptive to the input to ensure that no redundant operations were performed.

Furthermore, we typically chose the rendering configurations for the environments such that the resulting renders required minimal preprocessing in the form of resizing, cropping, type conversions, or copying, which was computationally expensive. For example, all renderings concurred with the required input for the used CLIP model (eg. 224×224 for the `ViT-L/14` variant) used for inference to avoid any resizing. This also helped to improve the simulation times significantly.

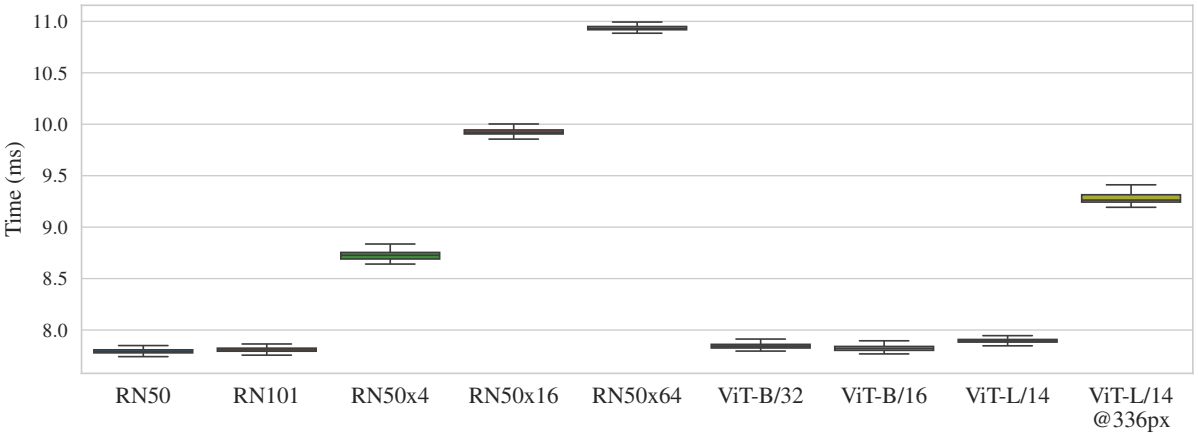


Figure F.1: Preprocessing times before optimizations.

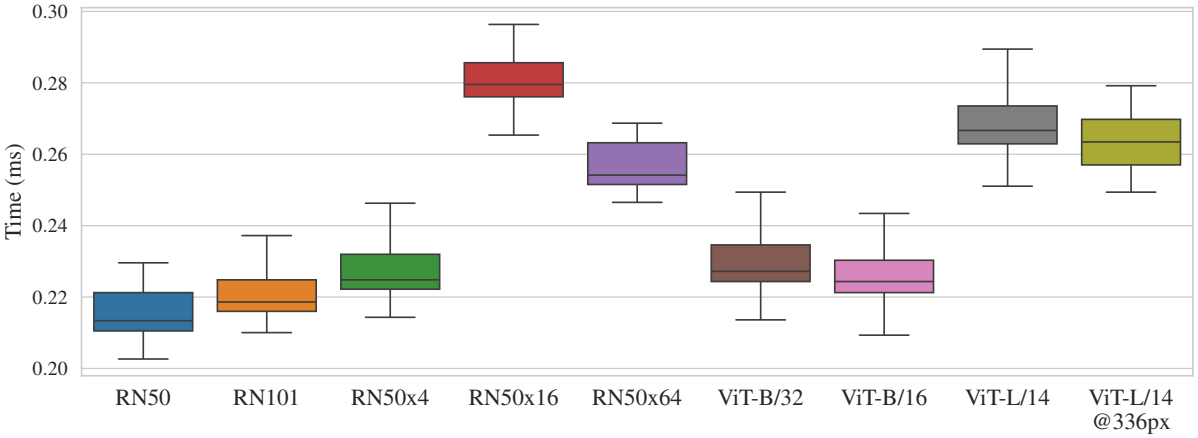


Figure F.2: Preprocessing times after optimizations.

These modifications are available on https://github.com/pulkitgoyal56/CLIP/tree/preprocess-tweaks.

Appendix G

Additional Simulations on ShapeGridWorld

The effect of the choice of categories on the semantics entropy reward in ShapeGridWorld was also significant.

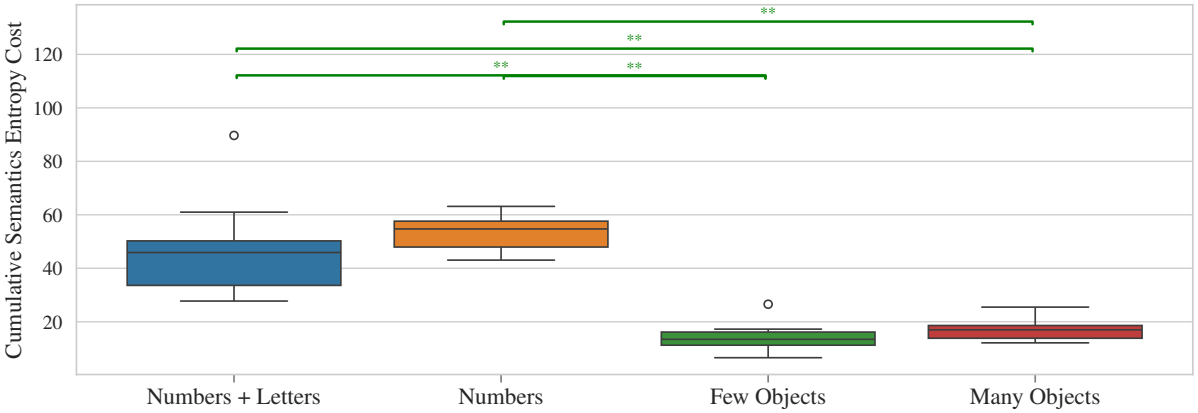


Figure G.1: Effect of categories on semantics entropy reward in ShapeGridWorld.

Fig. G.2 shows the effect of different categories on the reward landscape and Fig. G.3 shows some samples of the creations with different categories.

Observing the effect of the grayscale values on the effect of RaIR in ShapeGridWorld, we also ran simulations with different categories of colors in the pixels.

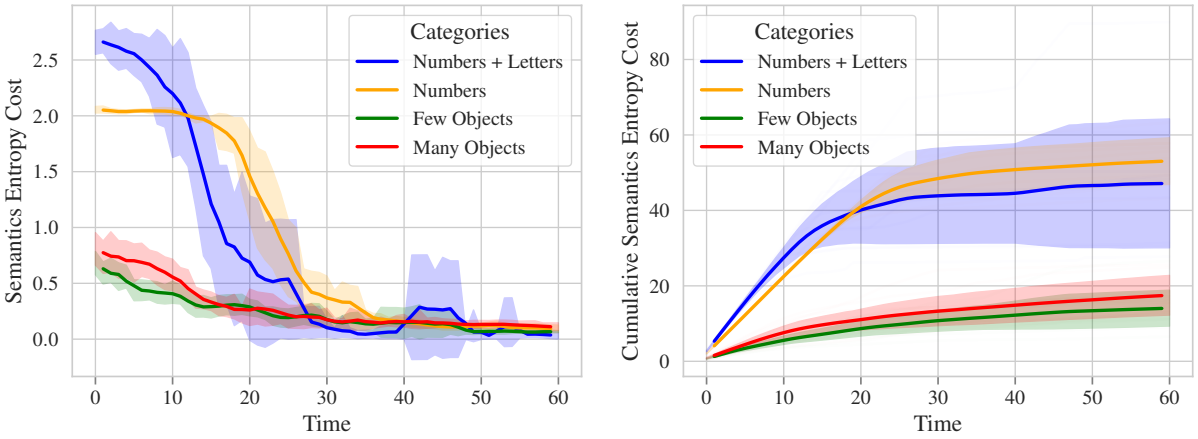


Figure G.2: Effect of categories on semantics entropy reward in ShapeGridWorld.

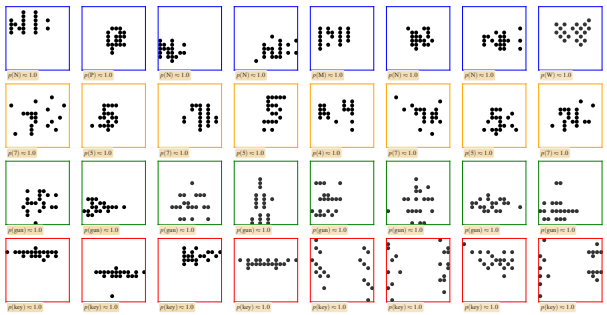


Figure G.3: Samples of creations with different categories in ShapeGridWorld.

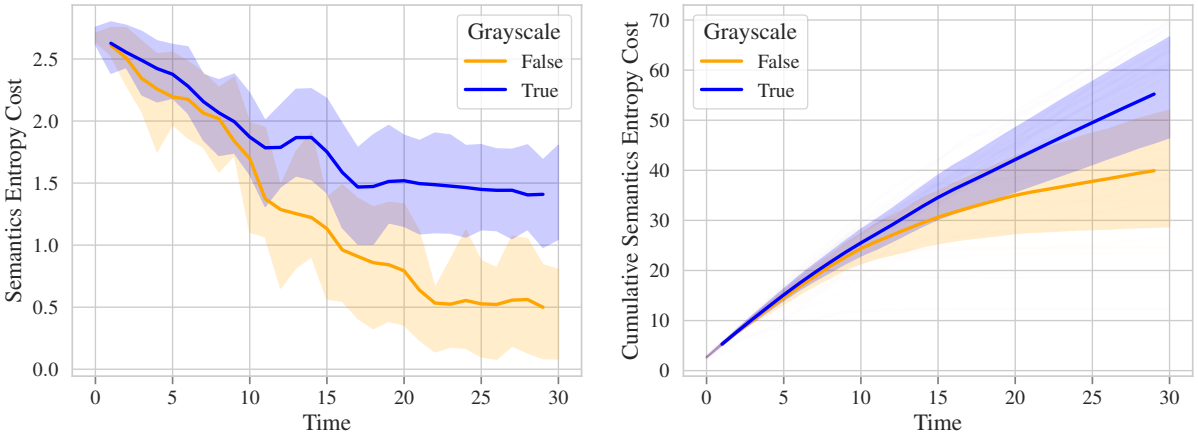


Figure G.4: Effect of the presence of grayscale pixels on semantics entropy reward in ShapeGridWorld.

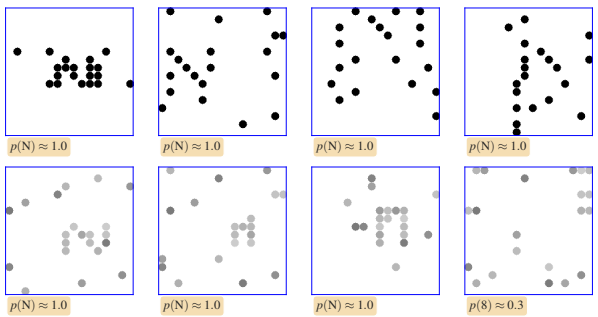


Figure G.5: Samples of creations with and without grayscale pixels in ShapeGridWorld.

Appendix H

Additional Graphs

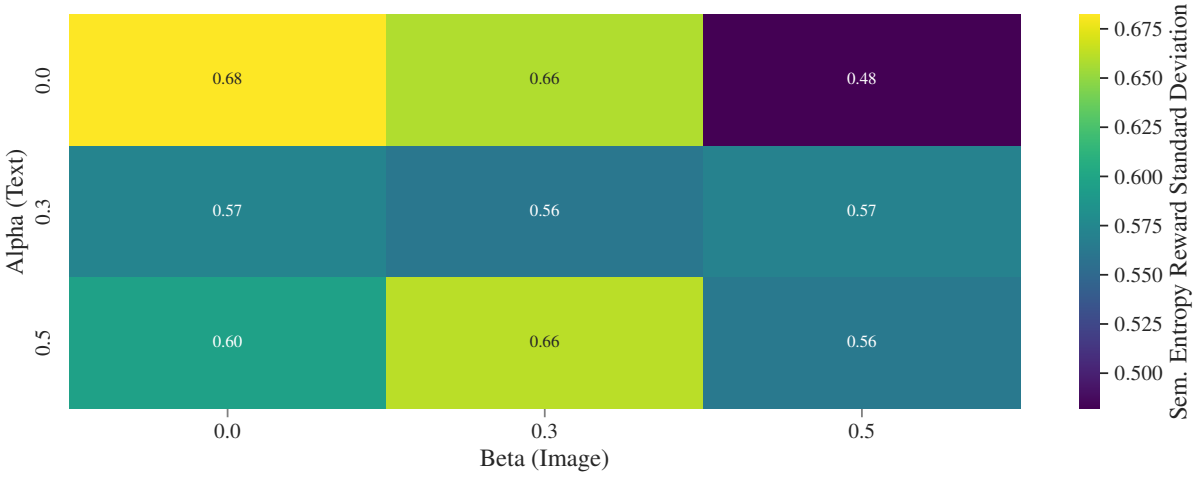


Figure H.1: Standard deviation of regularization strength performance on the semantics reward.

Appendix I

Curated Gallery

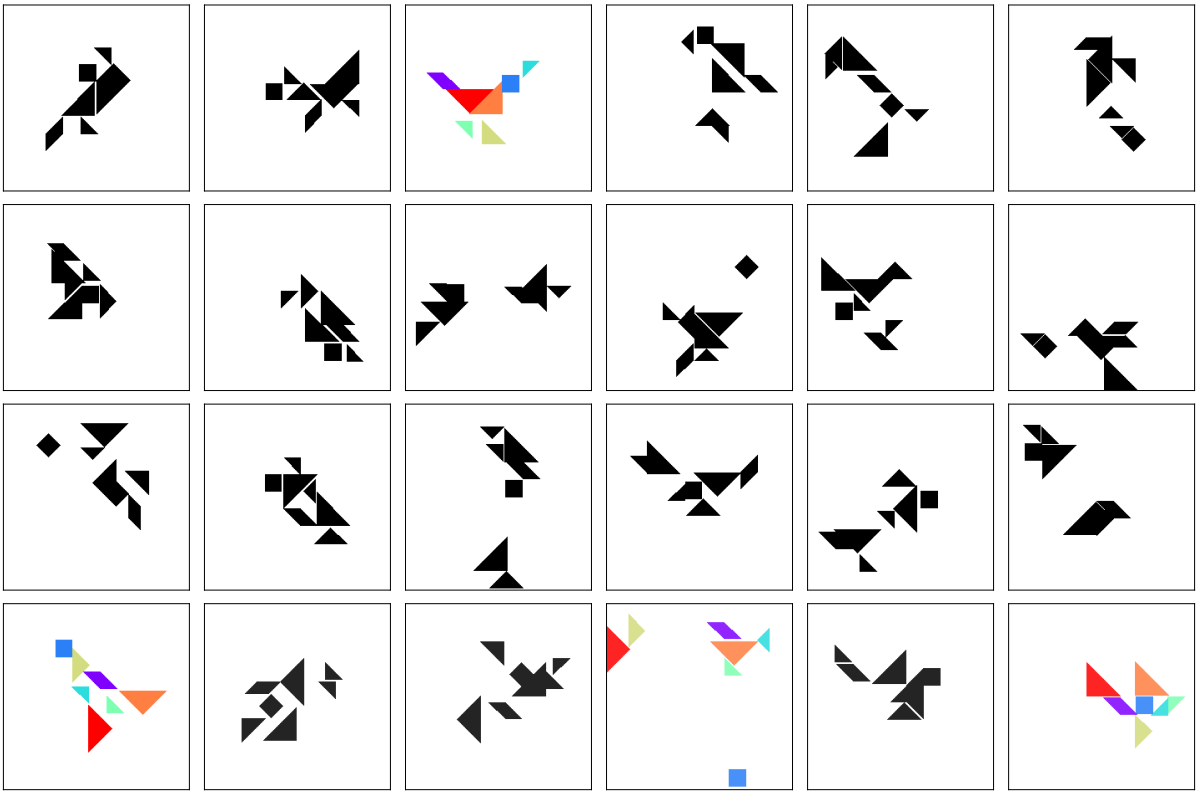


Figure I.1: Birds on Tangram.

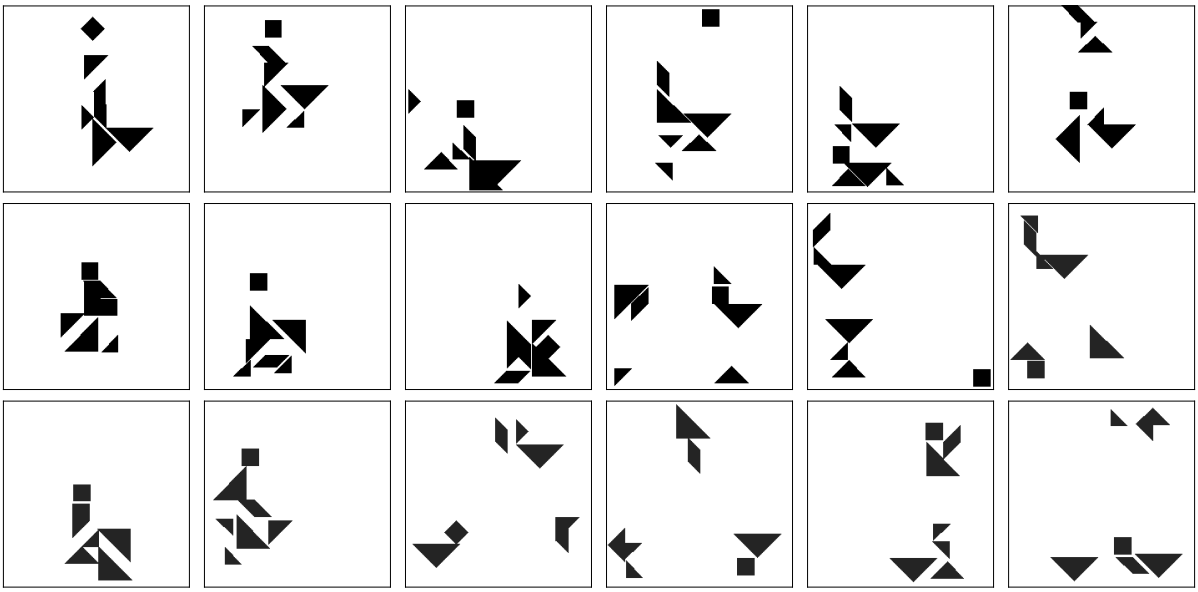


Figure I.2: Chairs on Tangram.

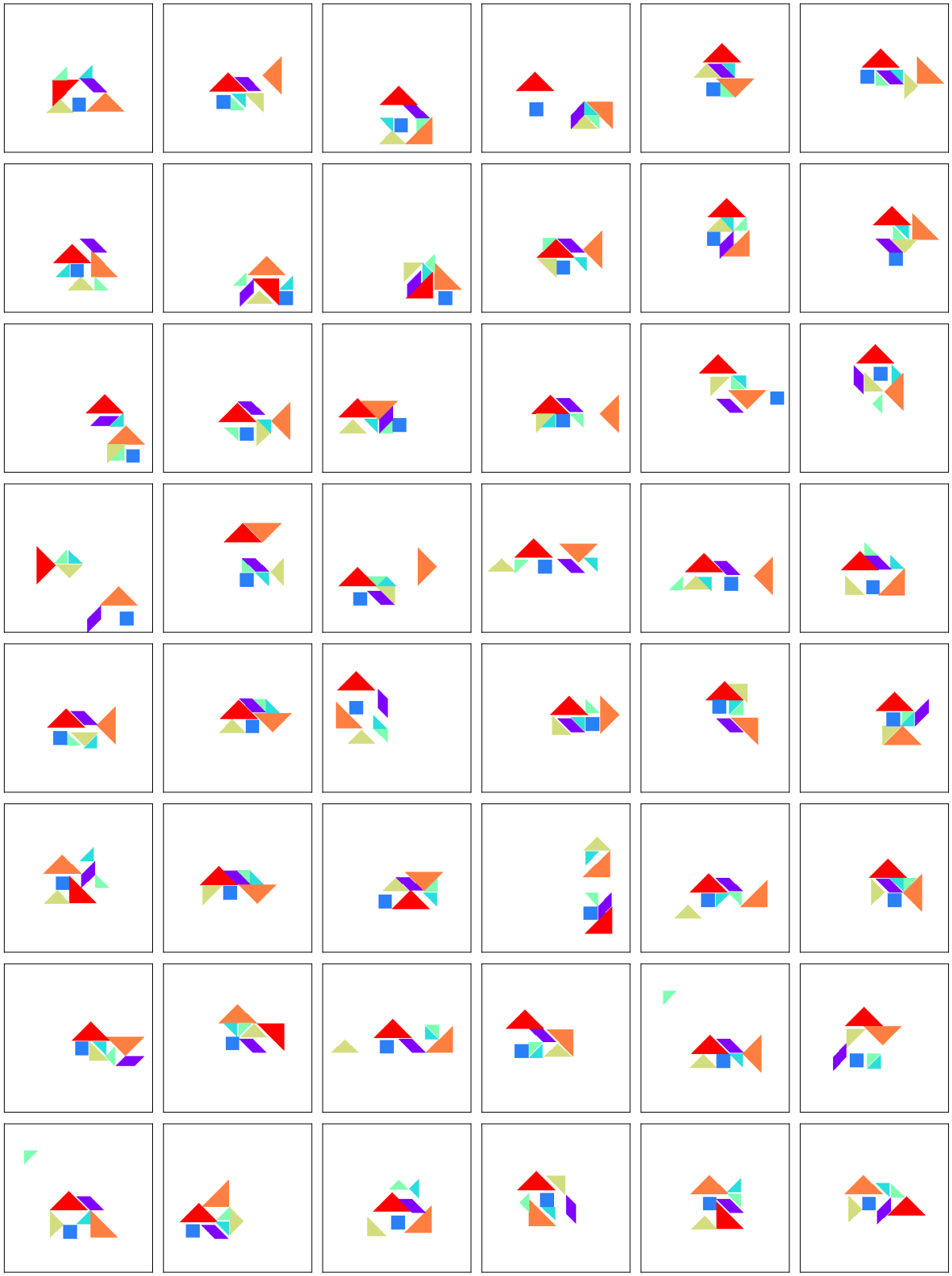


Figure I.3: Houses on Tangram.

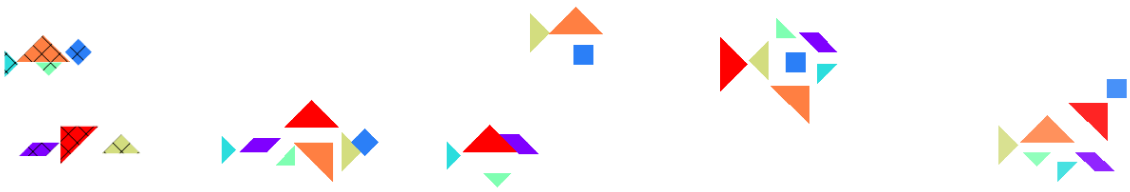


Figure I.4: Fishes on Tangram.

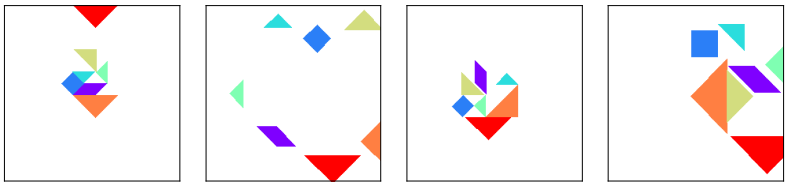


Figure I.5: Hearts on Tangram.

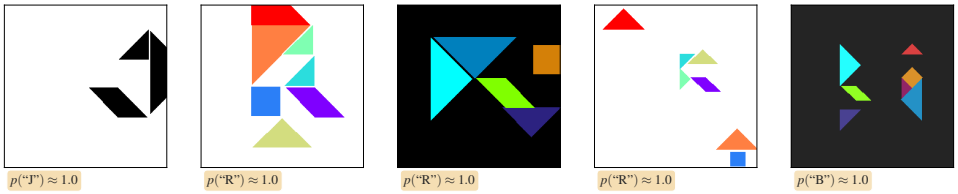


Figure I.6: Letters on Tangram.

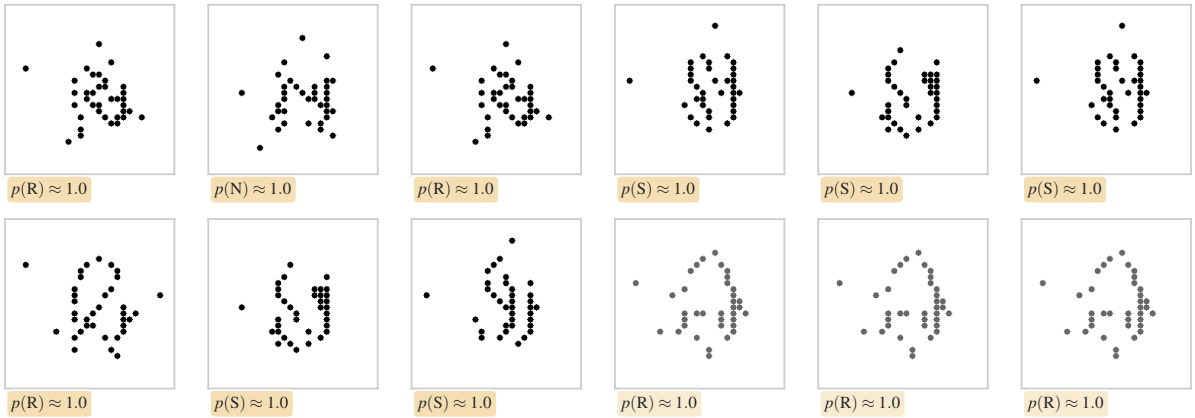


Figure I.7: Letters on ShapeGridWorld.