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 freely explore their environments in expressive manners that manifest their previous knowledge and understanding of the world. Particularly, we tend to exhibit this 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 prefer to create patterns and structures that are objectively meaningful to us, such as depictions of a house, a car, or a tree. In this study, we try to realize this emergent and semantically expressive behavior in artificial agents. To this end, we formulate an entropy-based intrinsic reward for self-supervised exploration during free play using large vision language models (VLMs). We use it in a model-based planning paradigm in two rich environments with building blocks as basic elements where we demonstrate that an agent optimizing its actions for these rewards can consistently create a range of diverse creative arrangements that are semantically expressive and reminiscent of real objects. To improve the reward quality, we experiment with different regularization methods and investigate if a complementary reward for regularity rooted in a similar bias towards symmetry and compression promotes semantic expression. This work improves upon and furthers the methods of using VLMs as a source of rewards and presents a novel perspective on imbuing emergent and creative behaviors in artificial intelligence.

Chapter 1

Introduction

Humans are capable of exploring and learning about the world in a self-supervised free-form manner without the presence of explicit predefined goals. Particularly observable in children, this is a process that is freely chosen and driven by intrinsic motivation, curiosity, and simply the desire to explore and experiment with the environment (Dietze et al., 2011).

The exact role of such play in human learning is debated across fields. Yet it has been established as a crucial component for the development of cognitive skills, acquisition of knowledge about the world, and adaptation to new environments (Anderson-McNamee and Bailey, 2010; Lai et al., 2018). As Chu and Schulz (2020) note, "...our ability to choose arbitrary costs and rewards allows us to pursue novel goals, discover unexpected information, and invent problems we would not otherwise encounter. Because (predefined) problems impose constraints on search, these invented problems may help solve a big problem: how to generate new ideas and plans in an otherwise infinite search space."

This process of *free play* in humans is also characterized by creativity, which is especially apparent in certain forms of play like construction activities. This involves imagination which allows children to create mental images related to their existing knowledge, feelings, thoughts, and ideas, which they then incorporate into their play (Bruce, 2011). Thus, free play is intertwined with visual perception and semantics (meaning), and our inherent preferences for visual semantics play a pivotal role in guiding our curiosity-driven exploration, i.e. we constantly seek to understand the semantics of the objects and scenes we encounter based on our prior experiences (Gibson, 1988).

In a study on free play conducted by Diggs-Galligan et al. (2021), in which they asked participants to freely interact with a grid of pixels with no predefined goal or external rewards, both adults and children showed a preference for semantic expression in the form of regular and symmetric patterns that depict real-world objects or concepts (see Fig. 1.1).

This furthers the idea that humans have an innate bias or intrinsic 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. This is related to "meaningful play" as defined by Zosh et al. (2017) and particularly present in children as they spend substantial time fiddling with toys, scribbling, and playing with building blocks, meanwhile creating patterns and arrangements that are meaningful to them.

Such play with meaning might serve as a mechanism for the brain to efficiently and quickly learn and adapt to new environments by linking new experiences to existing mental structures and frameworks, i.e. a form of transfer learning. Indeed, evidence from functional imaging studies in neuroscience shows that meaningful experiences present a blend of familiar and novel stimuli, and recruit a host of neural networks that are involved in novelty processing (medial orbitofrontal cortex), memory (hippocampus), and intrinsic rewards (striatum), which can together be forceful drivers of learning (Bunzeck et al., 2011) and building insight (Kizilirmak et al., 2016).

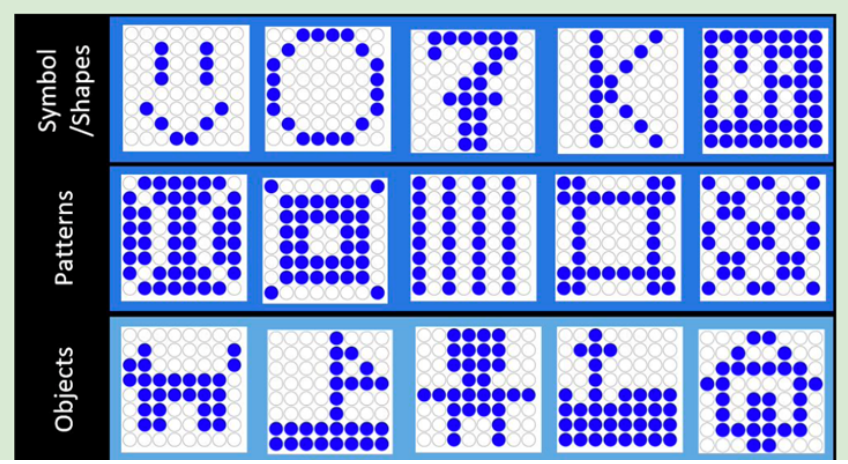


Figure 1.1: Some creations from the free-play study on humans by Diggs-Galligan et al. (2021). (Taken from the original publication.)

In the context of artificial intelligence, reinforcement learning (RL) has emerged as a powerful framework for training artificial agents capable of learning complex behaviors (policies) to solve tasks based on the pursuit of rewards through trial and error (Sutton and Barto, 2018). It has shown remarkable success in a wide range of environments. However, it can require a substantial amount of data to train and has yet to achieve a comparable quality of efficient learning and generalization that humans can achieve through exploration in high-dimensional and complex environments. Thus scalable and effective exploration remains a fundamental challenge in RL.

Among the various methods to model exploration in RL, formulations of *intrinsic rewards* (Ladosz et al., 2022) with curiosity-based (Pathak et al., 2017; Houthooft et al., 2016), novelty-based (Burda et al., 2018a; Zhang et al., 2021), or uncertainty-based (Burda et al., 2018b; Sekar et al., 2020; Pathak et al., 2019) methods have been proposed to encourage agents to explore their environment in a self-supervised manner. Yet these methods reward low-level exploration as they are designed explicitly to incentivize maximal information gain for improving internal models and policies. The precise novelties they promote might not be effectively (semantically) novel and the agent can be stuck in the pursuit of meaningless exploratory goals in an overwhelming search space.

This thesis considers the role of our visual semantic priors as abstractions in shaping exploration, embracing the idea that our knowledge and innate visual preferences act as a compass for more efficient and structured high-level exploration during free play by directing our attention and interactions toward semantic expression and discovery. We imbue artificial agents with this visual semantics bias that emerges and expresses itself automatically as the agent plays freely with the environment, akin to that in humans.

This is done by leveraging large vision language models (VLMs), which are deep neural networks trained on massive generic text and image datasets using the recent advances in self-supervised learning (Balestriero et al., 2023) and attention architectures (Dosovitskiy et al., 2020). The abstractions of natural language (Lupyan, 2012; Trott et al., 2023), which reflect the semantics of the world, allow them to learn efficient representations that encapsulate human visual understanding. Also termed as "foundation models", VLMs are capable of transferring to a diverse range of downstream applications beyond visual detection, classification, and object detection. For example, pre-trained vision-language encoders, such as CLIP (Radford et al., 2021), have successfully been used for image generation (Ramesh et al., 2022), decision making (Li et al., 2022), robot control (Shridhar et al., 2021; Khandelwal et al., 2022), and more (Zhang et al., 2024). Recent work (Cui et al., 2022; Baumli et al., 2023; Rocamonde et al., 2023; Adeniji et al., 2023) has shown that they can also be used as effective abstractions to generate zero-shot rewards for language-guided goal-conditioned tasks in reinforcement learning.

Additionally, VLMs have also been used to improve exploration as tools for refining intrinsic reward signals by abstracting away pseudo-novelty (Tam et al., 2022) or to pursue language-based internally-generated goals (Mu et al., 2022). Yet these approaches to abstracted exploration do not generate the rich and diverse creative behaviors characteristic of human exploration during play.

Our approach is fundamentally different from these existing studies as we are specifically interested in emergent and creative semantic expression. We use CLIP to generate exploratory intrinsic rewards that incentivize the agent to freely play in the environment and use its entities to express itself such that a meaningful creation emerges automatically. This intrinsic reward formulation is based on minimizing the entropy of a VLM's predictions over a set of creative possibilities that the environment offers. Although the range of freedom is limited to these possibilities, there is no explicit goal specification, and our controller can exploit this intrinsic entropy reward to guide the agent to any semantically expressive state that the VLM finds confidently meaningful.

We test our formulation in two rich creative environments: the traditional puzzle Tangram and a pixel grid similar to Fig. 1.1 and show the limitations of CLIP that pose challenges in shaping our reward. Subsequently, we introduce several modifications and regularization techniques adapted from previous work on using VLMs as a source of rewards in goal-conditioned tasks to mitigate these limitations and demonstrate their effects in independent ablation studies. These modifications help the agent to reach a meaningful state, keep it from getting stuck with imperfect creations (false positives or local optima), and improve its exploration to a diverse range of creative possibilities.

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

Our results show that the regularized semantics entropy reward with and without the additional reward for regularity is effective in consistently guiding the agent to discover a range of diverse semantically expressive states in both environments.

In the broader context, the work presented in this thesis improves upon and furthers the methods of using VLMs as a source of rewards in reinforcement learning and presents a novel perspective on imbuing human-like free-play behaviors in artificial intelligence with emergent exploration and free-form creativity.

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 $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \Theta, \Gamma, R, \mathbf{s}_0, \mathbf{o}_0)$, 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) $\Gamma : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, 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 $\mathbf{o}_0 = \Theta(\mathbf{s}_0)$.

A trajectory $\mathcal{T}_t = \{(\mathbf{s}_t, \mathbf{o}_t, r_t, \mathbf{a}_t), (\mathbf{s}_{t+1}, \mathbf{o}_{t+1}, r_{t+1}, \mathbf{a}_{t+1}), \ldots\}$ at time t is a sequence of observation-reward-action tuples starting from the current state $\mathbf{s}_t$, where $\mathbf{s}_k \in \mathcal{S}$, $\mathbf{o}_k \in \mathcal{O}$, $r_k \in \mathbb{R}$, and $\mathbf{a}_k \in \mathcal{A}$. In a typical reinforcement learning (RL) setting, the returns of such a trajectory $\mathcal{T}_t$ are the discounted sum of rewards given by,

$$\Upsilon(\mathcal{T}_t) = \sum_{k=0} \gamma^k r_{t+k}, \tag{2.1}$$

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 $\pi^* : \mathcal{S} \mapsto \mathcal{A}$ that maximizes its expected reward;

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

where the expectation is under trajectories *rolled-out* with $\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t)$, $\mathbf{s}_{t+1} = \Gamma(\mathbf{s}_t, \mathbf{a}_t)$, $\mathbf{o}_t = \Theta(\mathbf{s}_t)$, 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 optimizing the expected return over a limited planning horizon η.

Notation: We follow the following notation conventions throughout the text, unless explicitly noted otherwise. Scalars are denoted using lowercase letters, e.g. $x \in \mathbb{R}$, and scalar hyperparameters are specially denoted with lowercase Greek letters, eg. χ. Vectors are denoted with boldface letters, e.g. $\mathbf{x} \in \mathbb{R}^n$. Sets of vectors are denoted italicized boldface letters with a length indicator, e.g. $\boldsymbol{x}^{(n)} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The length indicator is omitted for readability at times. Calligraphic letters are used for higher vector sets, e.g. $\mathcal{X} = \{\boldsymbol{x}_1^{(n)}, \ldots, \boldsymbol{x}_m^{(n)}\}$, and other sets, including spaces $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{L}, \mathcal{V})$. Functions are denoted by uppercase letters, e.g. X, and those with scalar codomains are italicized, eg. X. This is also true for the reward function denoted by R whose value is denoted by r. This is not usually the case in RL literature (which denotes them just in the opposite way), but we use this notation to maintain consistency. However, to avoid radically diverging from the prevalent notations, there are two notable exceptions – π is used to denote the policy function and p for probability distributions.

2.2 Vision-Language Models

We define vision-language models (VLMs) 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 $\mathcal{V} \in [0, 1]^{n_v}$, which can thus be compared to each other using a vector similarity metric $\Lambda : \mathcal{I} \times \mathcal{L} \mapsto \mathbb{R}$, 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,

$$\Lambda(\mathbf{i}, \mathbf{l}) := S_c(\Psi_I(\mathbf{i}), \Psi_L(\mathbf{l})) = \frac{\Psi_I(\mathbf{i}) \cdot \Psi_L(\mathbf{l})}{\|\Psi_I(\mathbf{i})\| \|\Psi_L(\mathbf{l})\|} = \Psi_I(\mathbf{i}) \cdot \Psi_L(\mathbf{l}). \tag{2.3}$$

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, $\boldsymbol{\Lambda}^{(n)}(\boldsymbol{i}, \boldsymbol{l}) \in \mathbb{R}^{n \times n}$, where each item $\boldsymbol{\Lambda}^{(n)}_{j,k}$ 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 Chapter D of the appendix.

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 each 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,

$$P(\mathbf{l}_k; \mathbf{i}, \boldsymbol{l}, \tau) := \frac{\exp(\Lambda(\mathbf{i}, \mathbf{l}_k)/\tau)}{\sum_{\mathbf{l}_j \in \boldsymbol{l}} \exp(\Lambda(\mathbf{i}, \mathbf{l}_j)/\tau)}, \tag{2.4}$$

where $\tau \in \mathbb{R}^+$ 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 (2.5). The original paper suggests a value of 0.01 for most purposes, but we usually use values between the range $0.01 - 0.02$ for our experiments. See Section 3.4.3 for details.

Prompts for CLIP

Due to the inherent ambiguity in natural language, the exact phrasing can influence CLIP's inference. We experiment 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 `" and "`, on a white background`" are the prefix and suffix respectively.

2.3 Semantic Rewards for Free Play

This section describes the mathematical formulation of the intrinsic semantic reward that encourages 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 ($-H$) 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*,

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

Goal-Baseline Regularization

To improve the reward quality (shape), 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 (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, providing 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 improve 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,

$$R_{\text{goal-reg}}(\mathbf{o}; \mathbf{g}, \mathbf{g}_b, \gamma) := 1 - \frac{1}{2} \left\| \gamma \, \text{proj}_{\mathbf{v}} \Psi_I(\mathbf{o}) + (1-\gamma) \, \Psi_I(\mathbf{o}) - \Psi_L(\mathbf{g}) \right\|_2^2, \tag{2.8}$$

where

$$\mathbf{v} := \frac{\Psi_I(\mathbf{g}) - \Psi_L(\mathbf{g}_b)}{\|\Psi_I(\mathbf{g}) - \Psi_L(\mathbf{g}_b)\|}$$

is the direction spanned by the vector between the embeddings of $\mathbf{g}$ and $\mathbf{g}_b$,

$$\text{proj}_{\mathbf{v}} \Psi_I(\mathbf{o}) := (\mathbf{o} \cdot \mathbf{v}) \, \mathbf{v}$$

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[1].

For $\gamma = 0$, the original CLIP similarity metric from (2.3) is recovered as an unregularized goal-conditioned reward function after simplification (since all embeddings are unit vectors, i.e. $\|\Psi(\cdot)\|_2 = 1$);

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

For $\gamma = 1$, there is a trade-off of components of $\mathbf{o}$ orthogonal and parallel to "$\mathbf{g} - \mathbf{g}_b$".

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

Note that the magnitude of the projection $\|\text{proj}_{\mathbf{v}} \Psi_I(\mathbf{o})\|_2$ 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.

[1] We want to note here that the official implementation of VLM-RM available at https://github.com/AlignmentResearch/vlmrm is slightly different from the formulation given in the original paper, which is used here. Simplifying the other formulation leads to an additional term in the reduced form, but the overall essence of the reward from a conceptual standpoint still holds.

2.3.2 Regularized Semantics Entropy Reward

We combine these ideas below to derive a *baseline regularized semantics-entropy reward* from (2.5). Let $\boldsymbol{p} := \text{proj}_{\mathbf{v}}\Psi_I(\mathbf{o})$, $\boldsymbol{o} := \Psi_I(\mathbf{o})$, $\boldsymbol{g} := \Psi_L(\mathbf{g})$, $\boldsymbol{g}_b := \Psi_L(\mathbf{g}_b)$ 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 further, 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 in (2.24) 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 $\mathbf{l}_b \in \mathcal{L}$;

$$\hat{\Lambda}(\mathbf{i}, \mathbf{l}; \mathbf{i}_b, \mathbf{l}_b, \alpha_l, \beta_i) := S_c(((1-\beta_i)\;\Psi_I(\mathbf{i}) - \beta_i\;\Psi_I(\mathbf{i}_b)), ((1-\alpha_l)\;\Psi_L(\mathbf{l}) - \alpha_l\;\Psi_L(\mathbf{l}_b))), \tag{2.22}$$

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

The updated likelihood distribution $\hat{p}$ of the predictions of the VLM for the image observation $\mathbf{o} \in \mathcal{O}$ over the creative possibilities $\boldsymbol{c} \subset \mathcal{C}$ using this regularized similarity metric is then given by,

$$\hat{P}(\mathbf{c}_k \mid \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha_l, \beta_i, \tau) := \frac{\exp(\hat{\Lambda}(\mathbf{o}, \mathbf{c}_k; \mathbf{o}_b, \mathbf{c}_b, \alpha_l, \beta_i)/\tau)}{\sum_{\mathbf{c}_j \in \boldsymbol{c}} \exp(\hat{\Lambda}(\mathbf{o}, \mathbf{c}_k; \mathbf{o}_b, \mathbf{c}_b, \alpha_l, \beta_i)/\tau)}, \tag{2.23}$$

where $\mathbf{c}_k \in \boldsymbol{c}$. Using these definitions, the regularized semantics-entropy reward is given as,

$$
\begin{aligned}
R_{\text{sem-entropy-reg}}(\mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha_l, \beta_i, \tau) &:= -H(\hat{p}(\boldsymbol{c} \mid \mathbf{o}; \mathbf{o}_b, \mathbf{c}_b, \alpha_l, \beta_i, \tau)) && (2.24) \\
&= \sum_{\mathbf{c}_k \in \boldsymbol{c}} \hat{P}(\mathbf{c}_k \mid \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha_l, \beta_i, \tau) \log \hat{P}(\mathbf{c}_k \mid \mathbf{o}; \mathbf{o}_b, \boldsymbol{c}, \mathbf{c}_b, \alpha_l, \beta_i, \tau),
\end{aligned}
$$

where $\mathbf{c}_b$ is the text baseline, which could either be the initial state description or the negation of the goal.

2.3.3 Regularity Reward

We complement the semantic reward with regularity as an intrinsic reward (*RaIR*) (Sancaktar et al., 2023), which encourages the agent to create regular and symmetric patterns in its creations. 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, we consider that the state space can be factorized into a composition of the different (n_e) entities in the environment and the configuration of the agent, $\mathcal{S} = (\mathcal{S}_{\text{obj}})^{n_e} \times \mathcal{S}_{\text{agent}}$, where $\mathcal{S}_{\text{obj}} \in \mathbb{R}^d$; $d = 2$ in our environments. 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 $\mathbf{s}$,

$$R_{\text{RaIR}}(\mathbf{s};\omega) := -H(\Omega(\mathbf{s};\omega)) = \sum_{\mathcal{X}\in\Omega(\mathbf{s})} P(\mathcal{X};\omega)\log P(\mathcal{X};\omega), \tag{2.25}$$

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

$$\Omega : (\mathcal{S}_{obj})^{n_e} \mapsto \{\mathcal{X}_1^{M(\mathcal{X}_1)}, \mathcal{X}_2^{M(\mathcal{X}_2)}, \ldots\}, \tag{2.26}$$

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

$$P(\mathcal{X};\omega) = \frac{M(\mathcal{X})}{\sum_{\mathcal{X}'\in\Omega(\mathbf{s};\omega)} M(\mathcal{X}')}. \tag{2.27}$$

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;

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

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 of the appendix.

2.3.4 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 $R_{\text{semantics-entropy-reg}}$;

$$R_{\text{semantics}}(\mathbf{s},\mathbf{o};\mathbf{o}_b,\boldsymbol{c},\mathbf{c}_b,\alpha_l,\beta_i,\tau) = R_{\text{semantics-entropy-reg}}(\mathbf{o};\mathbf{o}_b,\boldsymbol{c},\mathbf{c}_b,\alpha_l,\beta_i,\tau) + \lambda\, R_{\text{RaIR}}(\mathbf{s}), \tag{2.29}$$

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

To be able to properly compare the two rewards, we constrain them to the same range $[0, 1]$ by normalizing them with their maximum possible entropy. The regularity reward is normalized by the maximum possible entropy of the multiset – $\log\left(\binom{n_e}{2}\right)$, which is equal to the entropy of a uniform distribution over the $\binom{n_e}{2}$ possible pairwise object distances of the n_e entities/objects in the state space, and 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 Γ 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 online, 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 η, iCEM uses a clipped colored-noise Gaussian distribution of dimension $n_a \times \eta$ over action sequences ($\boldsymbol{a}^{(\eta)} = \{\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{\eta-1}\}$), given by,

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

where $\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t \in \mathbb{R}^{n_a \times \eta}$ are the mean and standard deviation of the sampling distribution at time t respectively, and $\mathcal{C}^{\beta}(n_a, \eta)$ is a sampling function that returns n_a (one for each action dimension) sequences of length η 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 ($\nu$) set of action sequences ($\mathcal{A}_t = \{\boldsymbol{a}_1^{(\eta)}, \boldsymbol{a}_2^{(\eta)}, \ldots, \boldsymbol{a}_\nu^{(\eta)}\}$) is sampled from this distribution. These samples are augmented with a fraction ζ 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 $i = 0$; this is denoted by $\zeta \cdot \mathcal{E}_t^{i-1}$ in (2.31). Additionally, to ensure that the mean action of the distribution is accessible to the controller, in the last inner iteration ι, the mean of the distribution $\boldsymbol{\mu}_t$ is added to the elite-set $\mathcal{A}^+$;

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

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

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

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

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

$$\mathcal{R}_t = \{\{r_{t+1}, r_{t+2}, \ldots, r_{t+\eta}\}\}_{j=1}^{\nu}, \tag{2.34}$$

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 intrinsic rewards. The corresponding trajectories are,

$$\mathcal{T}_t^{(\nu \times \eta)} = \{\{(\mathbf{s}_t, \mathbf{o}_t, r_t, \mathbf{a}_t), \ldots, (\mathbf{s}_{t+\eta-1}, \mathbf{o}_{t+\eta-1}, r_{t+\eta-1}, \mathbf{a}_{t+\eta-1}), (\mathbf{s}_{t+\eta}, \mathbf{o}_{t+\eta}, r_{t+\eta}, \varnothing)\}\}_{j=1}^{\nu}. \tag{2.35}$$

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

$$\mathcal{E}_t^i = \kappa \cdot \mathcal{A}_t^+ = \{\boldsymbol{a}_k^{(\eta)} \in \mathcal{A}_t^+ : | \; \boldsymbol{a}_j^{(\eta)} \in \mathcal{A}_t^+ : \Upsilon(\mathcal{T}_k^{(\eta)}) \leq \Upsilon(\mathcal{T}_j^{(\eta)}) \; | \leq \kappa\}. \tag{2.36}$$

Best κ 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.

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

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

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

The above steps are repeated for ι 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;

$$\boldsymbol{a}^{(\eta)*} = \underset{\boldsymbol{a}_k^{(\eta)} \in \mathcal{A}^+}{\operatorname{argmax}} \; \Upsilon(\mathcal{T}_k^{(\eta)}). \tag{2.39}$$

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

Since the standard deviation in iCEM adapts according to the elite-set statistics, when the controller is close to an optimum, the standard deviation automatically decreases, narrowing down the search and fine-tuning the solution. Therefore it is sufficient to sample fewer action sequences as the CEM iterations proceed. Thus, the population size is exponentially decayed over time with factor δ. At timestep t, it is equal to $max(\nu\delta^{-t}, 2\kappa)$, where the max ensures that the population size is at least double the size of the elite-set.

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 in sparse reward scenarios, as in our case (because of noisy CLIP rewards; discussed later in Section 3.2.2).

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 noisy nature of our semantics entropy reward by effectively regularizing the reward landscape. Furthermore, the addition of memory gives it good sample efficiency, which is 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 Γ 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 run simulations using ground truth (GT) models, i.e. with access to the true simulator of the environment itself. Thus, we can perform multi-horizon planning without accumulating any model errors, which allows us to better investigate the shape and global/local optima of our semantics reward landscape.

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`),

$$\texttt{sum}(\mathcal{T}^{(\eta)}) = \sum_{k=1}^{\eta} r_k. \tag{2.40}$$

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

$$\texttt{sum-}\gamma(\mathcal{T}^{(\eta)}) = \sum_{k=1}^{\eta} \gamma^{k-1} r_k. \tag{2.41}$$

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;

$$\texttt{best}(\mathcal{T}^{(\eta)}) = \max_{k=1}^{\eta} r_k. \tag{2.42}$$

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;

$$\texttt{best-}l(\mathcal{T}^{(\eta)}) = \max_{k=1}^{\eta-l+1} \sum_{j=k}^{k+l-1} r_{k+j}. \tag{2.43}$$

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;

$$\texttt{last}(\mathcal{T}^{(\eta)}) = r_h. \tag{2.44}$$

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;

$$\texttt{last-}l(\mathcal{T}^{(\eta)}) = \sum_{k=\eta-l+1}^{\eta} r_k. \tag{2.45}$$

Note that we often use the term *cost* ($-1 \times r$) instead of reward in our results and discussions. This is because the iCEM controller implementation we use minimizes costs instead of maximizing rewards, which are essentially equivalent and only differ in their signs.

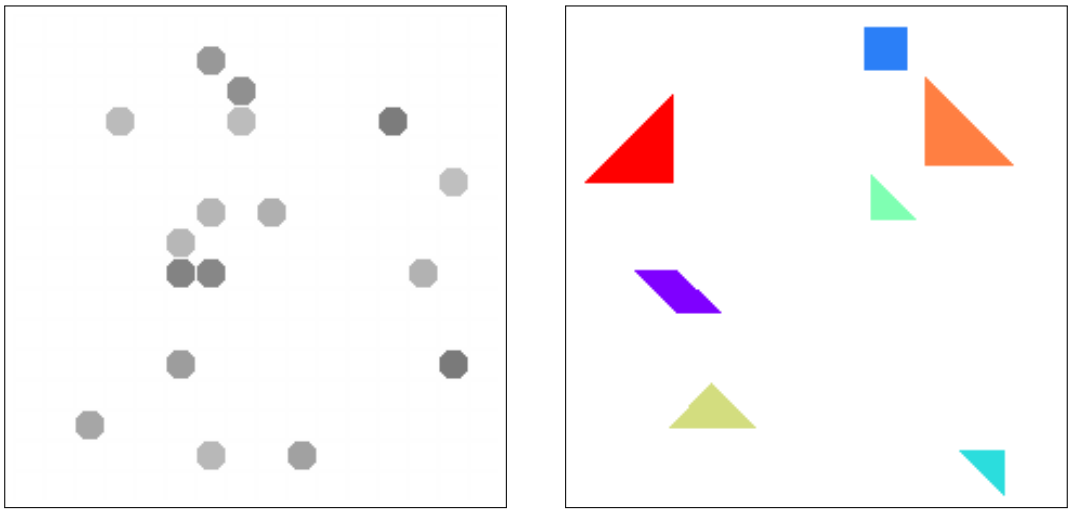
2.5 Experiment Setup

The simulations are run concurrently on multiple NVIDIA V100 and A100 GPUs, depending on the size of the simulations, which is determined by the size of the CLIP model used, the number of trajectory samples in iCEM, and the number of planning horizons. To minimize the time of a rollout, all $\eta \times \nu$ CLIP inferences for planning at an inner iCEM iteration are performed in a single batch in parallel. Environment rendering and simulations are multi-processed on CPU in parallel.

Several optimizations were put into place to improve the simulation time, particularly in environment rendering and CLIP preprocessing. Please refer to Chapter F of the appendix for more details.

2.6 Environments

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



(a) ShapeGridWorld (b) Tangram

Figure 2.1: The environments.

2.6.1 ShapeGridWorld

ShapeGridWorld (SGW) 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 is adapted from the pixel grid environment used in the work by Sancaktar et al. (2023) with some modifications summarised in A.1 of the appendix.

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.

2.6.2 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. Because of its reduced degrees of freedom compared to ShapeGridWorld, it mitigates the problems discussed in sections Section 3.2.1 and Section 3.2.2. It is a major contribution of our work.

We mostly configured the environment to correspond with the traditional Tangram game. The shapes involved are the same, no overlap is allowed, and the shapes could be rotated, flipped, and translated. Yet, we conducted experiments with colors to improve the performance of CLIP and also tested giving the agent the freedom to use a subset of the shapes to make its creations.

Please refer to Section A.2 of the appendix for more technical details about this environment.

Chapter 3

Experiments

3.1 Initial Tests with CLIP

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

3.1.1 CLIP on Sketches and ShapeGridWorld

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

We also compare these results with renderings of these sketches on ShapeGridWorld grids of different resolutions by registering them to a grid (see Section A.1.1 of the appendix for more details on registration). For an example, see Fig. 3.1. We find that CLIP can recognize these sketches quite well. As the grid becomes coarser, the confidence and accuracy of CLIP decrease significantly. For very coarse grids, the accuracy is almost as bad as random guessing.

We also experiment with inverting the images and find that the results are consistently slightly better with black-on-white sketches (shown here) than with white-on-black sketches. Yet, we do not notice any difference in performance with the addition of grayscale values in the pixel blocks in these experiments.

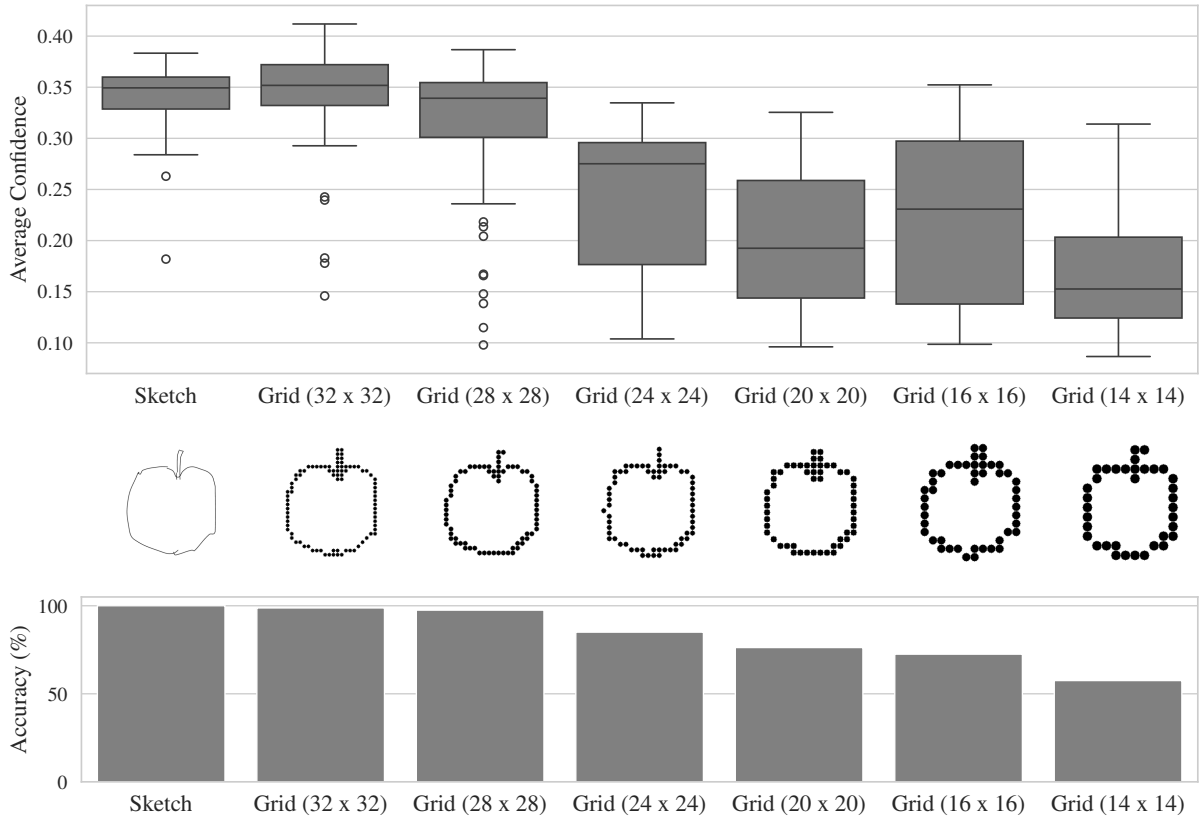


Figure 3.1: CLIP on sketches and ShapeGridWorld grids of different resolutions. The first row shows the comparative similarity of the embeddings of 80 sketches of apples (and their grid counterparts) to the label "`sketch of an apple`" using the CLIP variant `ViT-L/14`. The probabilities are calculated using (2.4), with temperature $\tau = 1$, 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.

In most cases, we also notice that using specific labels like "`apple`" for a picture of an apple has better accuracy than using hypernyms like "`fruit`" (Fig. 3.2).

Additionally, we find that the bigger CLIP models perform better on these tasks than the smaller ones which is in line with observations from other studies on VLMs as a source of rewards, particularly Rocamonde et al. (2023). We tried the same experiments with different sets of categories, prefixes, and suffixes; the results followed similar trends.

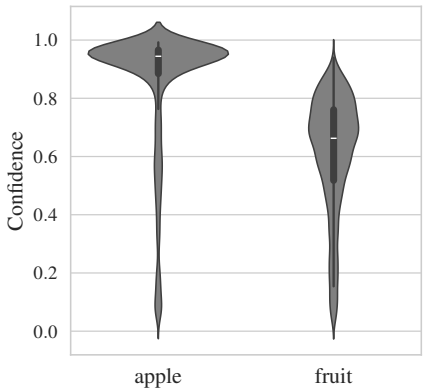


Figure 3.2: CLIP on different levels of hypernymy and hyponymy of labels.

3.1.2 CLIP on Tangram

Tangram has fewer degrees of freedom than ShapeGridWorld. 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 test CLIP's accuracy as a zero-shot classifier on some images of creations on Tangram from the internet using classes with simple Tangram arrangements for the text input, l. We again find it to be quite good at recognizing these creations with high confidence. Fig. 3.3 shows a few examples.

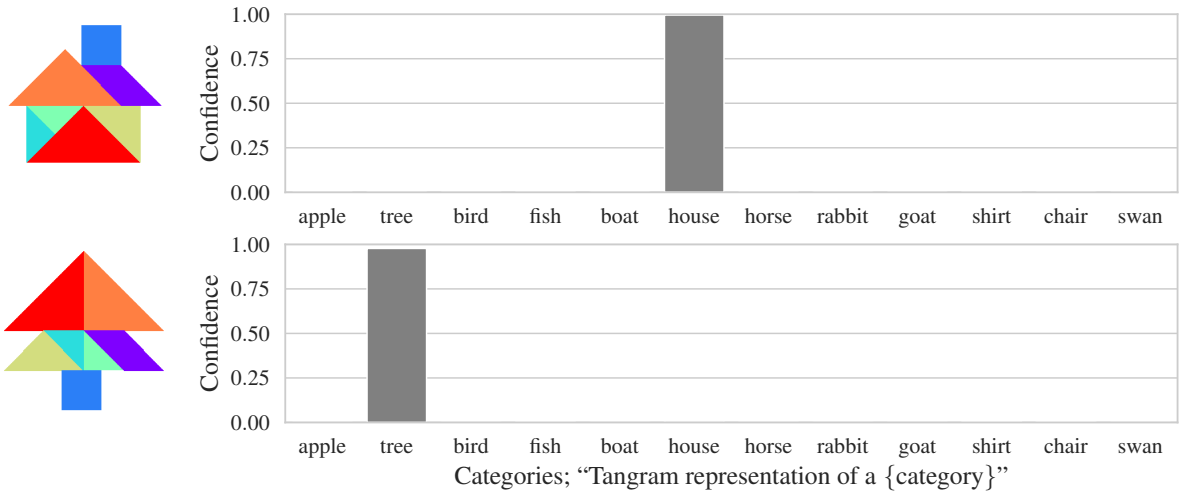


Figure 3.3: CLIP on a few simple Tangram creations.

We do not find a significant trend in the effect of colors on the performance of CLIP, but inversions seem to slightly affect the distribution of inferences in certain cases (one such example is given in Fig. H.4 in the appendix, where the distribution of CLIP is skewed in grayscale and white-on-black cases). Consequently, we use both colored and black-on-white binary renderings for our experiments.

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

3.2 Trajectory Analysis with Random Rollouts

Although we find that CLIP is good enough at recognizing creations in our environments, we wanted 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 conduct some experiments with *random rollouts* in the environments, i.e. starting from a meaningful creation, we perturb the creation with randomly sampled actions and observe the effects on the reward trajectories. One such rollout in ShapeGridWorld is shown in Fig. 3.4.

These experiments show that CLIP is susceptible to noise in its inferences. As a consequence of this, the reward is potentially somewhat sparse, and more critically, there is a large semantic bias in not-so-meaningful (random) images. The following sections discuss these problems in detail.

3.2.1 Noisy Rewards

We observe that the inference of CLIP breaks suddenly with small changes in the image, i.e. its confidence in its classification is sensitive and adjusting the position of a single pixel can trigger an abrupt response in the distribution of CLIP's inferences. This problem is visible in the time range $\sim 100 - 250$ in Fig. 3.4. This phenomenon can be interpreted as a jagged reward landscape with many local optima.

This renders the rewards unreliable as they can suddenly change with every little action leading to a lack of continuous incremental feedback towards any goal, potentially stagnating the agent in a state of indecision. If we imagine this scenario in reverse with an active controller, i.e. when the agent starts from the random image and tries to converge to a meaningful image by relying on the semantics entropy reward, it might register a sudden burst of rewards for small changes in the image, but these rewards could just be artifacts of the noise in CLIP's inferences. This would make it difficult for the agent to gauge the values of its actions to lead itself to eventual rewards, which in turn will lead to difficult and inefficient learning of action policies.

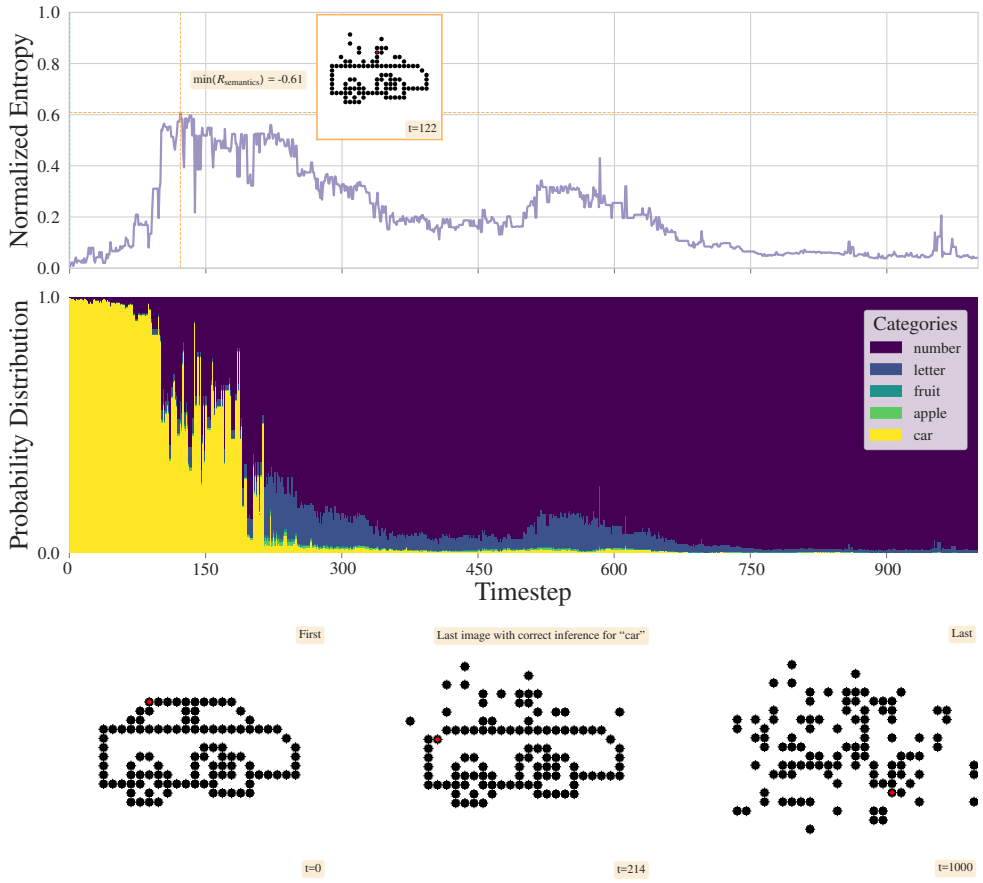


Figure 3.4: Random rollouts in ShapeGridWorld. Going from left to right in time, we perturb the grid pixel-by-pixel and observe CLIP's confidence change. The bottom row shows the distribution of CLIP's inferences over the labels "`car`", "`apple`", "`fruit`", "`letter`", and "`number`", formulated as "`sketch of a {}, in a grid`". The graph in the top row shows the distribution's entropy (negative of the semantics entropy reward). The first and last images in the bottom row are the initial and final images respectively. The middle image shows the image after which "`car`" is no longer the categorized class.

This problem is also affected by the choice of the environment, and we notice that it is less pronounced in the Tangram environment, albeit still very much present.

3.2.2 Semantic Bias in Random Images

Another consequence of the noise in CLIP we observe from the timesteps after ~ 250 in Fig. 3.4 is that it confidently classifies random images to a class instead of having a flat distribution, i.e. false positives, as evident here with the class "number".

This noisy reward signal can lead the agent to be stuck in plateaued local optima (with a rather meaningless creation that it erroneously finds confident).

The two problems, noisy rewards and semantic bias in random images, are not independent. There is a non-trivial anti-correlation between the two. That is, increasing reward density (or reducing reward noise) involves increasing confidence in imperfect/partial creations (or adding more semantic bias in random images).

To be able to effectively use the semantics entropy reward, these problems need to be alleviated. Our efforts to achieve this are discussed Section 3.4.

3.2.3 Class Preference in CLIP

Another intertwined issue we face related to the problem of semantic bias in random images is that of class imbalance. There are some classes in which CLIP is consistently more confident than others and leans towards them over others when unsure. This includes classes that signify broader concepts such as "animal", "fruit", "letter", "object", or "number", depending on which is present. Thus, in free-play, CLIP might converge to these classes more often than others.

We can circumvent this issue by avoiding certain classes and choosing enough creative possibilities for the semantics entropy reward such that CLIP's preferences balance out (see Section 3.4.2). Although, this is non-trivial. Especially for the Tangram environment, in which the creations can be quite abstract, there is only a limited set of semantically distinct, simple, and feasible categories.

3.3 Trajectory Analysis with Closeness Costs

Before we start improving and running simulations using inferences from CLIP, we need to also ensure that the agent could ideally reach a reward-conditioned goal in our environments, i.e. that they are *solvable*, especially for Tangram as it is introduced as a new environment in this study. Moreover, it is crucial to establish this solvability under sparse rewards. To ensure that our controller does this efficiently and robustly, a good choice of the iCEM controller hyperparameters and the environment configuration is essential. These hyperparameters and configurations are dependent on each other so they need to be optimized together.

For example, 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 need not be very high, because the controller can potentially reach the goal in a few steps and does not need to plan far ahead, but on the other hand, 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 of its actions.

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 predefined 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 make the hyperparameter optimization problem quite complex and the sheer number of these parameters renders this task very expensive. Given the high computation resources and time required to run simulations with the semantics entropy reward using CLIP, it is infeasible to do a proper search over all the hyperparameters. Thus, we instead use an alternative reward function to analyze the effect of the hyperparameters in the Tangram environment. We call this ersatz reward function the *closeness reward*.

Closeness reward is defined in the context of a fixed target creation in the environment, and it is formulated as the negative of a distance function between the current creation and the target creation in the state space of the environment.

We use two different distance functions, 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,

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

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.

Fig. 3.5 shows a sample closeness reward run for the dense closeness cost on the Tangram environment.

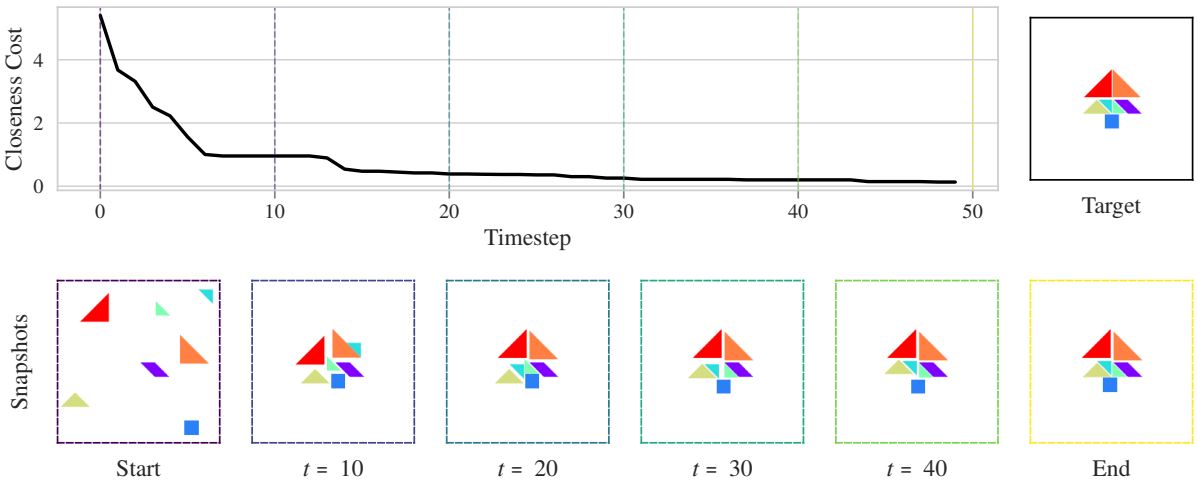


Figure 3.5: A successful closeness reward rollout in Tangram[1].

Using these closeness reward/cost formulations, we perform 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 judge the resulting trajectories and their underlying parameters, we compare their cumulative rewards over the rollouts.

We find that the controller is 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 do not notice much difference in the performance of the controller.

The comparison results are only summarised here for brevity. Please refer to the appendix chapters A and B for more details about these hyperparameters of the iCEM controller and the environments respectively. It also lists the optimal and default values we observed and typically used in our simulations (unless specified otherwise).

Generally, in a continuous Tangram grid, with a step size of 4, moving one object at a time for one action step, an iCEM controller with 128 trajectories and a minimum planning horizon of 8 with 3 inner iterations can reach the goal in $30 - 60$ timesteps.

The tuning is more demanding under sparse incremental rewards, which requires a higher planning horizon or more sampled trajectories to reach the goal. This directly corresponds to higher computational costs.

The cost aggregation functions `best` and `sum` are comparable in performance and perform better than `best-`l, `last`, and `last-`l. With the semantics entropy reward, we expect the `sum` aggregation function to perform better, due to its summation operation which averages over the rewards in the planning horizon and makes it more robust to noise, essentially having a regularizing effect on the reward landscape. However, given effective regularization using other methods, the greedier `best` aggregation function could be more efficient in reaching the goal and escaping local minima.

From this analysis, we gained intuitions about the minimal set of required hyperparameters that allow the controller to solve the environment. These minimal parameters are later used as a starting point for the simulations with the semantics entropy reward.

[1]Simulation (with animation) of another similar closeness rollout with a house as the target is available at https://t.ly/jV1dH or https://drive.google.com/drive/folders/1yGFYLp96O3bEkMu-byQIRGGnOpkvJRcT.

3.4 Improving CLIP Rewards

The problems discussed in Section 3.2 demand some regularization techniques that can mitigate the noise in CLIP. The regularized semantics reward introduced in Section 2.3.2 (2.24) has some hyperparameters that can be tuned to make the reward landscape smoother. Namely, the temperature of the softmax function (τ), the text baseline regularization strength (α_l), and the image baseline regularization strength (β_i). There are also additional settings such as the categories for the semantics entropy reward, the choice of prefixes and suffixes, the baselines, and the rendering function that can be tweaked to reduce the noise in the reward signal.

The complex nature of the reward landscape and the high dimensionality of the search space make it difficult to analyze the effects of these hyperparameters. In this section, we try to gain an intuition over them and reduce the search space. For this purpose, we use the best of the previous rollouts with the closeness reward from Section 3.3 and run post hoc inferences on the resulting sequence of image observations using CLIP to calculate the trajectories of the resulting *indirect* semantics entropy reward.[1]

3.4.1 Effect of Prefix and Suffix

Given a generic set of simple Tangram creations for the creative possibilities, we find that the choice of prefix and suffix can affect the reward trajectories in unexpected ways. Fig. 3.6 compares the performance for some combinations. Performance here is measured in terms of the mean-shifted cumulative rewards.

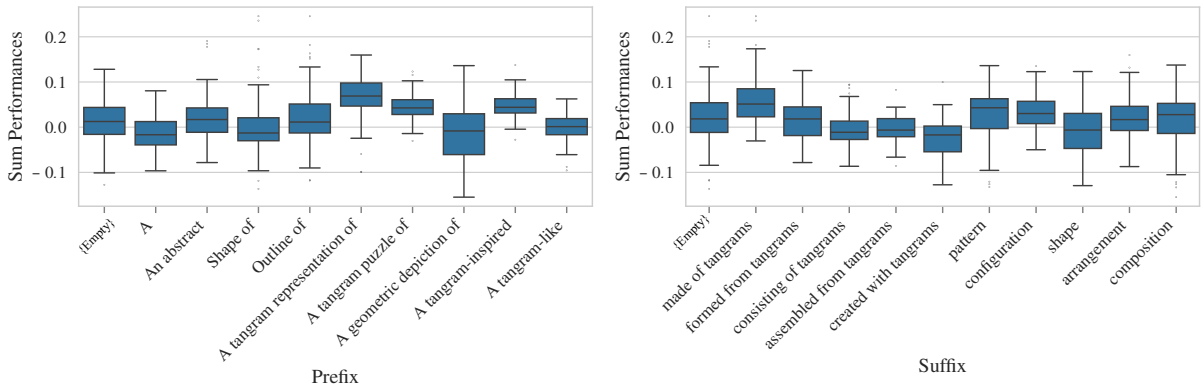


Figure 3.6: Effect of prefixes and suffixes on semantics entropy reward trajectories in Tangram.

Consequently, to capture the underlying trends and find robust values that are agnostic to such prompt engineering in the other parameters discussed next, we analyze and compare their effects over multiple choices and combinations of these prefix-suffix pairs.

This analysis shows some clear trends in the effects of these hyperparameters on the reward landscape which we present in independent ablation studies in the following sections. Unless specified otherwise, we chose the best-performing values of hyperparameters not under consideration in that section.

To demonstrate the effects, we look at the mean cost and cumulative cost trajectories of semantics entropy reward averaged over the many combinations of prefix-suffix pairs as a measure of performance. Ideally, lower cumulative costs indicate better performance, but it has to be interpreted with caution due to imperfections in CLIP and the subjective nature of the goal here (semantic expression). Even with a higher cumulative cost, the agent can reach a more semantically expressive state. We discuss these results, caveats, and effects accordingly in prose where necessary.

[1]The large number of combinations due to the high dimensional space makes it infeasible to show the subtleties of the interplay between the different 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 combined effects.

3.4.2 Effect of the Number of Creative Possibilities

To alleviate the problem of class preference in CLIP, we experiment with many combinations of different numbers of categories. Fig. 3.7 shows its effect on the entropy of CLIP inferences on a random image.

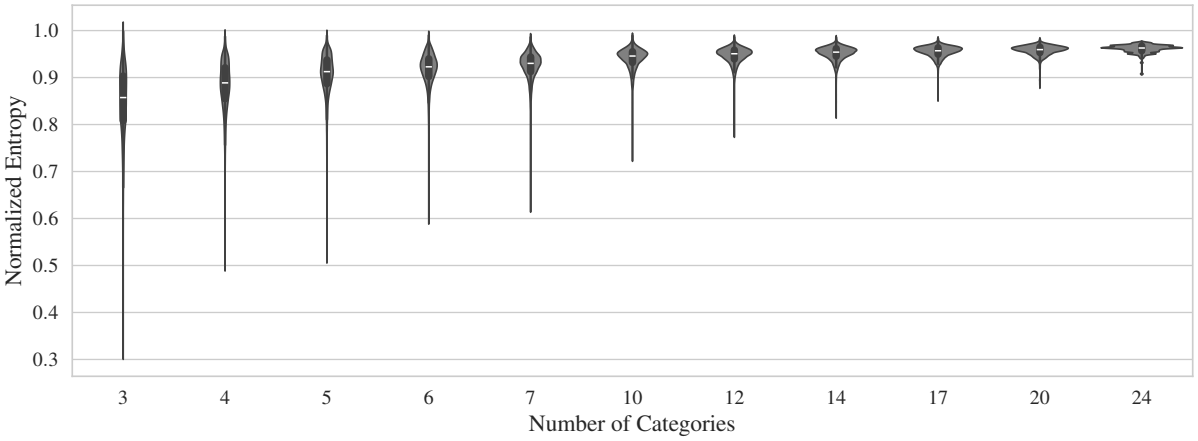


Figure 3.7: Effect of the number of categories on CLIP inference distribution over a random image.

We observe that too few categories can promote semantic bias and class preference in random images as they can have low entropies. Yet, too many categories can exacerbate the problem of noisy rewards since there are potentially more misleading distractions for the agent. This suggests that a moderate number of categories would be optimal. For our simulations, we typically choose a set of $10 \sim 20$ categories with simple Tangram patterns like house, bird, boat, or fish, and avoid broad categories like "number" or "letter" to circumvent the problem with class preference (see Section C.1.1 for the complete list). For SGW, we extensively experiment with different sets (see Section G.1 for results).

3.4.3 Effect of Temperature

Temperature is a crucial hyperparameter for the softmax function in the semantics entropy reward as it directly controls the entropy of the CLIP distribution (see (2.4)). Fig. 3.8 shows its effect, averaged over multiple choice of prefixes and suffixes. The original CLIP publication recommends a temperature of 0.01 for most use cases, which we also find to most often work best to reach a good optimum. Temperatures up to 0.02 are good as well but any value below this range performs significantly worse as it adds to the noise of CLIP, and any temperature above this range renders the reward landscape too smooth and the reward trajectories too flat, leading to a loss of any semantic bias from CLIP.

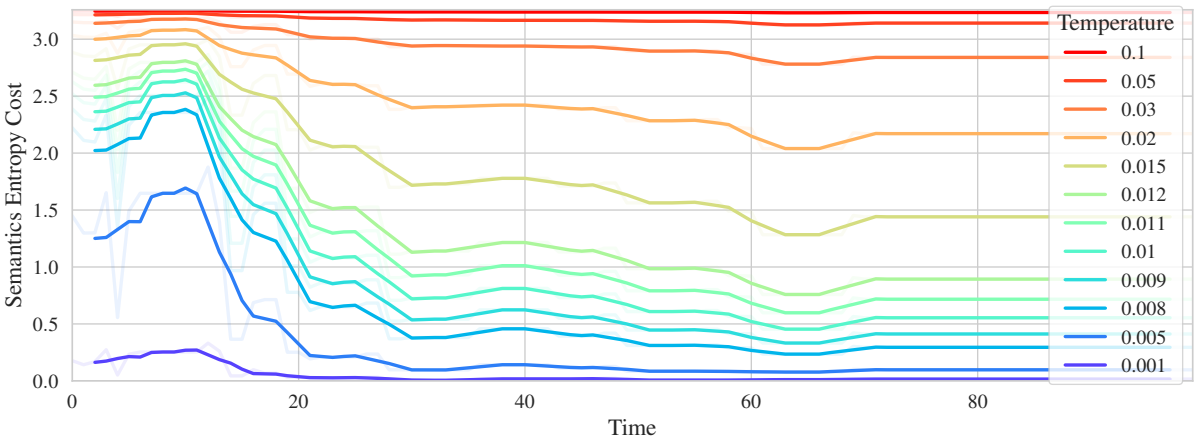


Figure 3.8: Effect of temperature on semantics entropy reward trajectories.

3.4.4 Effect of Baseline Regularization

The regularization strengths of the text baseline (α_l) and the image baseline (β_i) are arguably the most important hyperparameters for the regularized semantics entropy reward. Baseline regularization helps by providing better directional guidance in the CLIP embedding space.

We compare the cumulative semantics entropy rewards of the post hoc inferences on closeness reward trajectories with different values of α_l and β_i to find the optimal values and find the best values of the regularization strengths to be somewhere in the middle of the two extremes, with the optimal image baseline regularization strength slightly higher than the text baseline regularization strength. This is in line with the findings of Rocamonde et al. (2023) in goal-conditioned reward settings.

Fig. 3.9 shows these results, averaged over multiple choice of prefixes and suffixes.

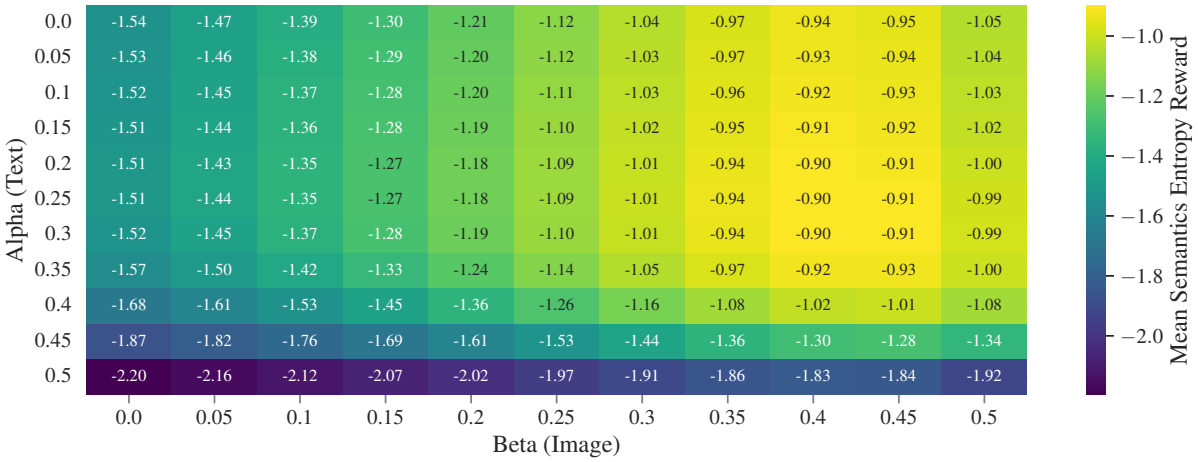


Figure 3.9: Effect of regularization strength on semantics entropy reward trajectories.

3.4.5 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 experiment with this as well, and instead of using the initial description of the environment as the text baseline, we use a negative formulation of this description.

Yet, we do not find this to be helpful in our experiments. Instead, it seems to flatten the reward landscape (see Fig. 3.10[1]). We think this is a consequence of CLIP's language encoder being a bag-of-words model, which makes the negative embeddings essentially close to the target embeddings, and effectively zeros them out.

3.4.6 Effect of Adding Post-Suffix

Roth et al. (2023) found that adding a concept and *post*-suffix to the label in the text input to CLIP improved the quality of the inferences. This post-suffix can even consist of a random string of characters, and by itself also improves the quality of the inferences, albeit slightly.

Following these insights, we also experiment with adding random jibberish to the end of the label as a post-suffix but do not find it to affect the reward. Fig. 3.11[1] shows the effect of different post-suffixes on the semantics entropy reward trajectories, averaged over combinations of prefixes and suffixes.

[1]Note that in the trajectory plots; Fig. 3.8, 3.10, and 3.11, the mean trajectories are filtered with a moving average filter of length 3 to reduce clutter and emphasize the trends. The original unfiltered mean trajectories are underlaid in the same plot in fainter colors. In the cumulative cost trajectory plots, the underlying trajectories with different prefix-suffix pairs are also shown.

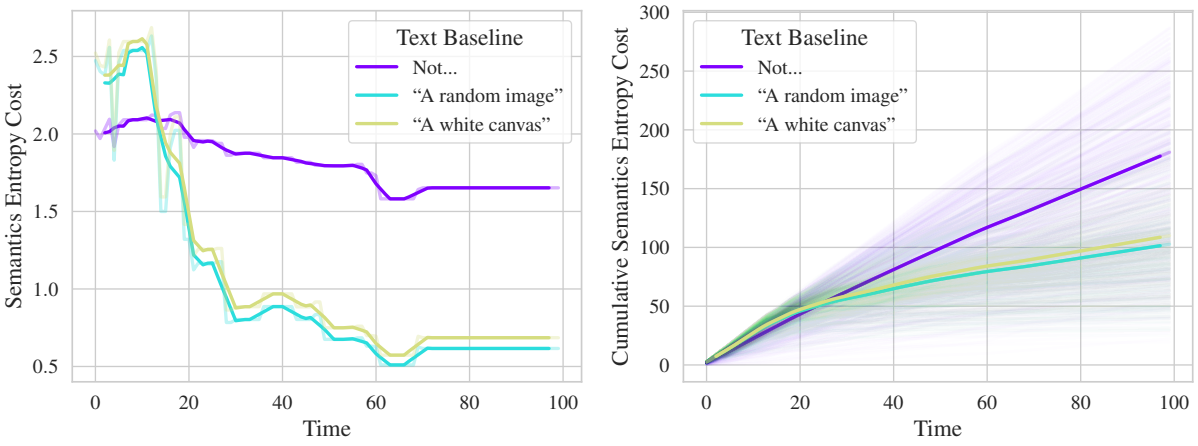


Figure 3.10: Effect of different text baselines on semantics entropy reward trajectories.

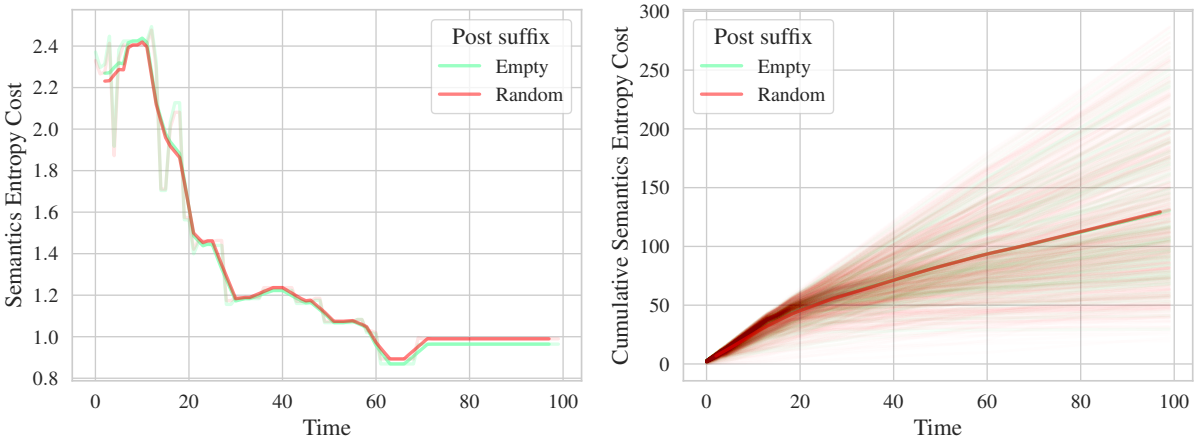


Figure 3.11: Effect of different post-suffixes on semantics entropy reward trajectories.

3.4.7 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,

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

where L_ι is the contrastive cosine-similarity loss used to train CLIP, κ is the regularization strength, 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 experiment with this regularization method to train toy convolutional neural networks (CNNs) for classifying handwritten digits from the MNIST dataset (LeCun and Cortes, 2010), which we call *flatnet*. Furthermore, we augment the training dataset with varying sizes of random-image samples labeled with a uniform distribution over the categories and train multiple networks.

We find the preliminary results to be quite effective in reducing the noise of the model; it seems to relieve both of the problems from Section 3.2 – the reward trajectory is smoother, and there is less semantic bias in random images and even less class preference for random inputs.

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

3.4.8 Adversarial Performance

To particularly tackle the problem of semantic bias in random images, we collect samples of the false positive image observations from our rollouts, called *adversarial observations* (Fig. 3.12), by filtering out the image observations with low entropy from all our runs and then manually removing the ones that are true positives.

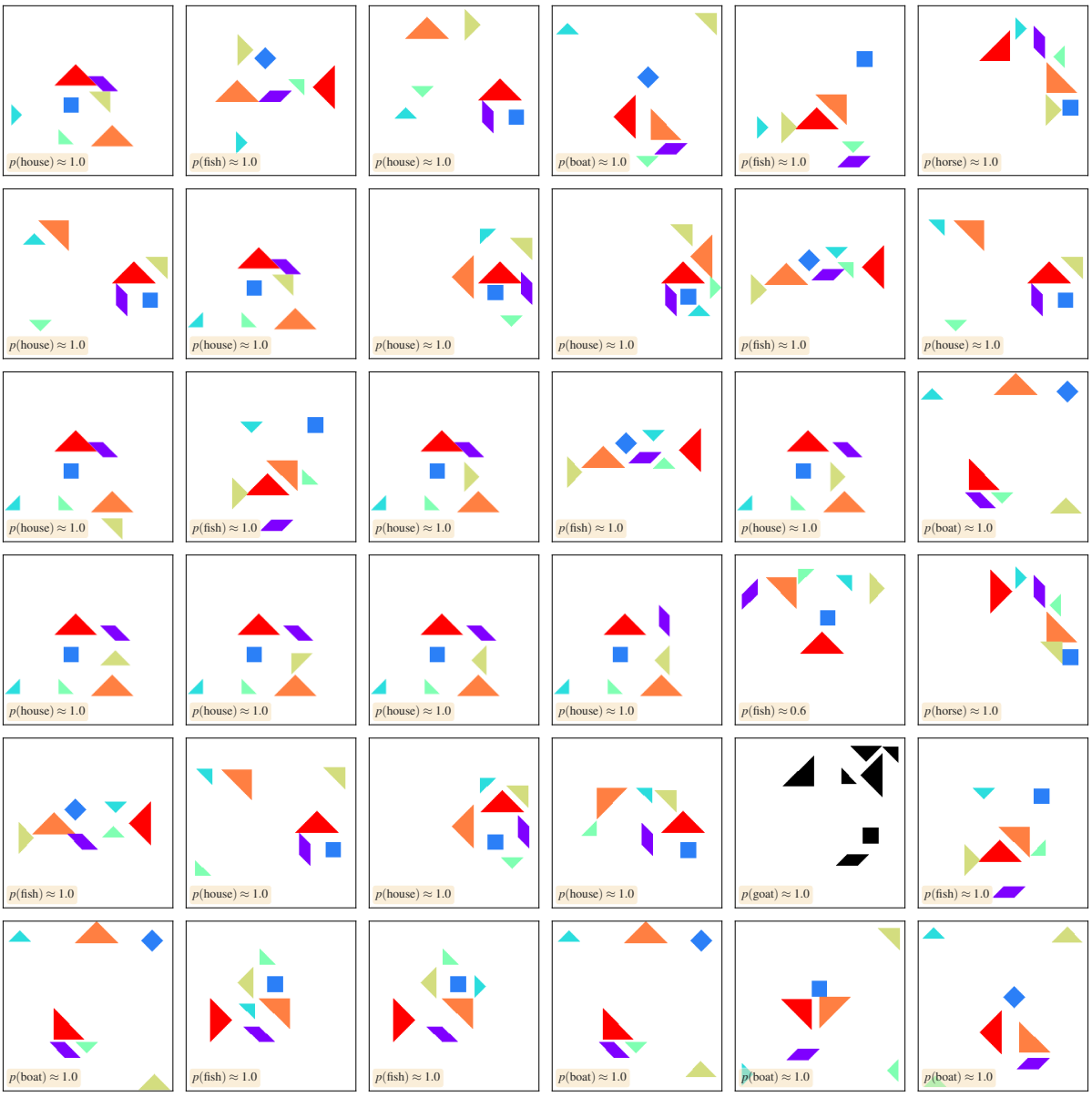


Figure 3.12: Adversarial samples from simulations.

Subsequently, we search 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. increases its specificity while maintaining its sensitivity. In this analysis, we use the mean entropies of the false positive and true positive images as our metrics to gauge the performance improvements.

Fig. 3.13 to 3.16 show the effect of temperature, regularization, negative embeddings, and post-suffixes on discerning true positives from false positives, over a range of prefix-suffix combinations. The results from these can be compared to the corresponding results shown before in Section 3.4.3 to 3.4.6.

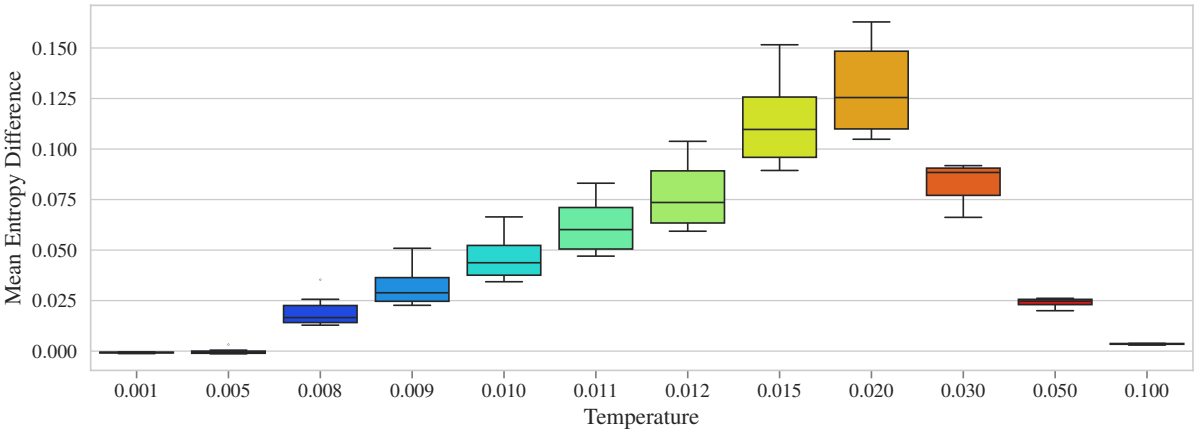


Figure 3.13: Effect of temperature on discerning true positives from false positives.

Fig. 3.13 shows the effect of temperature on the adversarial performance with the difference in the mean entropies of the false positives and true positives. Higher temperatures up to 0.02 are better at smoothing the reward landscape just enough to improve its adversarial performance.

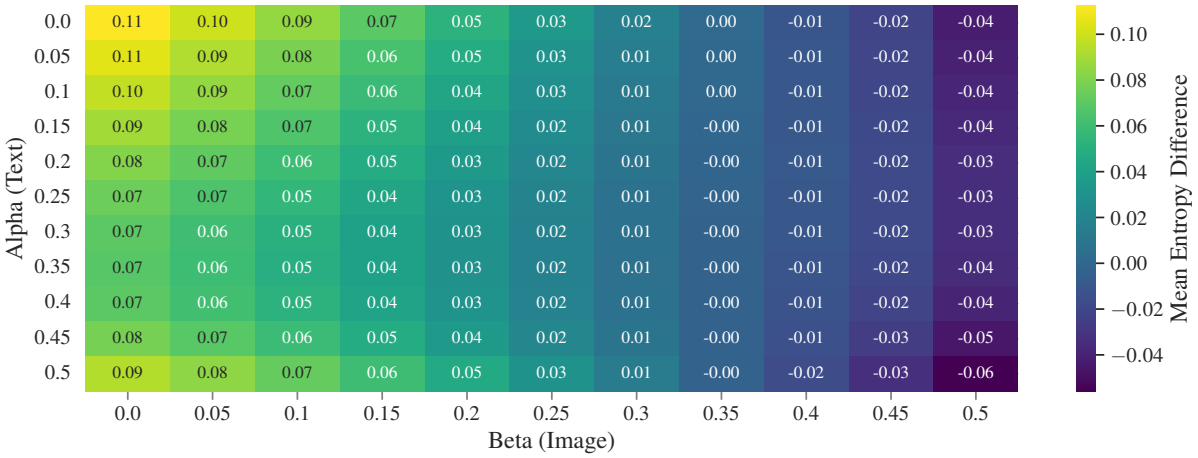


Figure 3.14: Effect of regularization strength on discerning true positives from false positives.

Interestingly, the trends in the regularization strength of the text baseline (α_l) and the image baseline (β_i) shown in Fig. 3.14 are essentially opposite from the ones in Fig. 3.9 that showed the effect on the reward landscape. These results are not at odds with each other, but rather complementary. As noted before, there is an anti-correlation between reward noise and semantic bias in random images.

An optimal choice of baseline regularization strength in 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 states away from the goal in such a way that it is better at intimating the agent if it is close to its ultimate goal, i.e. it increases the similarity or correspondence of this state to the goal state. This is in contrast to the adversarial performance where we desire to make this difference more pronounced. These results are further discussed in detail in Chapter 4.

Next, we look at the effect of different text baselines and post-suffixes on the adversarial performance. To visualize these results, we plot the mean entropies of the false positives and true positives for different combinations of prefixes and suffixes for the text input in a 2D space. Ideally, we would like the points (which represent prefix-suffix combinations) to live above the diagonal, i.e. the mean entropy of the false

positives should be higher than the true positives. Additionally, we plot boxplots of the difference in the mean entropies of the false positives and true positives to summarize the effects.

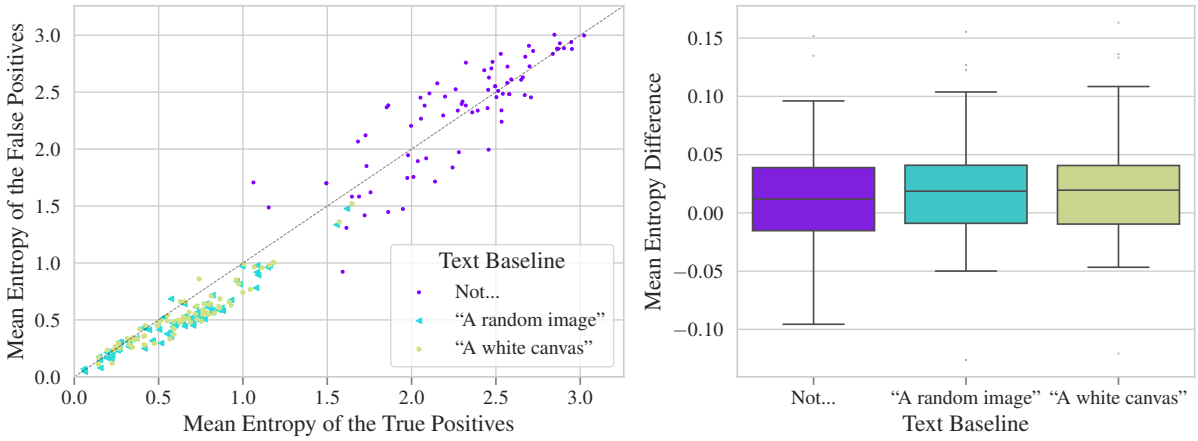


Figure 3.15: Effect of different text baselines on discerning true positives from false positives.

Fig. 3.15 shows the effect of different text baselines on the adversarial performance. 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.

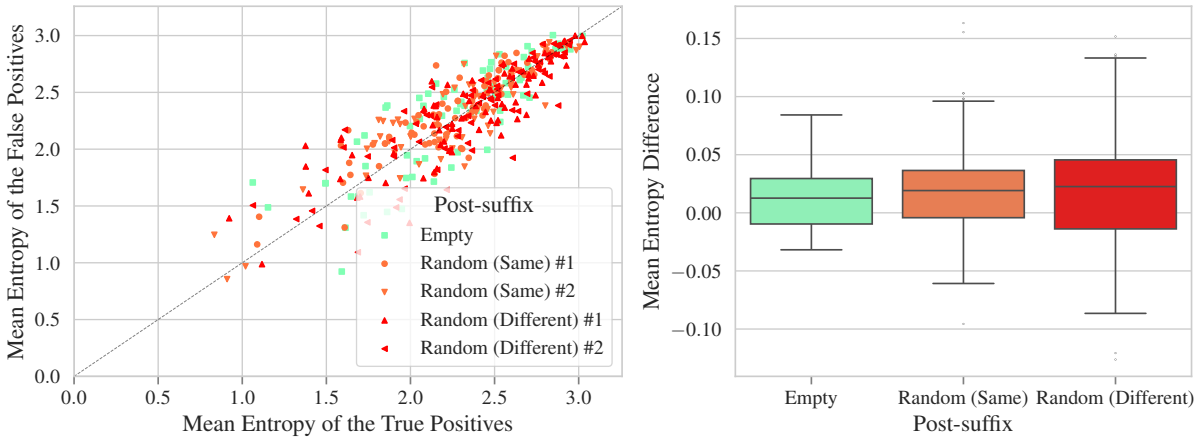


Figure 3.16: Effect of different post-suffixes on discerning true positives from false positives.

Fig. 3.16 shows the effect of different post-suffixes on the adversarial performance. Since the mechanism for choosing a post-suffix that works well in general is not exact, for this analysis, to ensure more exhaustive and rigorous tests, we also use different formulations of the post-suffix and consider cases such as having the same random post-suffix for the categories across every prefix-suffix combination and having a random one for each of the categories and combinations. As before, we see no net effect in adding or omitting the post-suffix, but just a difference in variance.

3.4.9 Effect of Image Texturing

We also experiment with adding textures in the environment renderings in hopes of bringing these image inputs closer to in-distribution for CLIP to improve its inference quality. This is done by adding a constant shading to the Tangram polygons (see Fig. 3.17). Although, we do not find a significant improvement in the reward landscape with this texturing. Fig. 3.18 shows its effects.

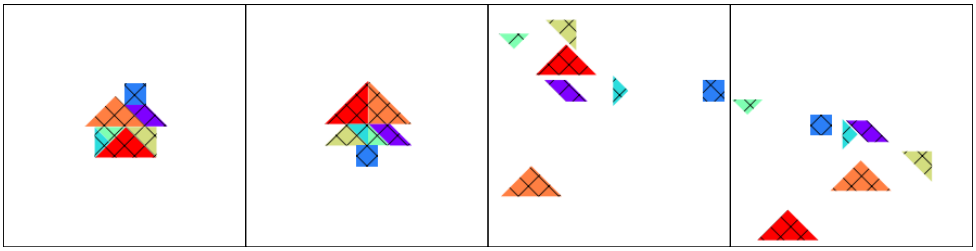


Figure 3.17: Texturing in Tangram.

3.4.10 Effect of Image Operations

We also test the idea of using subtle image operations like shearing before inference to denoise the CLIP signal. We expect 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.

As shown in Fig. 3.18 trajectory plot, shearing the images before inference does improve the quality of the inferences slightly by smoothing the reward landscape. It has a significant performance improvement in the adversarial case as well (Fig. 3.19 boxplot), but it also affects the confidence in the true positive images, thus leading to lower performance by our cumulative reward/cost metric as seen in Fig. 3.18 cumulative reward plot.

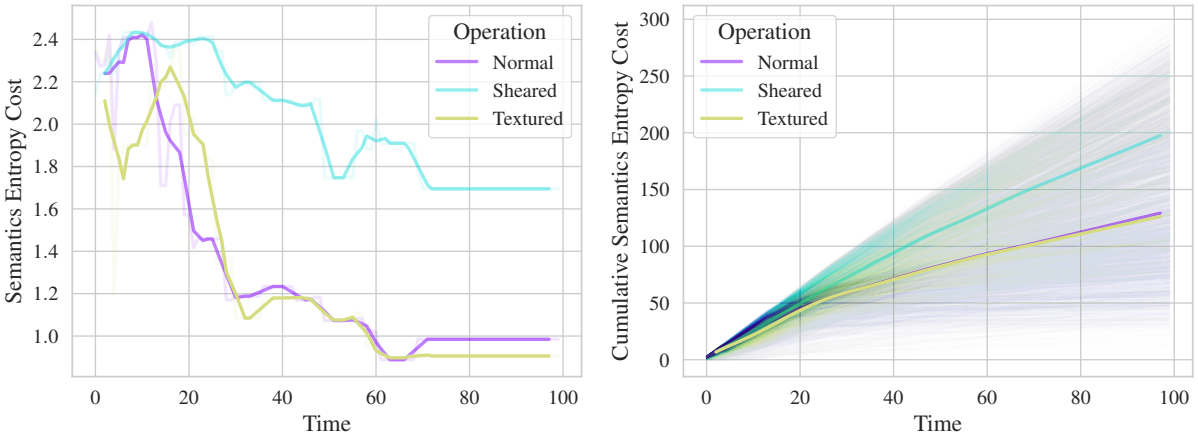


Figure 3.18: Effect of texturing and image operations on semantics entropy reward trajectories.

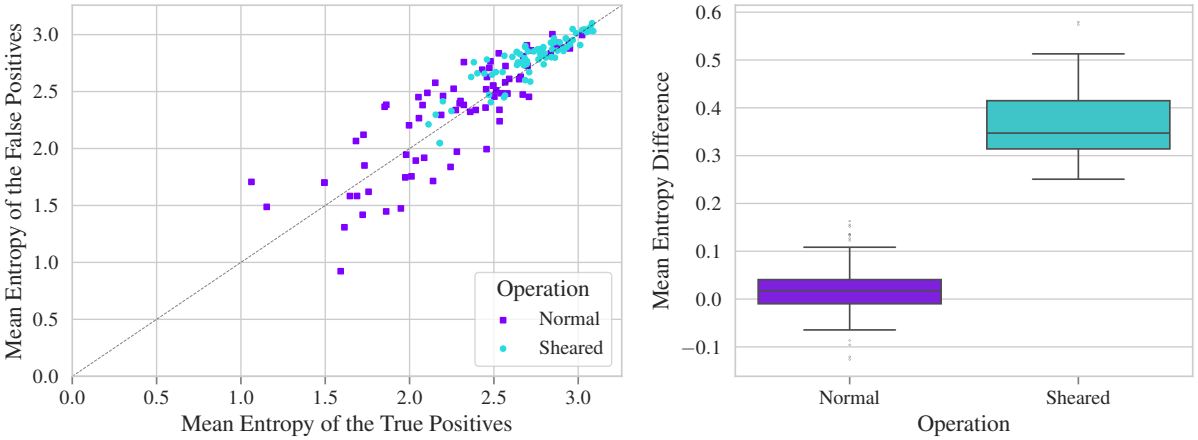


Figure 3.19: Effect of image operations on discerning true positives from false positives.

To additionally show the interdependence of the hyperparameters discussed above, Fig. 3.20 and Fig. 3.21 compare the adversarial performance of negative embeddings and different post-suffix patterns over sheared images respectively.

We observe that the shearing operation when used with baselines based on initial states helps improve the adversarial performance of CLIP as seen by the points gathering in the upper right corner of Fig. 3.20.

Yet, in the following sections, we do not use this method, because of the associated added computational costs of rendering, and because we realize that we can get a similar effect by tuning the baseline regularization strengths and the temperature.

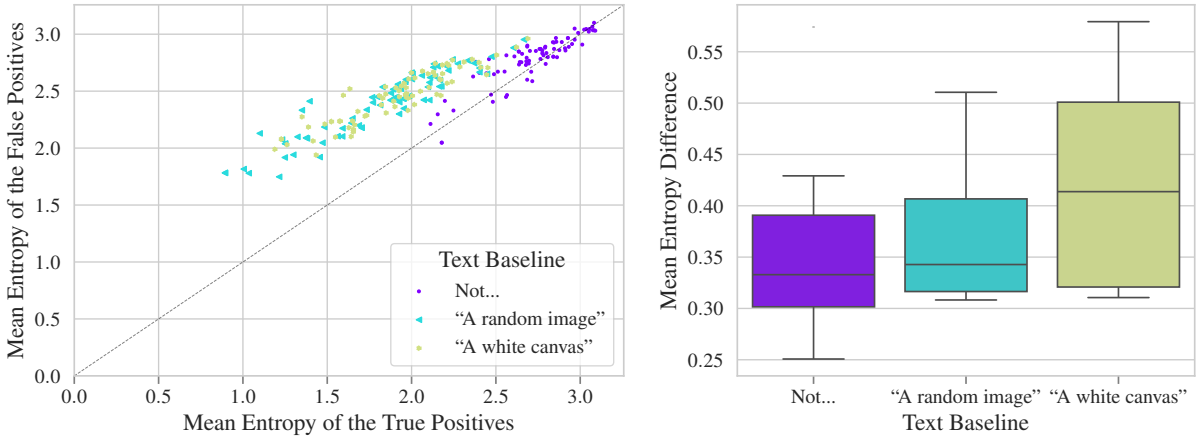


Figure 3.20: Effect of different text baselines on discerning true positives from false positives with sheared images.

The addition of post-suffixes in the sheared image case on the other hand impairs the adversarial performance, as shown in Fig. 3.21. Shearing seems to surface the post-suffixes' originally intended regularization/smoothing effect which is not as visibly pronounced in the non-sheared case in the study of its effect on the reward trajectories (Fig. 3.11).

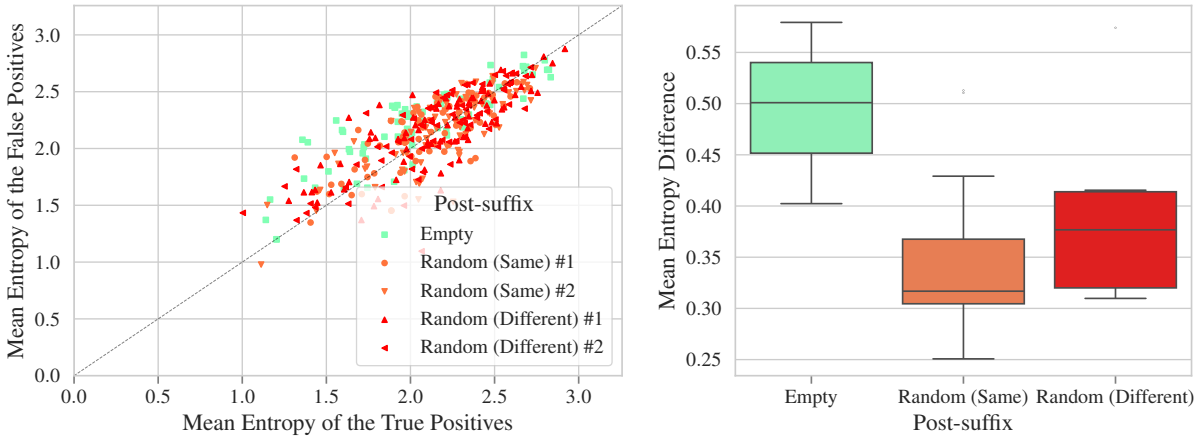


Figure 3.21: Effect of different post-suffixes on discerning true positives from false positives with the sheared image. Note that the temperature used here is lower, $\tau = 0.012$, as the points gather in the upper right corner at higher temperatures.

3.5 Simulations using the Semantics Reward

Using the suitable combinations of the controller and environment hyperparameters from Section 3.3 and insights into the regularized semantics entropy reward hyperparameters from Section 3.4, we finally run simulations (rollouts) with the semantics reward to generate creations in the environments. We are generally able to consistently generate meaningful creations in both environments, but the quality of the creations varies slightly with the exact choice of these reward hyperparameters.

These simulations were computationally expensive and required significant time to run. Thus, we were limited in our ability to further investigate, confirm, and fine-tune all the hyperparameters exhaustively enough. Consequently, we focus our analyses on the main few – the baseline regularization strengths and the effect of the additional regularity reward, RaIR. The results of these analyses are showcased in the following sections. All figures plot the mean cost trajectories over the rollouts for 10 different random seeds for each hyperparameter combination.

While we also run enough simulations to confirm our intuitions about the other hyperparameters, only a few of our analyses use enough restarts with different random seeds to have the statistical power to make justified general claims. These additional results are available in Chapter G of the appendix.

To evaluate the obtained final creations, in addition to the cumulative reward, we also ask three human evaluators to manually rank all the final creations on a scale of 1 to 5 based on their apparent underlying semantic expressiveness and subjective feel/quality of the patterns. In the following parts, shown creations (in Tangram) not particularly interesting according to these rankings are desaturated to avoid visual clutter. Interesting creations based on these results are curated in the gallery in Chapter I of the appendix[1].

3.5.1 Effect of Regularization Strengths

We run simulations with different regularization strengths to reconcile their effects with the results from the previous sections. Fig. 3.22 shows the heatmap of the mean cumulative rewards for different combinations of the regularization strengths.

We find that non-zero regularization strengths (α_l and/or β_i) do improve the performance, with mid-range values $(\alpha_l, \beta_i) = (0.3, 0.3)$ having the best performance by these metrics. This corroborates the results in the post hoc trajectory analysis study in Section 3.4.4.

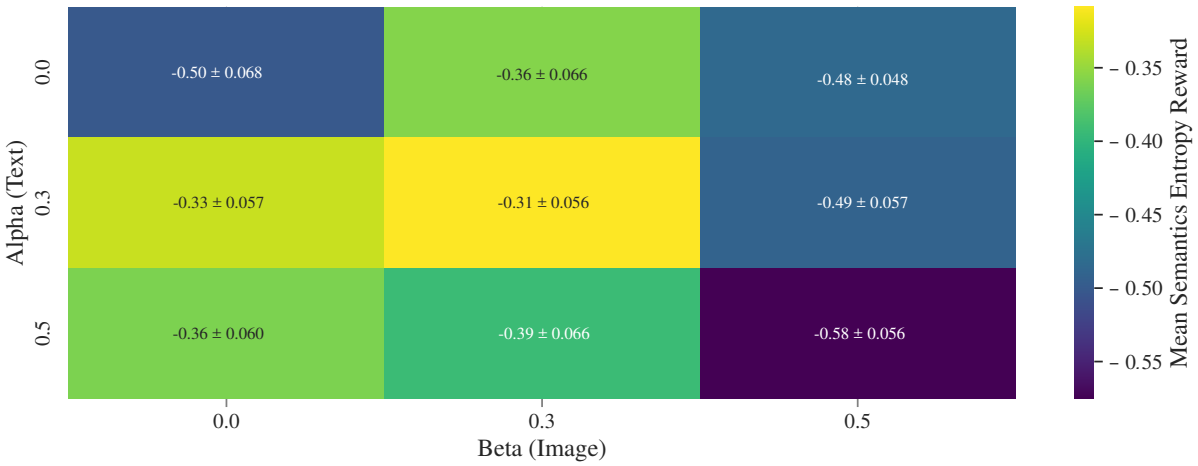


Figure 3.22: Performance of regularization strengths on semantics reward.

[1]Animations and summary graphs of all the simulations and creations referenced in this thesis are available on https://drive.google.com/drive/folders/1RdG86GLLujH3z6eDRIvdaczokpr8krnc or https://t.ly/49O3h.

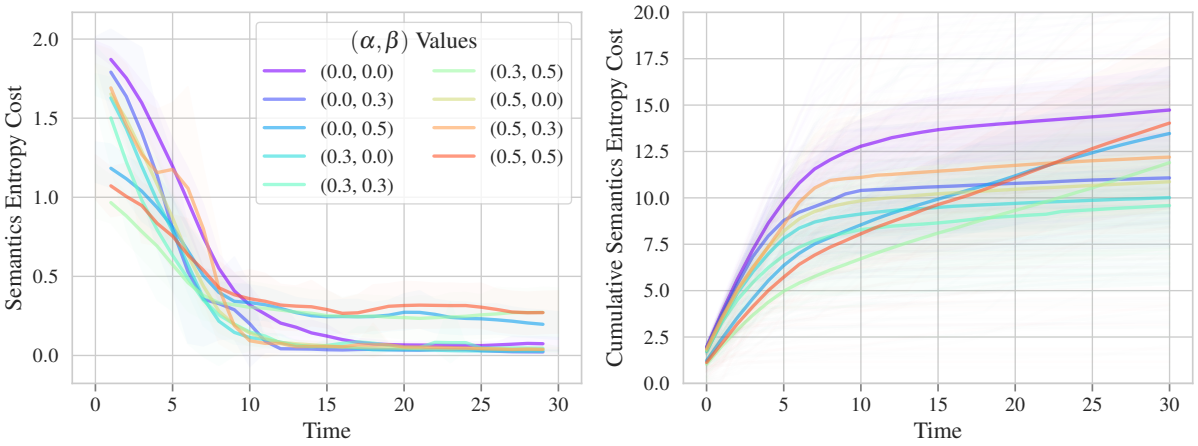


Figure 3.23: Effect of regularization strengths on semantics entropy reward trajectories.

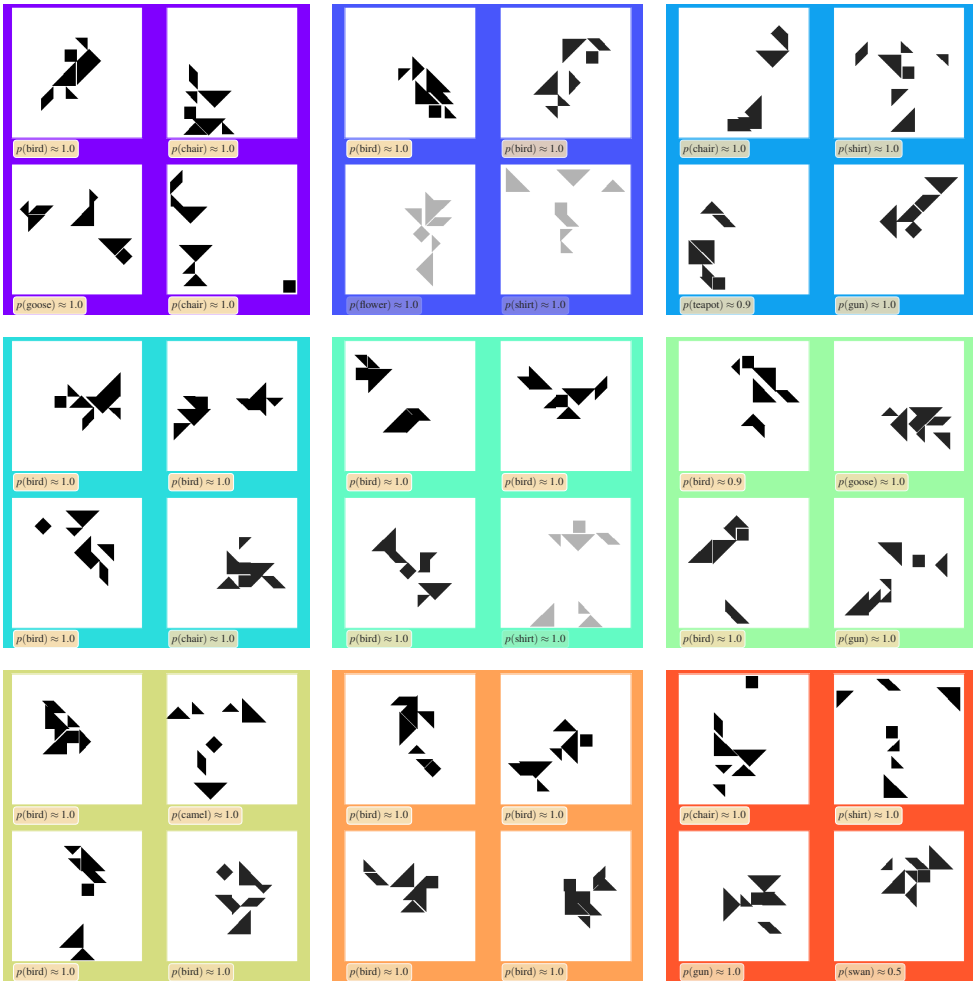


Figure 3.24: Samples of creations with different regularization strengths.

Fig. 3.23 shows the effect of regularization strengths on the semantics entropy cost trajectories and Fig. 3.24 shows some samples of the creations with different regularization strengths. Noticeably, the three cost trajectories with high image baseline regularization strength ($\beta_i = 0.5$) are too flat and do not reach an optimal zero entropy. At $(\alpha_l, \beta_i) = (0.5, 0.3)$ and $(0, 0.3)$, the cost trajectories seem slightly more sparse and noisy. For the remaining combinations, $(0.5, 0)$ and $(0.3, 0)$, the trajectories are well-shaped and there is no significant difference in the performance or quality of creations.

3.5.2 Effect of RaIR

Throughout our tests, we noticed a high degree of regularity in semantic creations and high correlations between RaIR and the semantics entropy reward, suggesting a strong connection between the two. Indeed, we find that creations simulated with RaIR are much better in some cases than without it.

Even though the performance with RaIR in Tangram is slightly worse in terms of the cumulative reward shown in Fig. 3.26, the creations shown in Fig. 3.27 are semantically more expressive and recognizable. Adding structural bias using RaIR helps with grounding and coalescing the arrangements in better forms and patterns, resulting in creations that are more organized and appealing. It is particularly helpful in the later stages of planning as the agent first quickly converges to local minima in the semantics reward landscape and then optimizes for regularity while maintaining its semantic expressiveness. Thus RaIR helps in escaping and navigating different local minima, and in turn, leads to more meaningful creations. This mechanism is quite apparent in the simulation summary shown in Fig. 3.25.

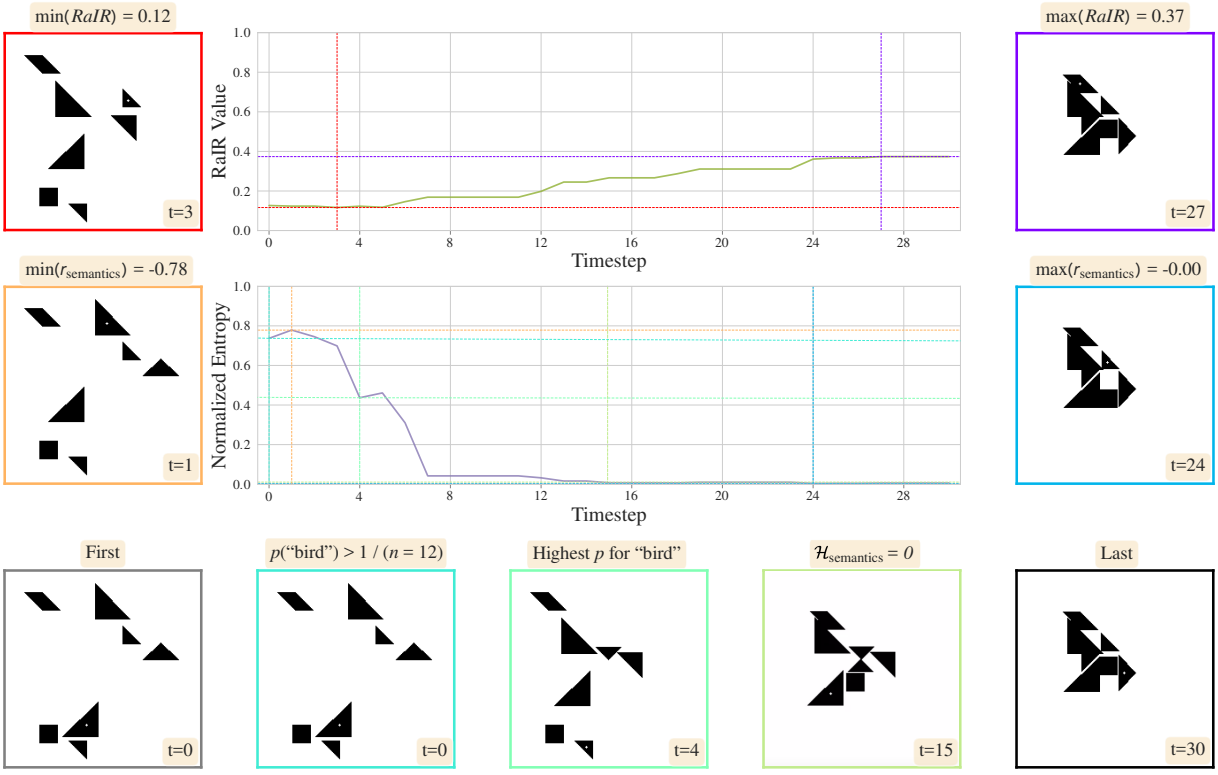


Figure 3.25: Simulation on Tangram with the complete semantics reward leading to a bird creation. The first row shows the RaIR trajectory, the second row shows the semantics entropy cost, and the third row shows snapshots of labeled milestones across time. Notice how the semantic reward is optimized first, but does not lead to a very meaningful bird creation. As the regularity is optimized further, while maintaining the semantic state in the CLIP reward landscape, the creation becomes more recognizable.

We also discover that the exact proportion of the weight of the RaIR to the semantics entropy reward λ from (2.29) is not as important as the presence of RaIR itself. Although, very high values of λ can lead to a loss of semantic expressiveness. Consequently, we use a value of $\lambda = 1$ in our simulations, which results in consistent well-expressed creations.

Fig. 3.28 and Fig. 3.29 show the effect of RaIR on the semantics reward in ShapeGridWorld, where the effect is only marginal, both in terms of the cumulative reward/cost performance, and the quality of the creations. Although the creations are neater with RaIR, with fewer random pixels, there is no effective difference in the semantic expressiveness of the creations. Results for many more simulations on ShapeGridWorld are given in Chapter G of the appendix.

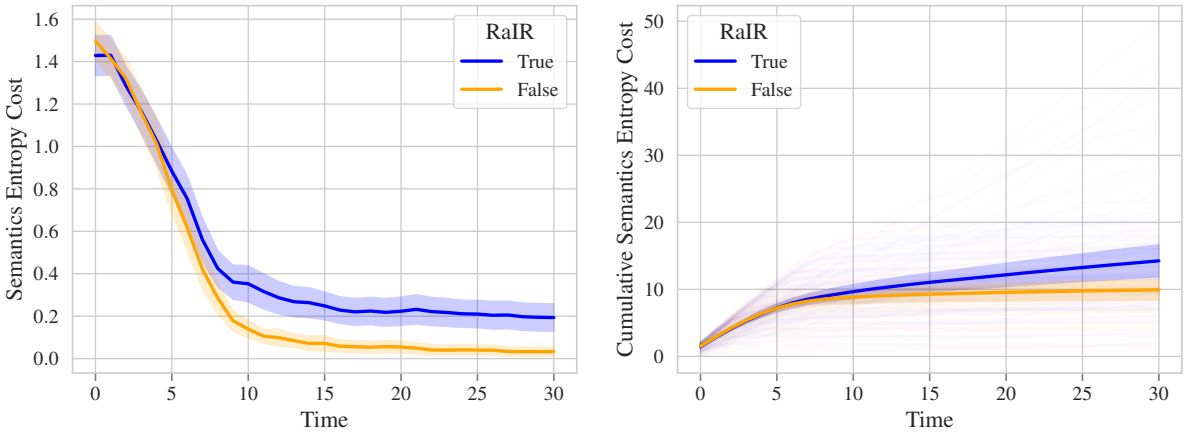


Figure 3.26: Effect of RaIR on semantics entropy reward in Tangram.

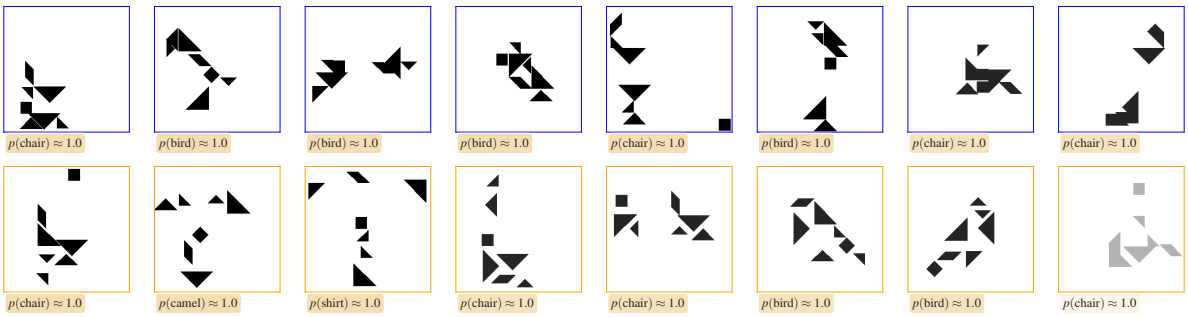


Figure 3.27: Samples of creations with and without RaIR in Tangram.

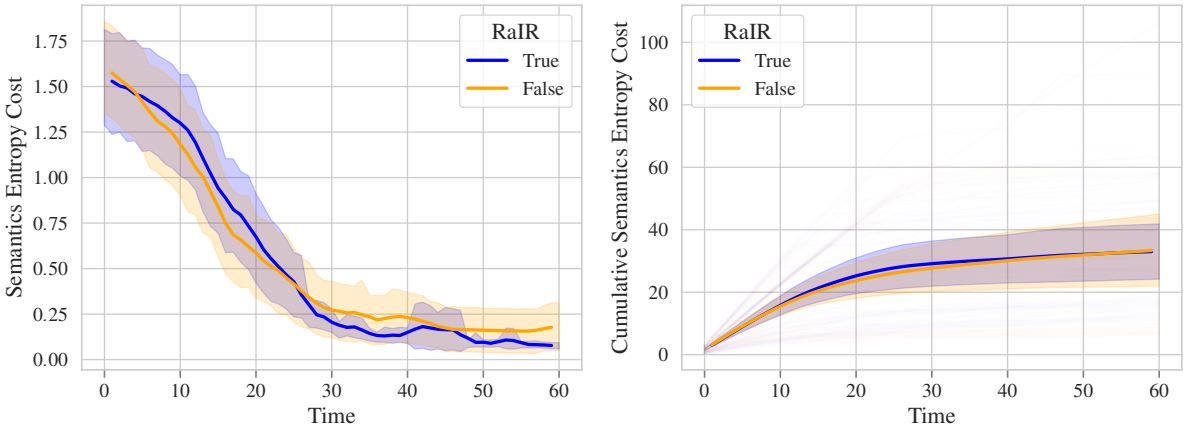


Figure 3.28: Effect of RaIR on semantics entropy reward in ShapeGridWorld.

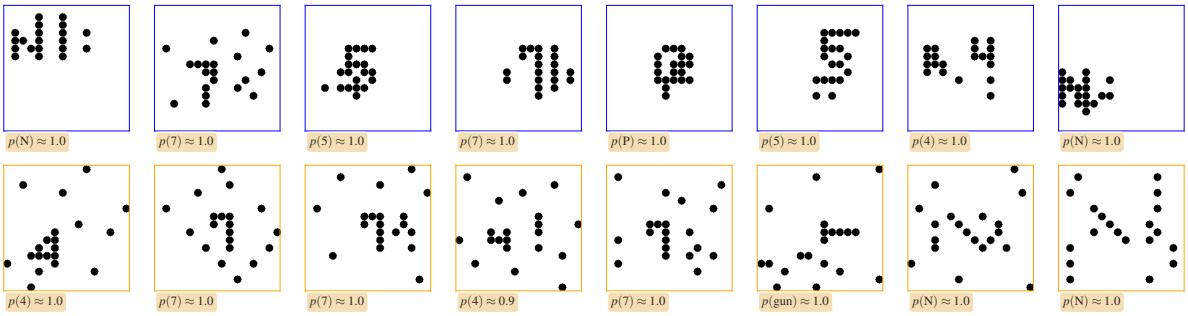


Figure 3.29: Samples of creations with and without RaIR in ShapeGridWorld.

Chapter 4

Discussion

Semantic expression is an innate nature of human exploration during creative free-play. Motivated by these observations in cognitive science and developmental psychology, in this thesis, we try to leverage multimodal foundation models to realize this behavior in artificial agents, i.e. bias curious exploration in free play towards visual semantics. Towards this end, we formulate the semantics entropy reward using semantic inductive bias from the large vision language model CLIP as a means to give the agent a freedom akin to free play in humans and enable it to reach semantically meaningful and visually expressive states.

To test the efficacy of this reward in achieving rich and diverse creative expression, we extended the ShapeGridWorld environment and created the Tangram environment. We demonstrate the performance of CLIP in these abstract settings across parameters and establish that CLIP is noisy and suffers from problems such as semantic biases in random images.

Thus we develop methods to improve its use for our purposes, where we build upon recent work on baseline regularization in using VLMs as a source of goal-conditioned rewards to formulate the regularized semantics entropy reward. Furthermore, we adapt and devise other additional methods such as negative embeddings, post-suffixes, and tweaking the rendering function (adding texturing and modifying the images with a little shearing) to denoise and improve the reward signal.

Subsequently, in our preliminary analyses, we show how these techniques, agnostic to the variations due to prompt engineering, affect the reward trajectories. The different additional methods have subtly disparate desired effects on the reward signal, but we find the baseline regularization strengths and temperature of the softmax activation function to be the most effective in reducing CLIP's noise.

Baseline regularization improves the shape of the reward landscape by introducing tradeoffs between the directions of effective semantic change in the CLIP embedding space. The delta embeddings it employs at higher strengths capture the important aspects of the desired change in the text embeddings so the reward function can focus on imbuing this change in the image observation space, and the absolute embeddings at lower strengths help in providing context. By considering a tradeoff between the two as a hyperparameter in either modality, we try to balance these two important aspects. We find that moderate values of the two baseline regularization hyperparameters ($\sim (0.3, 0.3)$) are the most effective in guiding the agent to semantically meaningful states. Overall, we think that baseline regularization makes the reward landscape denser and softens/improves its gradients.

However, this results in poorer adversarial performance as a consequence, i.e. a larger semantic bias in random and imperfect images, which renders the final creations less expressive. This is tackled by the temperature parameter, which complements baseline regularization, and at values between $0.015 - 0.02$, smoothens the reward landscape and reduces this semantic bias. The resulting reward trajectories have less noise and the creations are less abstract. (A comparable effect could be achieved by shearing the image renderings.)

The regularization methods improved and extended here, particularly the addition of image baseline regularization, can also be used in other applications in reinforcement learning where VLMs are used as reward models, such as for goal-conditioned rewards. Additionally, the insights gained from our analysis of the different methods for improving the noise of CLIP can be used to improve the applications of CLIP and other VLMs in general.

We use these insights from our results to guide the hyperparameter choice and search in the main experiments using the regularized semantics reward and show that tweaking just the regularization strengths and temperature is generally enough to achieve good reward signals. Moreover, these hyperparameters are robust and slight variations in their values in the given ranges do not affect the reward signal significantly. The modified reward is effective in consistently guiding the agent to a range of semantically meaningful states in both environments.

Although, these creations can still sometimes show a lack of expressiveness. This degenerative effect can be attributed to poorly shaped (flat) reward signals near goal states of absolute meaning and could be exacerbated by artifacts of the environment design or limitations of the agent's capabilities since the controller uses finite-horizon planning which hinders it from finding global optima.

Regardless of the reason, this is further mitigated by the introduction of the complementary regularity reward RaIR which improves the quality of these arrangements by grounding and coalescing the objects in better forms and patterns, resulting in improved creations that are more reminiscent of real objects. This is especially evident in Tangram where the creations become more recognizable and appealing as they become more compact. In ShapeGridWorld, the creations are neater and more organized, with fewer random pixels. RaIR particularly archives this in the later stages of the rollout by incentivizing the agent to modify its creation while maintaining its semantic value. Effectively, it helps in escaping and navigating different local minima, encouraging exploration, and in turn, leads to more meaningful creations through degrees of visual abstraction.

Limitations and Future Work

Although our methods are effective in shaping the reward signal to guide the agent to semantically expressive states, the final creations the agent achieves still suffer from a lack of good diversity. Some categories such as "house" in Tangram are chosen more often than others. In ShapeGridWorld, the agent prefers the letter "N" followed by the letters "R" and "S" with numbers and letters, or "gun" with a set of simple objects. Even removing these categories just gives rise to other such categories especially preferred by CLIP. (Counts of all the creations we observed in our simulations are summarized in Fig. H.1 and Fig. H.2 in the appendix.)

This class preference can be in part due to the specific shapes of these creations and how they might be more suited or be easier to stumble upon in the environments we use. For example, in Tangram, a house can simply be created with a face-up triangle above a square. The agent also prefers to make a hammer with a parallelogram next to a square, and a heart with simply a red triangle pointing down. However, we could not identify such patterns in ShapeGridWorld.

We think that these limitations mainly stem from the inherent biases in the CLIP model. Perhaps using newer and larger open-source vision-language 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, with better generalization, reduced noise, and diminished biases.

In our supplementary work, we also explore the possibility of de-noising CLIP by fine-tuning with entropy regularization and re-training with random images with our toy models vis-à-vis *flatnet*. This shows some promising results with smoother entropy transitions, a stable (less noisy) inference confidence over time, and a flatter distribution (high entropy) for false positive (random) samples. Scaling it to the full CLIP model could improve results further.

Another promising method of reducing CLIP noise could be adding redundant (nonsensical) labels in the creative possibilities and computing the semantics entropy reward only over the main set of labels. This could potentially trim away the noise and attenuate the problem of large semantic bias in random images. Redundant labels have been shown to facilitate learning in humans (Lupyan et al., 2007).

Other improvements could be using temporally adaptive weights for the regularity reward to explicitly force the mechanism that we observed with RaIR in improving expression, i.e. starting with a smaller λ to encourage semantic expression and then increasing it later. These phases could also be shorter and interleaved to further ensure that the agent does not degenerate to local minima.

Furthermore, multiple terms for regularity can be introduced, for example, a local regularity in every object's limited vicinity with higher precision and a global coarser regularity with higher granularity (see (C.1)). This is motivated by observations that structures in the real world are regular at different scales in a similar manner, and might lead to more complex/richer creations.

Future work can also experiment with giving the agent control over the choice of object to move. This is a harder control problem and is not tested in our work for the same reason. Instead, we reduce the complexity of the problem and focus on the reward signal. Similarly, simulations could be run in complex or realistic environments, with more objects, elaborate shapes, or additional dimensions.

Currently, our experiments are limited to fully observable MDPs with known dynamics using ground-truth models, and the application of RaIR additionally requires object-centric state-space representations where the observations are disentangled per object. In our case, this is manually specified in the form of the precise positions of all the objects. Such representations that can capture relational inductive biases of the objects-object interactions in the environment could instead be learned. Accordingly, testing the semantics reward in partially observable environments with learned models (with or without object-centric representations) could be a good direction for future work.

Our entropy-based approach to modeling emergent behavior presents a novel perspective on imbuing free-form creativity in artificial intelligence. With the advances in deep learning and neural architectures, as the open-source vision-language models further improve and overcome the limitations of noise and biases, this approach can be extended to span a large number of creative possibilities, potentially in more complex and realistic environments, and can be used to model complex human behaviors and interactions.

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 efficient human-like exploration i.e. curious information gathering behaviors with semantically-sound creative expression.

Eventually, this can be linked to the mechanisms of learning through play observed in humans, where an agent improves its understanding of the environment and builds better models and policies with a structured manner of exploration by guiding and linking its new experiences to its previous knowledge and experiences. This work is a small step along the path to recreate this incremental learning effect in artificial intelligence.

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 insightful advice.

I would also like to thank Georg Martius for giving me this opportunity and the members of the Autonomous Learning Group for their suggestions and discussions.

I am also grateful for the resources of the Max Planck Institute for Intelligent Systems, and the Graduate Training Center for Neuroscience at the University of Tübingen for their support.

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

Anderson-McNamee, J. K. and Bailey, S. J. (2010). The importance of play in early childhood development. *Montana State University Extention*, 4(10):1–4. 1

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y., and Goldblum, M. (2023). A cookbook of self-supervised learning. 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, 6, 22

Bruce, T. (2011). *Learning through play*. Oxford University Press, USA. 1

Bunzeck, N., Doeller, C. F., Dolan, R. J., and Duzel, E. (2011). Contextual interaction between novelty and reward processing within the mesolimbic system. *Human brain mapping*, 33(6):1309–1324. 1

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018a). Large-scale study of curiosity-driven learning. 2

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018b). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*. 2

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. 34

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, 6

Dietze, B., Dietze, D., and Kashin, D. (2011). *Playing and learning in early childhood education*. Wadsworth Publishing Company. 1

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). ii, 1, 2, 13

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*. 2, 5

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. 3, 9

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. 5

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29. 2

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95. 52

Itseez (2015). Open source computer vision library. https://github.com/itseez/opencv. 52

Khandelwal, A., Weihs, L., Mottaghi, R., and Kembhavi, A. (2022). Simple but effective: Clip embeddings for embodied ai. 2

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. 50

Kizilirmak, J. M., Thuerich, H., Folta-Schoofs, K., Schott, B. H., and Richardson-Klavehn, A. (2016). Neural correlates of learning from induced insight: A case for reward-based episodic encoding. *Frontiers in psychology*, 7. 1

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

Lai, N. K., Ang, T. F., Por, L. Y., and Liew, C. S. (2018). The impact of play on child development-a literature review. *European Early Childhood Education Research Journal*, 26(5):625–643. 1

LeCun, Y. and Cortes, C. (2010). Mnist handwritten digit database. 23, 50

Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., Andreas, J., Mordatch, I., Torralba, A., and Zhu, Y. (2022). Pre-trained language models for interactive decision-making. 2

Lupyan, G. (2012). *What do words do? Toward a Theory of Language-Augmented Thought*. 2

Lupyan, G., Rakison, D. H., and McClelland, J. L. (2007). Language is not just for talking: Redundant labels facilitate learning of novel categories. *Psychological science*, 18(12):1077–1083. 35

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. 3

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. 53

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. 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. 4, 45

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)*. 2, 4, 5

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. 2

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, 7, 14, 15, 22

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. 22

Sancaktar, C., Piater, J., and Martius, G. (2023). Regularity as intrinsic reward for free play. 3, 9, 13, 41, 47

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. 34

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. (2020). Planning to explore via self-supervised world models. 2

Shridhar, M., Manuelli, L., and Fox, D. (2021). Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*. 2

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press. 2

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. 3

Trott, S., Jones, C., Chang, T., Michaelov, J., and Bergen, B. (2023). Do large language models know what humans know? 2

Umesh, P. (2012). Image processing in python. *CSI Communications*, 23. 43

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. 52

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. 14

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*. 2

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. (2021). Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34. 2

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. 3, 9

Zosh, J. M., Hopkins, E. J., Jensen, H., Liu, C., Neale, D., Hirsh-Pasek, K., Solis, S. L., and Whitebread, D. (2017). Learning through play: a review of the evidence (white paper). Technical report. 1

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 are developed in Python.

A.1 ShapeGridWorld

Table A.1 lists the parameters of the original ShapeGridWorld environment developed by Sancaktar et al. (2023), along with the best values we found for our experiments.

Table A.1: Original ShapeGridWorld parameters.

Property	Description	Best/(Default)
`width`	Width of the discrete grid.	28
`height`	Height of the discrete grid. Kept same as `width`.	28
`n_pixels`	Number of "On" pixels, n_e.	~ 20
`shape`	Shape of a pixel block.	("circle")
`size`	Size of a pixel block.	7
`persistency`	Number of time steps an object (pixel block) is moved.	1

The state space of this environment is composed of the x and y coordinates of the n_e block pixels, with the addition of two dimensions for the state space of the agent – one that specifies which object is currently in focus and another that specifies how many times it has already moved, i.e. $\mathcal{S} \in \mathbb{N}^{2n_e+2}$. 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; $\mathcal{A} \in [-1, 1]^{2n_e}$. 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.

Table A.2: Additional ShapeGridWorld parameters.

Property	Description	Best/(Default)
`step_x`	Maximum number of steps an object can be translated in the x-direction in one action.	7
`step_y`	Maximum number of steps an object can be translated in the y-direction in one action. Kept same as `step_x`.	7
`persistency*`	Added the ability to move all objects at once.	1
`color`	A flag to enable grayscale values for pixels.	False
`invert`	A flag to control the inversion of the rendered images.	True
`control_boundaries`	If specified, objects inside these limits *initially* are marked.	(None)
`control`	A flag to define the scope of control, i.e. whether all the objects or only those marked initially by the control boundaries can be moved.	("all")
`max_reset_dist`	Maximum distance an object can be moved from its original position on reset. Only the objects marked initially are reset. There are no constraints if set to -1.	(5)

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. $\mathcal{S} \in \mathbb{N}^{2n_e}$. The corresponding action space in this case would be $\mathcal{A} \in [-1, 1]^{2n_e}$. 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.

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.

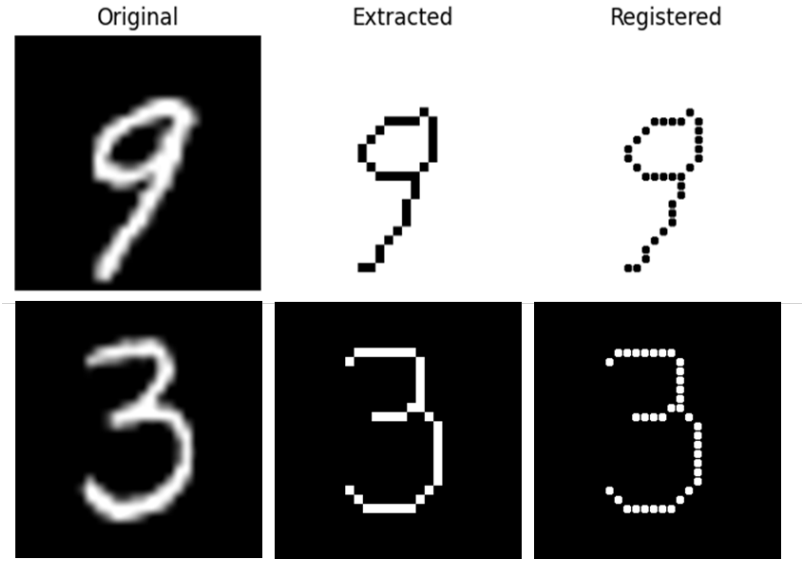


Figure A.1: ShapeGridWorld image registration on images from the MNIST dataset.

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, along with the default values.

Table A.3: Image registration parameters.

Property	Description	Default
`mode`	PIL Image mode in which the image is read.	"1" (Binary)
`invert`	A flag to control the inversion of the read image. The expected image should be white on black.	False
`threshold_ratio`	Threshold on the convolution sum, for binary grids.	0

A.2 Tangram

The Tangram environment is constructed 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.

Table A.4: Tangram parameters.

Property	Description	Best/(Default)
`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 $[0, 1]$.	1
`x_step, y_step`	Maximum number of steps a polygon can be translated in the x and y directions in one action.	5
`r_size`	Span of the discrete grid; number of rotation steps in $180°$. If set to 1, rotation is continuous in $[0, 180°]$.	1
`r_step`	Maximum number of steps a polygon can be rotated in one action.	4
`rotate`	A flag to enable/disable rotation.	True
`flip`	A flag to enable/disable flipping.	False
`persistency`	Number of time steps a polygon (pixel block) is moved. If set to 1, all polygons are moved at once.	1
`control_boundaries`	If specified, polygons inside these limits *initially* are marked.	(None)
`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" for the centroid, "center" for the mean of the vertices, or "complete" for the entire polygon.	("center")
`control`	A flag to define the scope of control, i.e. whether all the polygons or only those marked initially by the control boundaries can be moved.	("all")
`staging_boundaries`	If specified, only polygons inside these limits are rendered to get a state's corresponding image observation. See Fig. A.2	None
`staging_criteria`	Like `control_criteria` but for staging boundaries.	("complete")
`max_reset_dist`	Maximum distance a polygon can be moved from its original position on reset. Only the polygons marked initially are reset. There are no constraints if set to -1.	(-1)

The state space of this environment is composed of the x and y coordinates of the vertices of the $n_e = 7$ polygons, with the addition of two dimensions for the state space of the agent – one that specifies which polygon is currently in focus and another that specifies how many times it has already moved, i.e. $\mathcal{S} \in \mathbb{N}^{2+\sum_{i \in n_e} 2\ N_v(p_i)}$, 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. $\mathcal{S} \in \mathbb{N}^{\sum_{i \in n_e} 2\ N_v(p_i)}$. 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. The `step` parameter can then be understood as the minimum number of actions required to move a polygon across the entire grid. 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.

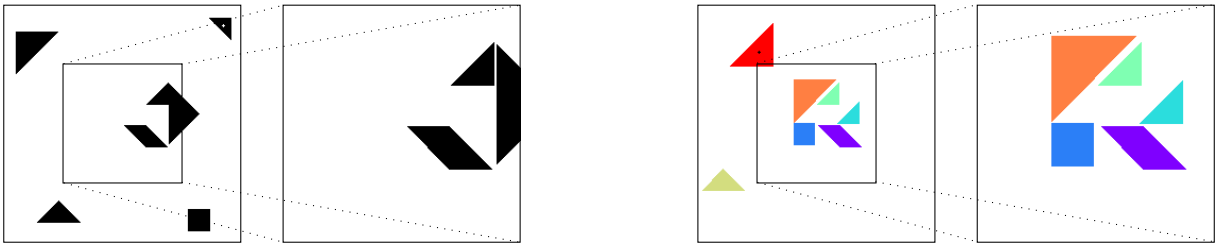


Figure A.2: Staging boundaries in the Tangram environment. Simulations using staging boundaries can be viewed at https://drive.google.com/drive/folders/1PLopyNdzpWiz6CK4EvR8oEAScACpUsW0 or https://t.ly/zb2cZ.

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, along with the best values we found.

Table B.1: iCEM controller parameters.

Property	Description	Best
`horizon`	Number of steps in the future to plan, η	16.
`n_trajectories`	Number of trajectories to sample, ν.	128
`cost_along_trajectory`	Cost/reward aggregation function to evaluate the trajectory, Υ. See Section 2.4.1 for explanations.	`best` or `sum`
`n_inner_iterations`	Number of iterations for the inner optimization loop, ι.	3
`init_std`	Standard deviation at which the sampling distribution is initialized, $\boldsymbol{\sigma}_0$.	0.6
`elites_size`	Size of the elite set, κ. See (2.36).	20
`discount_factor`	Discount factor for returns expected in the future, γ. See (2.41).	1.0

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

Table B.2: iCEM controller fixed parameters.

Property	Description	Default
`factor_decrease_num`	Factor by which population size is decreased, δ.	1.0
`alpha`	Momentum term for updating parameters of the sampling distribution, α.	0.1
`keep_previous_elites`	A flag to enable passing elites to the next timestamp (outer iteration).	True
`shift_elites_over_time`	A flag to enable passing elites over inner iterations.	True
`fraction_elites_reused`	Fraction of passed elites from the previous iteration (inner/outer) added to the sampled trajectory set, ζ.	0.3
`use_mean_actions`	If the mean of the sampling distribution is appended to the elite set at the end of inner iterations.	True
`noise_beta`	Colored noise exponent of the distribution, β	1.0.

Please refer to the original paper (Pinneri et al., 2021) for more details on the parameters of iCEM.

Appendix C

Reward Hyperparameters

This chapter details 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 (2.24), along with the best values we found for our experiments.

Table C.1: Semantics entropy reward parameters.

Property	Description	Best
`categories`	List of creative possibilities, $\boldsymbol{c}$.	$\lvert c\rvert \approx 20$
`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, $\mathbf{l}_b$. "A white canvas"	
`image_baseline`	Baseline image input, $\mathbf{i}_b$.	
`alpha_text`	Regularization parameter for text inputs, α_l.	0.5[1]
`beta_image`	Regularization parameter for input image observation, β_i.	0[1]
`model_version`	Name of the CLIP variant used.	(`ViT-L/14`)
`model_temperature`	Temperature for the softmax function in the semantics entropy reward, τ.	$[1, 2]$
`normalize`	If the reward should be normalized by maximum entropy, $\log(\lvert\boldsymbol{c}\rvert)$.	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 (2.29).	1.0

The list of creative possibilities we typically used in Tangram is – "`house`", "`bird`", "`tree`", "`horse`", "`rabbit`", "`goat`", "`shirt`", "`chair`", "`swan`", "`goose`", "`camel`", "`teapot`", "`hammer`", "`boot`", "`key`", "`gun`", "`apple`", "`car`", "`guitar`", "`flower`", and "`heart`". Since some categories were identified to be preferred by CLIP, such as "`house`" and "`bird`", they were removed in some experiments. In ShapeGridWorld, we typically used either a set of all single-digit numbers or numbers with letters together. We also experimented with the above list of categories in ShapeGridWorld. The effects of this choice are also compared in Section G.1.

[1] Please see Section 3.5.1 for more details.

C.1.2 Regularity Reward

In the introduction to the regularity reward in Section 2.3.3, we mentioned the additional hyperparameters ω – *resolution* and *bidirectionality*. This section provides more details about these hyperparameters.

The hyperparameter for resolution consists of two parts – precision and granularity. The following equation extends (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, along with the (default or) best values we found for our experiments.

Table C.2: Regularity reward parameters.

Property	Description	Best
`precision`	Precision of the compression algorithm.	SGW - 1 Tangram - 28
`granularity`	Granularity of the compression algorithm.	1
`bidirectional`	If the compression should be bidirectional.	False
`normalize`	If the reward should be normalized by the maximum possible entropy, $\log\left(\binom{N}{2}\right)$.	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 (2.29).	1.0

Please refer to the original paper (Sancaktar et al., 2023) for more details on the theory and implementation of *RaIR*.

C.2 Closeness Reward

Table C.3 lists the typical parameters of the closeness reward introduced in Section 3.3.

Table C.3: Closeness reward parameters.

Property	Description	Default
`reward_type`	Type of closeness reward – *sparse-incremental* or *dense*.	"sparse"
`reward_scale`	Factor with which the reward is scaled.	1.0
`reward_threshold`	Threshold, ε, for sparse rewards (3.1).	0.051

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.

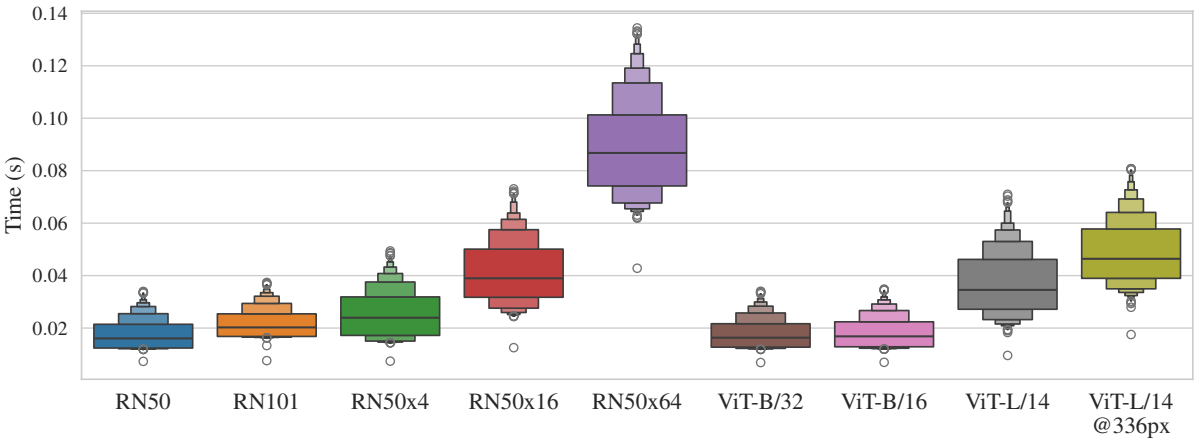


Figure D.1: Comparing times of different original CLIP models for embedding a single image.

Fig. D.2 shows the trajectories of different CLIP models on colored Tangram environments (without RaIR). The cumulative rewards and trajectories of the models are similar and do not give a clear picture of the models' performances, but the larger models seem to have better creations.

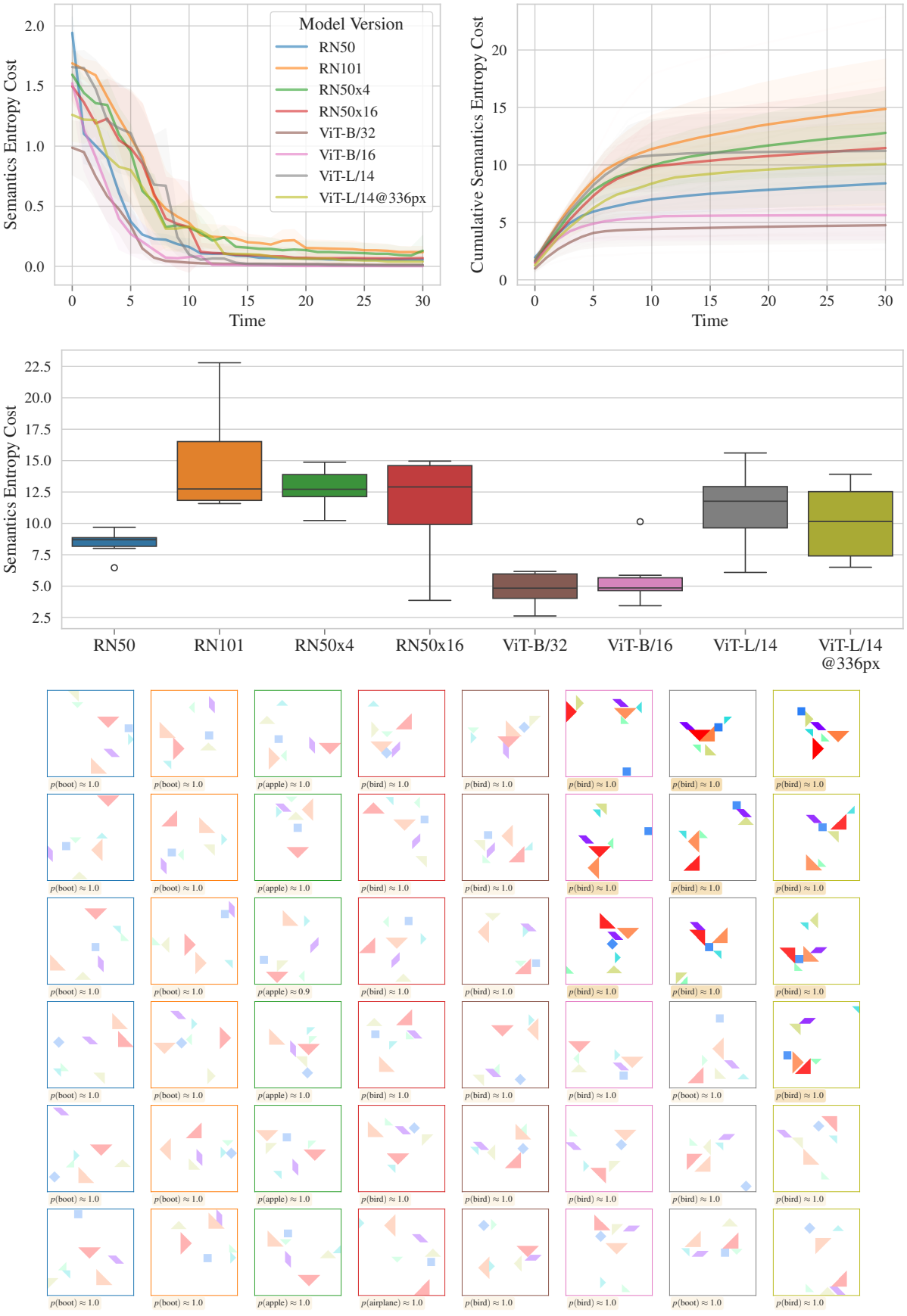


Figure D.2: Comparing reward trajectories of different CLIP models in Tangram. (10 seeds were used for each model.)

Appendix E

Flatnet

To test the feasibility and efficacy of fine-tuning CLIP with an entropy regularization technique (3.2) to reduce its inference noise, we conducted experiments on a smaller *toy* setup with convolutional neural networks trained as a single-digit number classifier on ShapeGridWorld registered images from the MNIST dataset (LeCun and Cortes, 2010). We additionally augment the MNIST training dataset with samples of images with random arrangements of pixels, target-labeled with a uniform distribution of confidence over the ten single digits.

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. The size of the MNIST training dataset is 60000. Table E.1 summarizes the models we compare. More details of these models can be found at https://pulkitgoyal56.github.io/flatnet.

Table E.1: Summary of the Flatnet models.

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

Fig. E.2 and E.2 show the reward trajectories of a regular and an entropy-regularized Flatnet model respectively. These trajectories are random rollouts (similar to Fig. 3.4) starting from a grid-registered image of the number zero. Trajectories of all the models listed in Table E.1 are neatly visualized and compared at https://www.sharecanvas.io/p/flatnet-comparison.

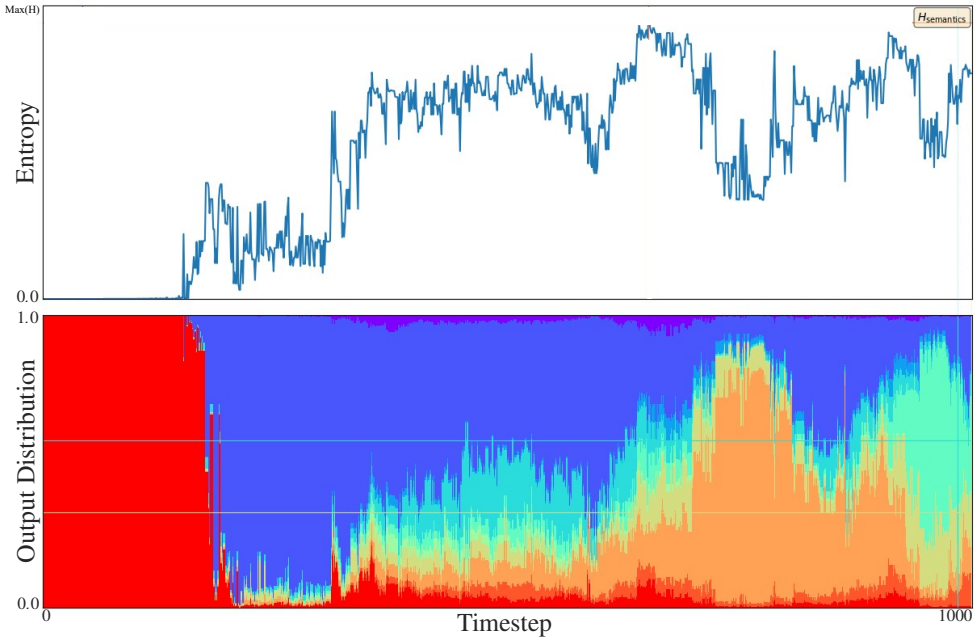


Figure E.1: A regular Flatnet MNIST classifier on random rollouts[1]. Flatnet 15.

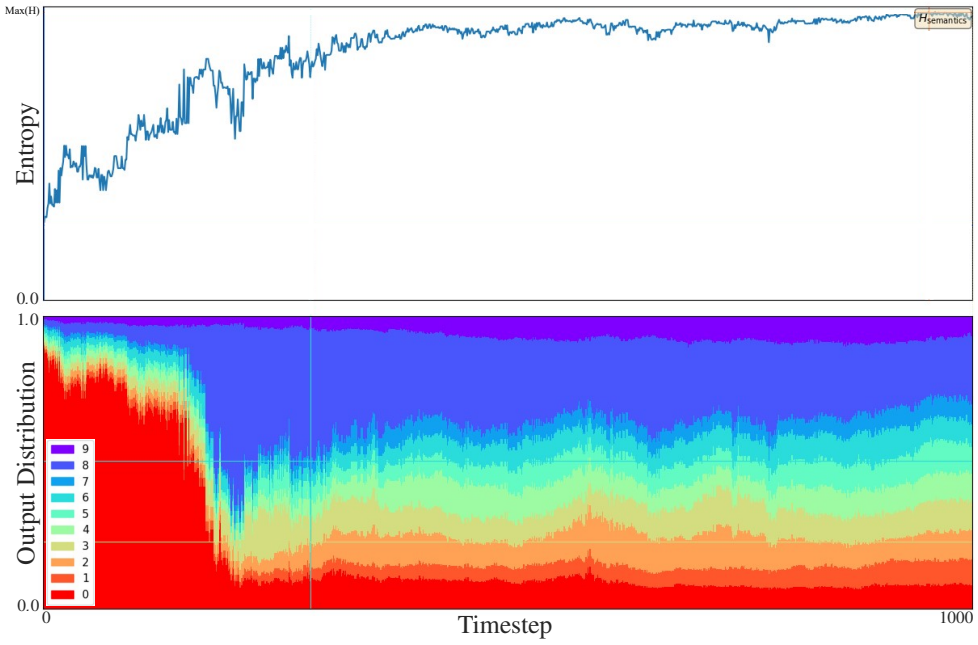


Figure E.2: An entropy regularized Flatnet MNIST classifier on random rollouts[1]. Flatnet 3.

Both, the entropy regularization and the addition of random samples, helped improve the reward trajectories of the models. The regularized models have smoother entropy transitions, a stable (less noisy) inference confidence over time, and a flatter distribution with higher entropy for false positive (random) samples[1].

[1]Note that the y-axis limits on the graphs in Fig. E.1 and E.2 are $[0, \max(H)]$, so the absolute values of the two graphs are not comparable, but the maximum entropy of the regularized version is ~ 1.5 times higher than that of the unregularized version.

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(ν) $\times$ horizon(η) $\times$ n_icem_inner_iterations(ι) 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.

Thus, 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.

Table F.1: Colored rendering-time comparison.

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.2: Grayscale 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

* - 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).

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.

Additionally, we picked 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.

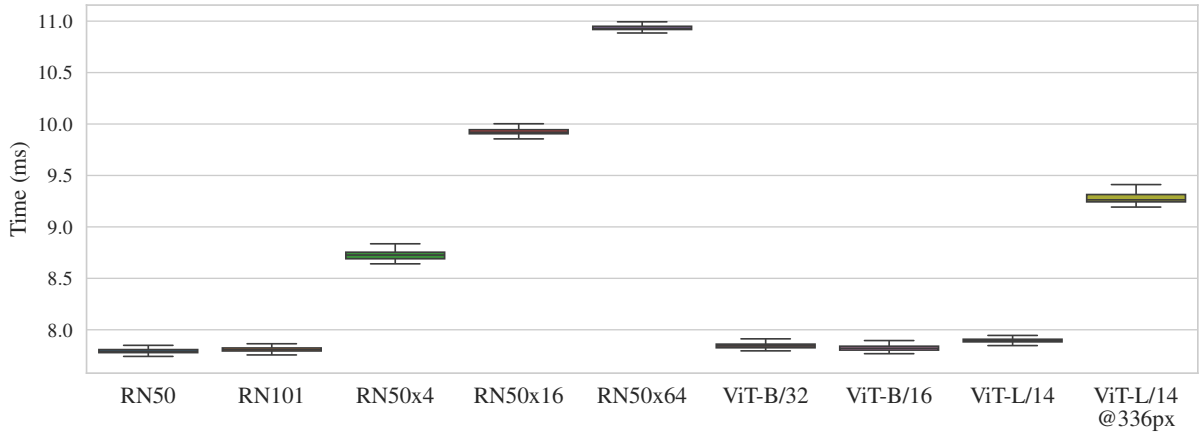


Figure F.1: Preprocessing times before optimizations.

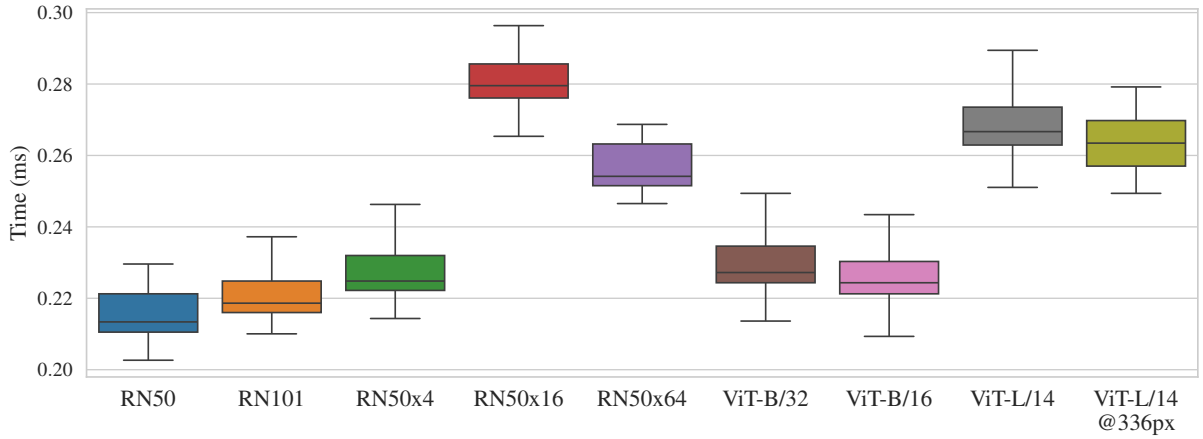


Figure F.2: Preprocessing times after optimizations.

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

Appendix G

More Simulations on ShapeGridWorld

G.1 Effect of Semantic Categories Set

The effect of the choice of categories on the semantics entropy reward in ShapeGridWorld was also significant. Fig. G.1 and Fig. G.2 show the effect of different sets of categories on the reward landscape and some samples of their creations, with and without RaIR respectively.

We observe that categories involving numbers ("numbers and letters" and "numbers") start with a flatter distribution compared to object categories (few[1] or many[2]). However, we do not notice a substantial difference in the diversity of final creations due to this even starting ground.

G.2 Effect of Grayscale Pixels

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.

It seems that the presence of grayscale pixels has a regularizing effect on the reward landscape and consequently there is less noise in CLIP. This is evident from the reduced confidence in the imperfect final creations in Fig. G.3.

G.3 Effect of Number of Pixels

As we noted earlier, the presence of a high number of controllable pixels in the ShapeGridWorld exacerbates the problem of noise in CLIP and is also a more difficult control problem, hence we also experimented with different numbers of pixels in the environment.

Having too few pixels also results in a rather noisy landscape with a lot of false positives, as seen in Fig. G.4. This seems to suggest that moderate numbers of pixels are optimal for the environment.

The effects of the number of objects, grayscale pixels, and RaIR are summarised in Fig. H.3.

[1]The categories referred to in "many categories" are "house", "bird","tree","horse", "rabbit", "goat", "shirt", "chair", "swan", "camel", "teapot", "hammer", "boot", "key", "gun", "apple", "car", "guitar", "flower", and "heart".

[2]The categories referred to in "few categories" is the subset "bird", "goat", "shirt", "swan", "goose", "teapot", "gun", "apple", "car", "airplane", "guitar", and "flower".

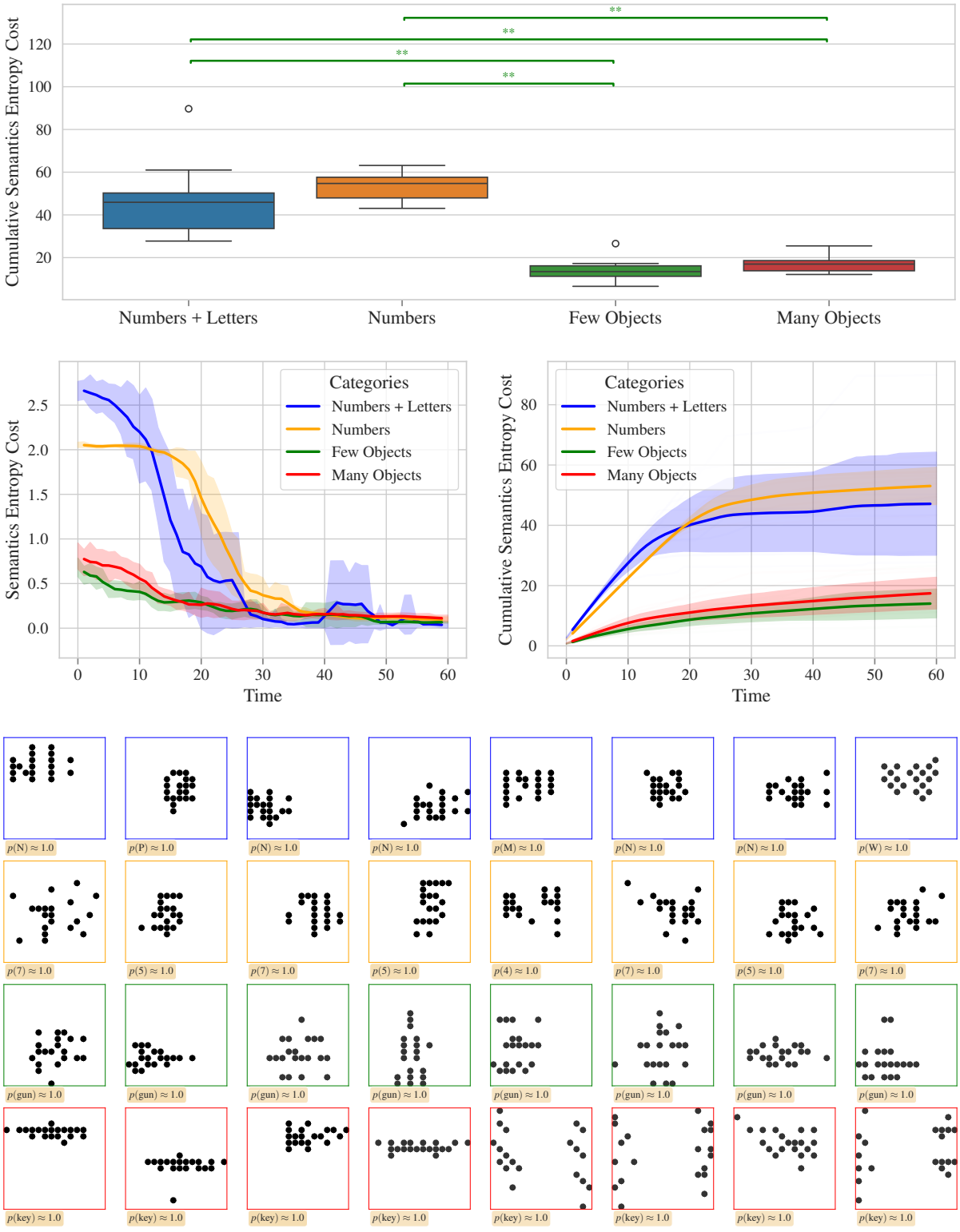


Figure G.1: Effect of categories on semantics entropy reward in ShapeGridWorld with RaIR. (10 seeds were used for each set.)

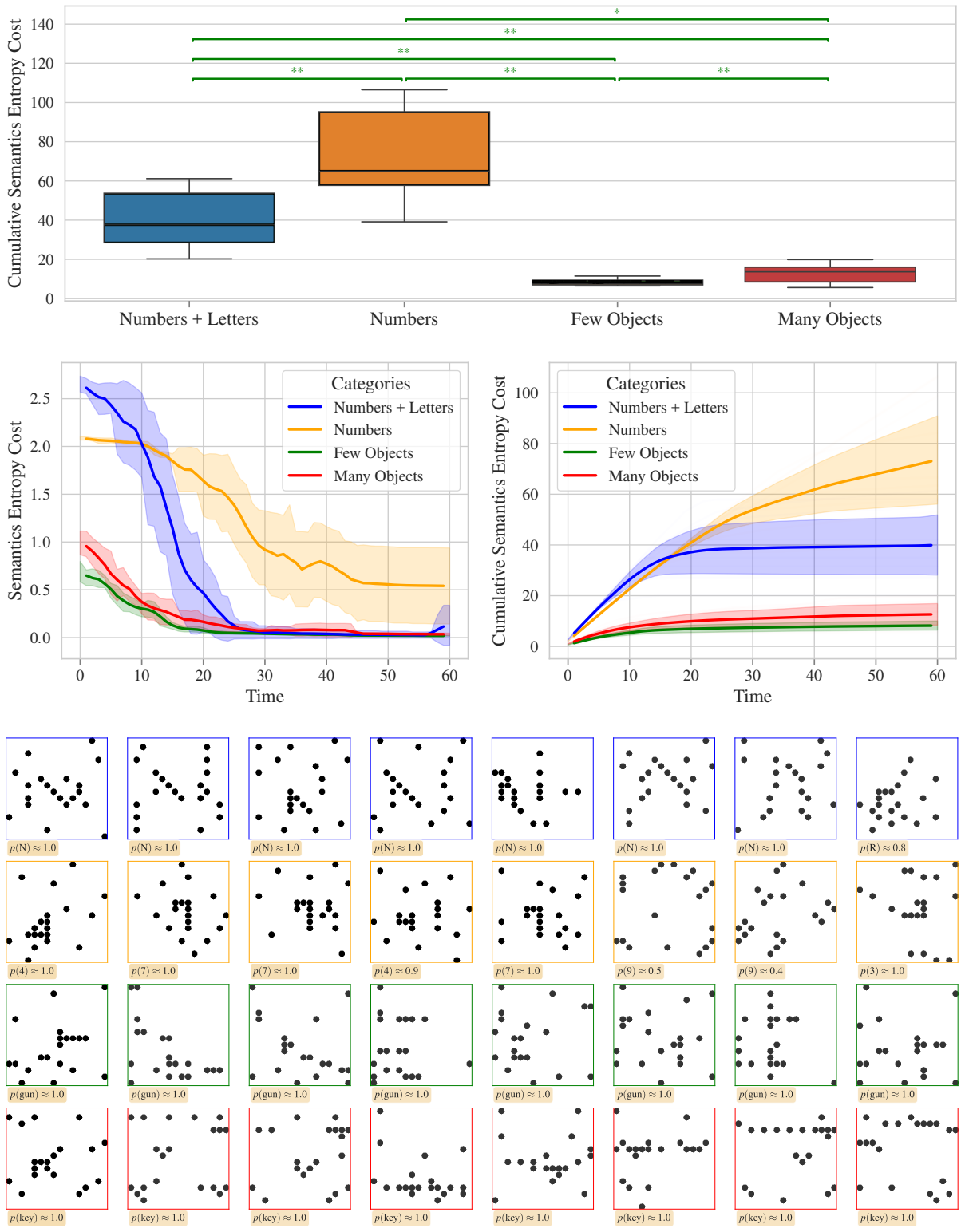


Figure G.2: Effect of categories on semantics entropy reward in ShapeGridWorld without RaIR. (10 seeds were used for each set.)

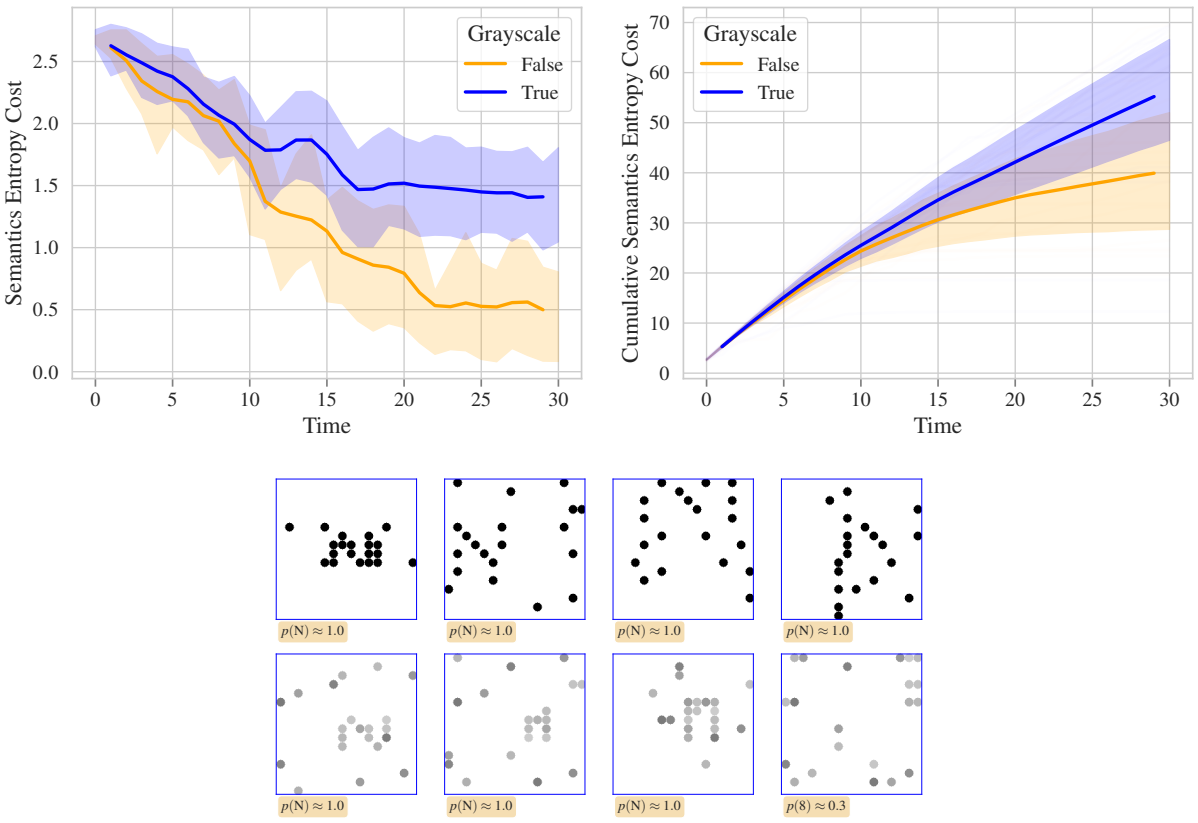


Figure G.3: Effect of the presence of grayscale pixels on semantics entropy reward in ShapeGridWorld.

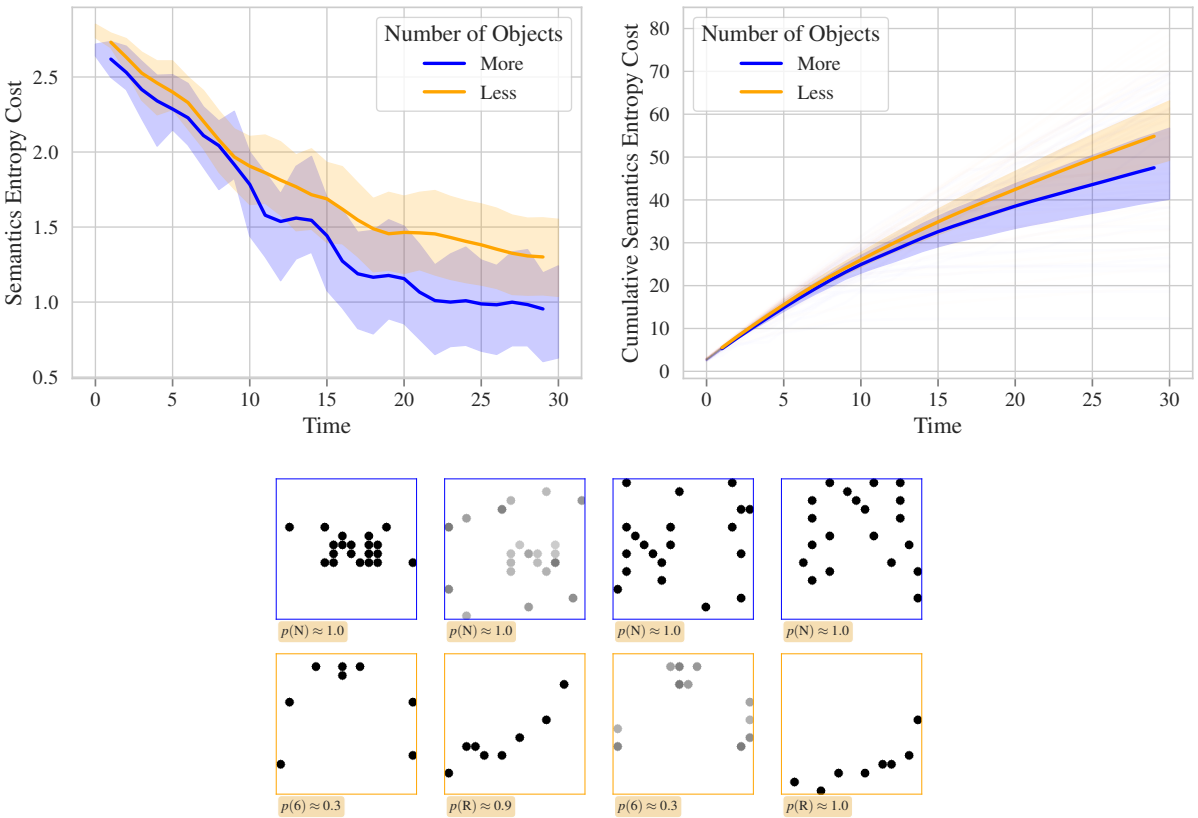


Figure G.4: Effect of the number of objects on semantics entropy reward in ShapeGridWorld.

G.4 Effect of Object Persistency

We also experimented with the "object persistency" of objects in the environment, which defines how many action steps an object is moved before the controller switches its focus on the next object in the predetermined order. This also included the case of moving all the objects together. This is related to the step size of the environment which defines how many steps in x/y-directions an object can be moved in one action.

Since the action space with moving all the objects at the same time can be very large (proportional to the number of objects), the controller needs more computational budget to realize the optimal policy, in the form of a higher planning horizon, more sampled trajectories, or more iterations to converge. We noticed that we obtained the best results when the objects were moved one at a time, only for one timestep, i.e. an object persistency of 1, with a step size around a quarter of the grid size. This is evident from the faster convergence in Fig. G.5.

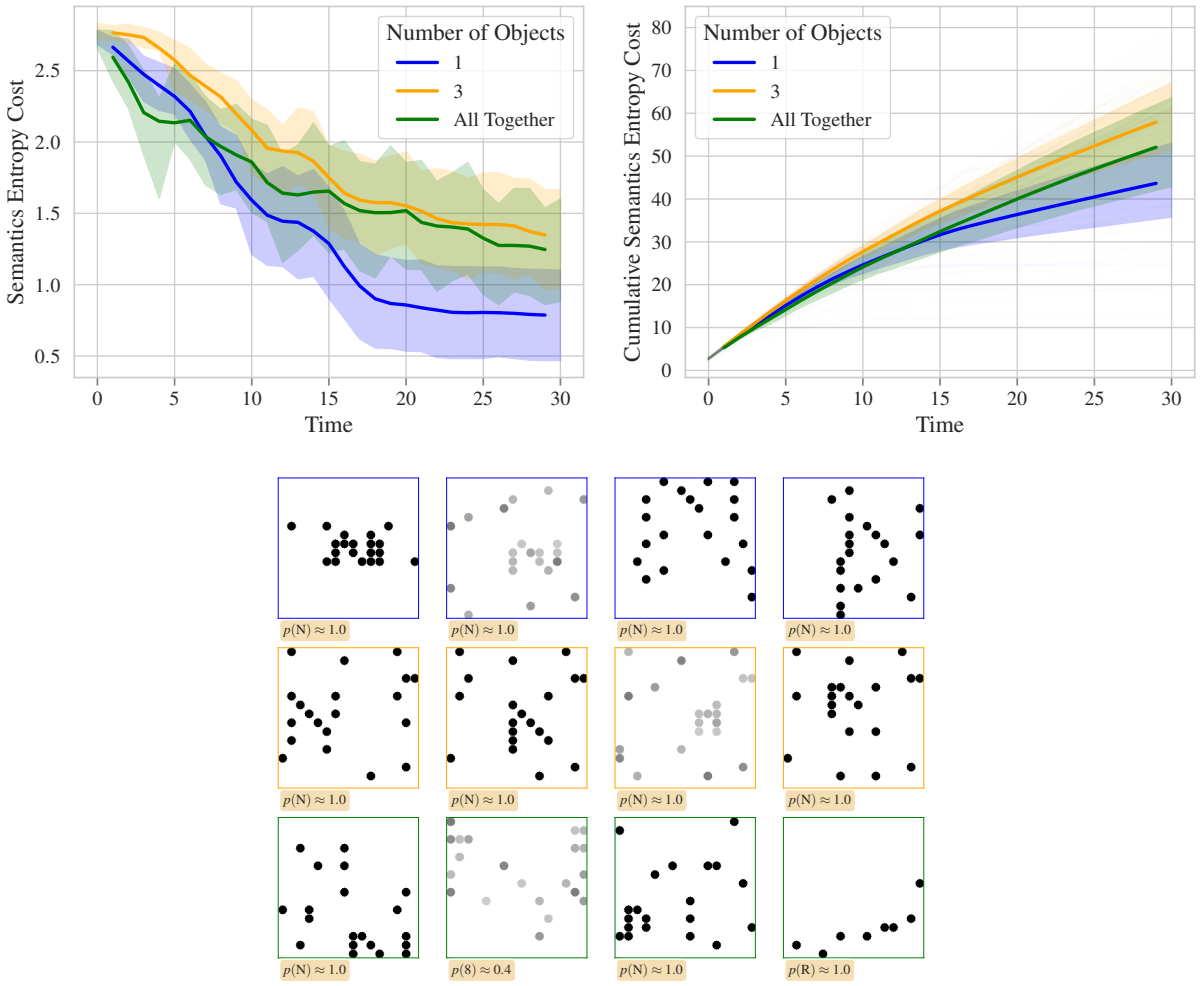


Figure G.5: Effect of object persistency on semantics entropy reward in ShapeGridWorld.

Appendix H

Additional Graphs

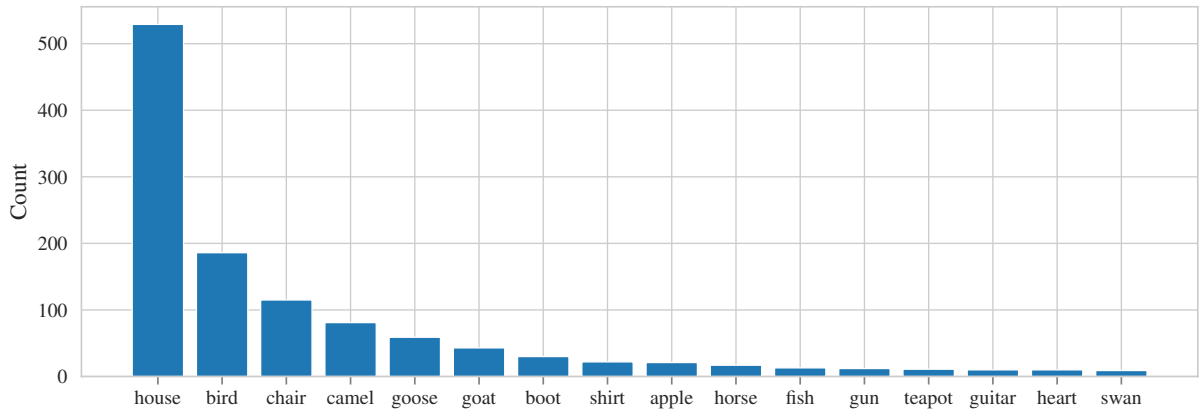


Figure H.1: Histogram of creations in our simulations on Tangram showing CLIP's class preference. The choice of classes varied in these simulations; when "`house`" or "`bird`" were present, they were chosen almost all the time. The other creations were obtained only when they were removed. Classes not mentioned in the graph include "`boat`", "`teapot`", "`gun`", "`car`", "`airplane`", "`guitar`", and "`flower`".

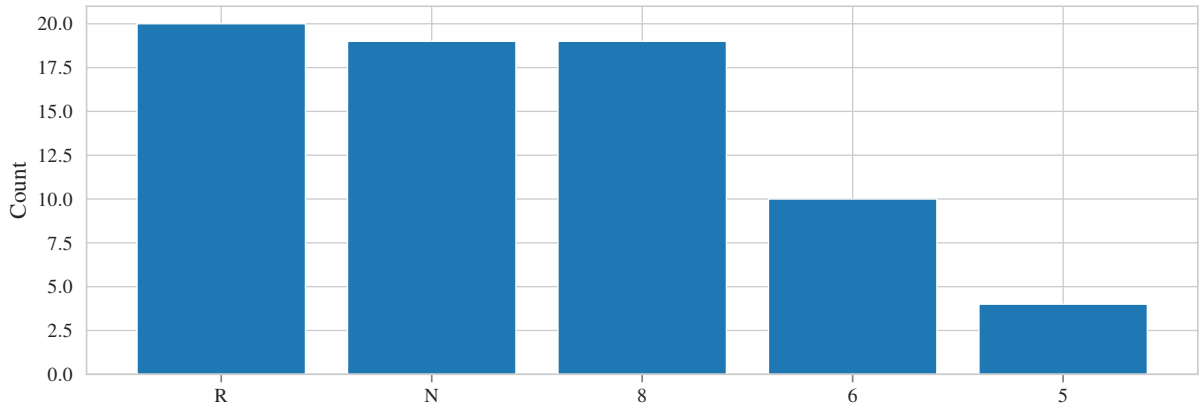


Figure H.2: Histogram of creations in our simulations on SGW showing CLIP's class preference. The 36 classes are all letters and numbers.

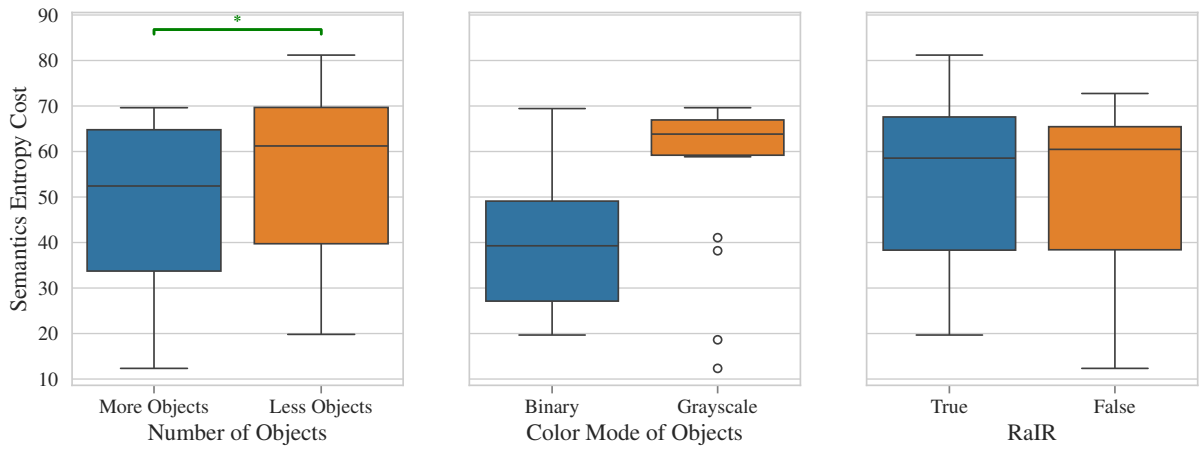


Figure H.3: Effect of the number of objects, grayscale, and RaIR on semantics entropy reward in SGW.

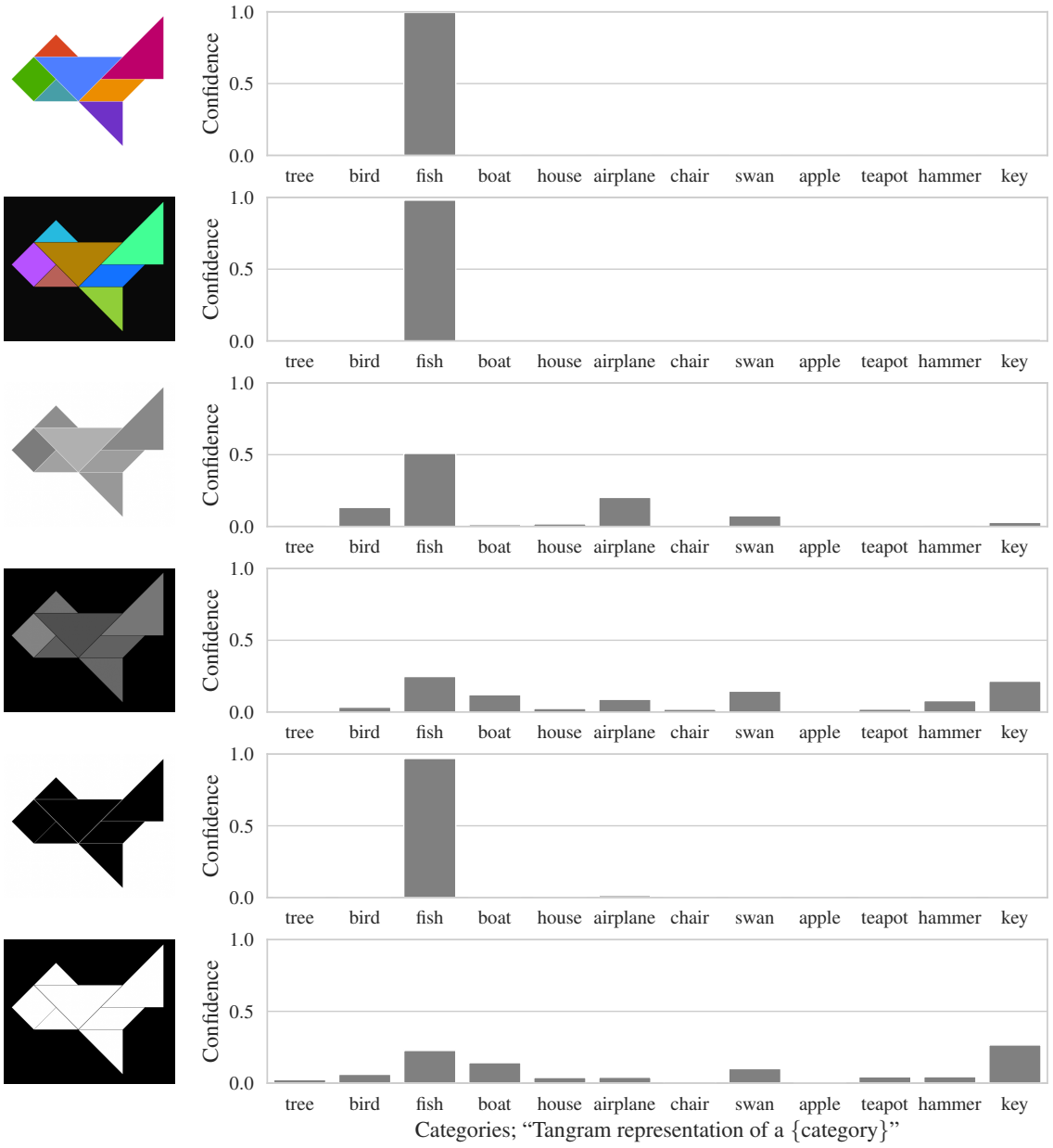


Figure H.4: CLIP on different inversions of the same Tangram creation ($\tau = 0.1$).

Appendix I

Curated Gallery

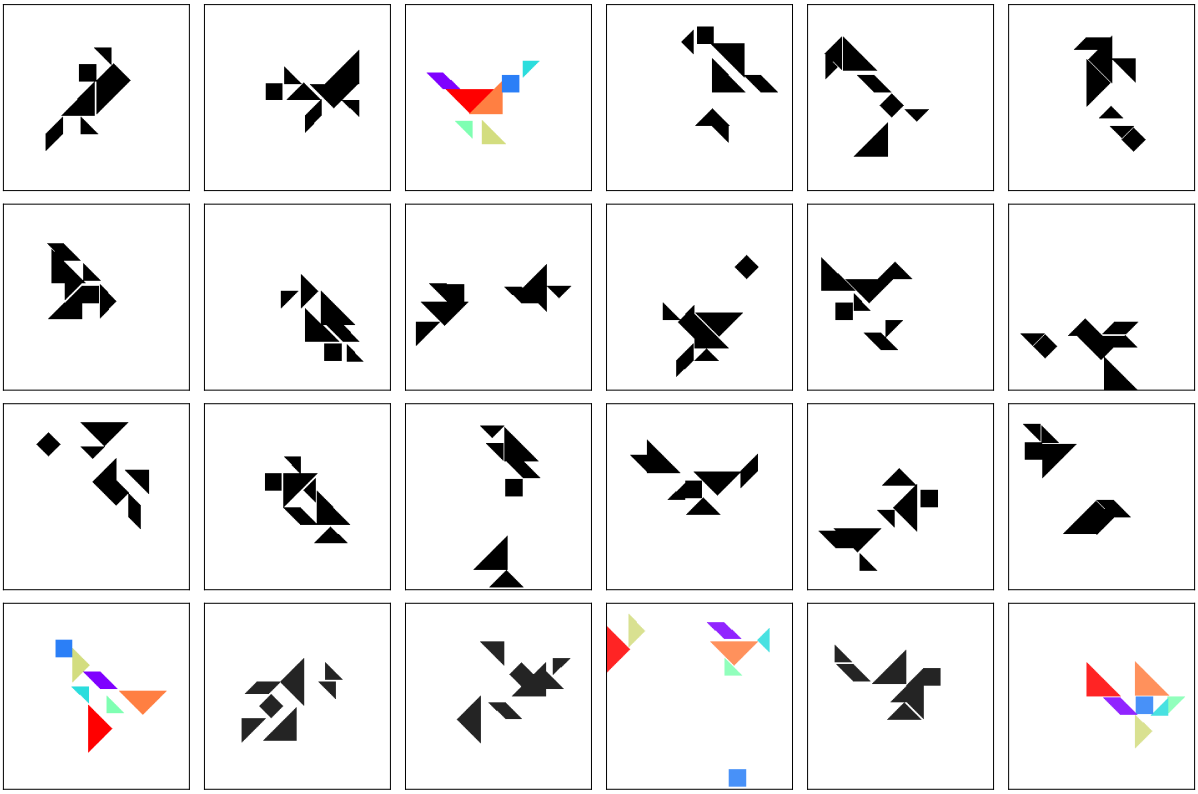


Figure I.1: Birds on Tangram.

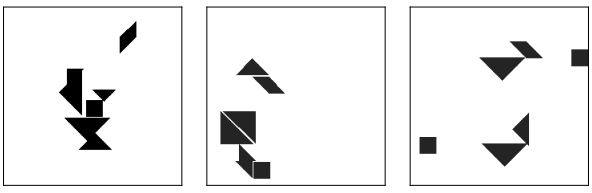


Figure I.2: Teapots on Tangram.

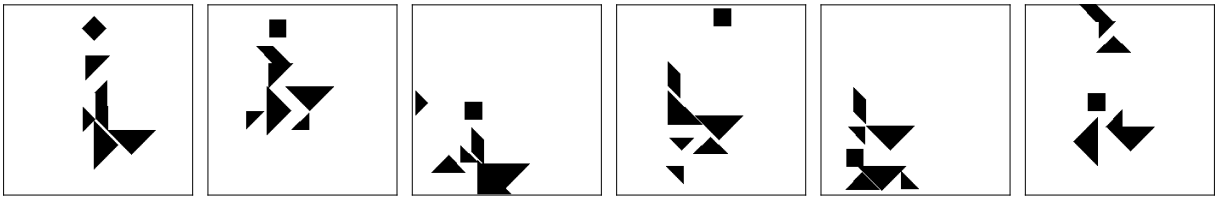


Figure I.3: Chairs on Tangram.

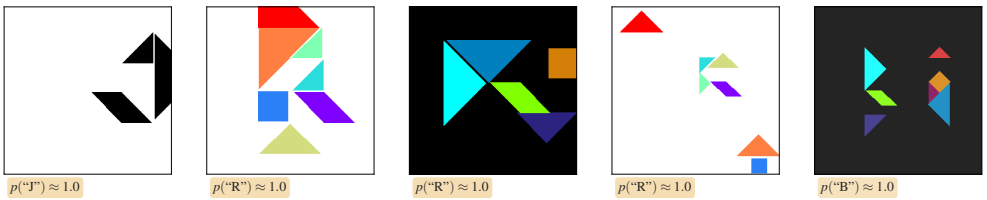


Figure I.4: Letters on Tangram.

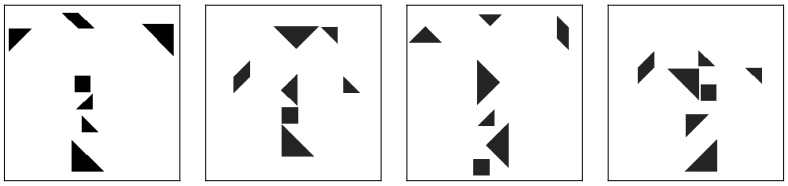


Figure I.5: Shirts on Tangram.

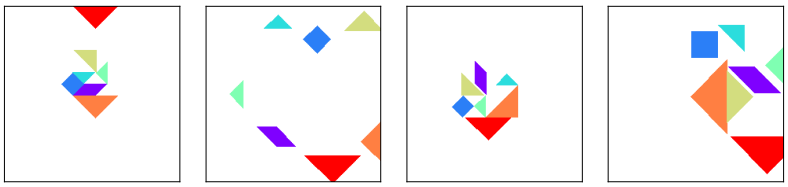


Figure I.6: Hearts on Tangram.

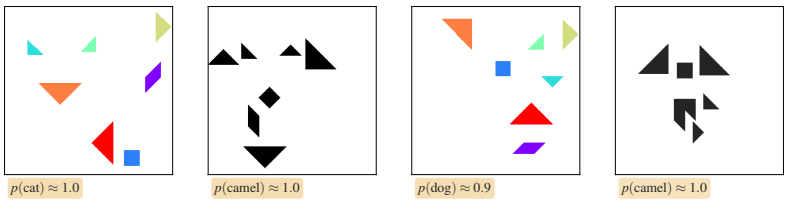


Figure I.7: Animals on Tangram. It might be difficult to see these creations as animals at first glance. Interestingly, this is because they seem to be faces of animals and not outlines as we might expect. We found these faces to have consistent subtle differences that set them apart.

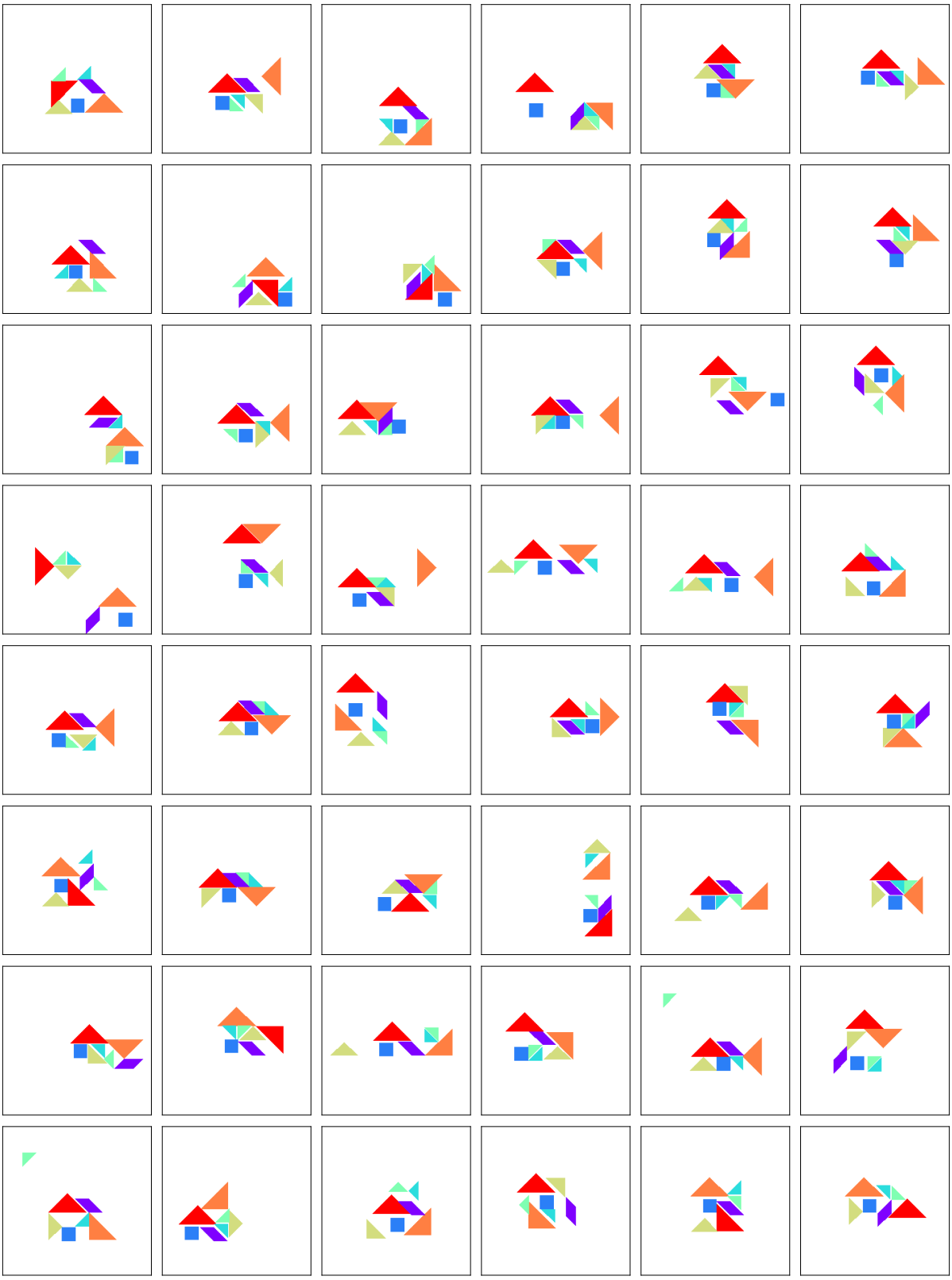


Figure I.8: Houses on Tangram.

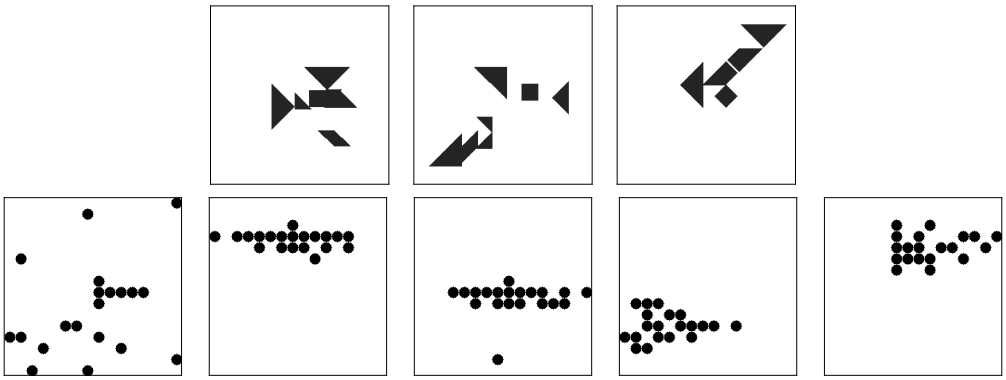


Figure I.9: Guns on Tangram and ShapeGridWorld.

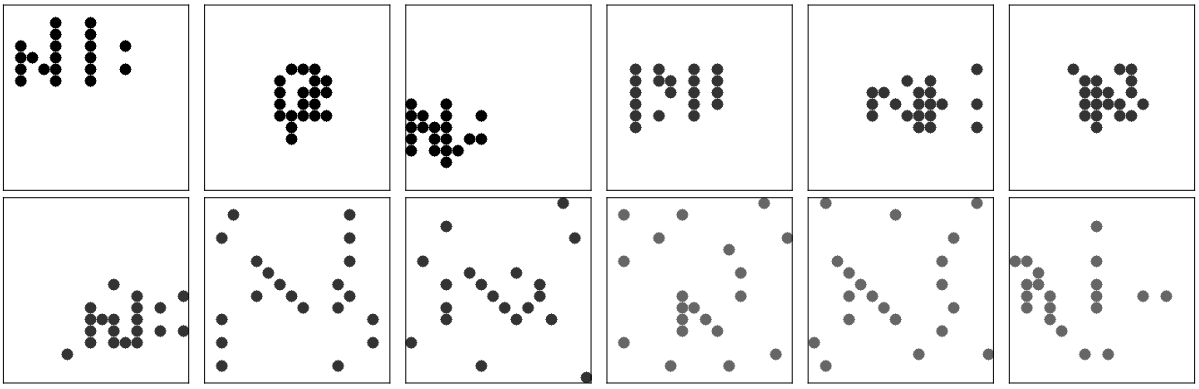


Figure I.10: Letters on ShapeGridWorld.

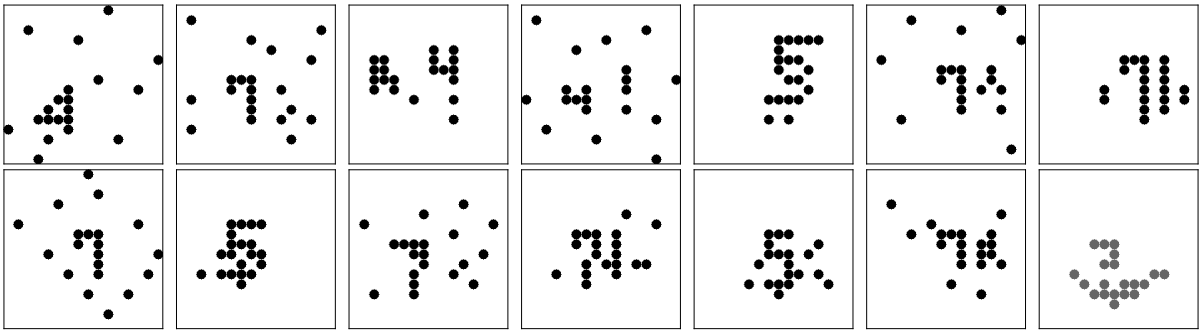


Figure I.11: Numbers on ShapeGridWorld.

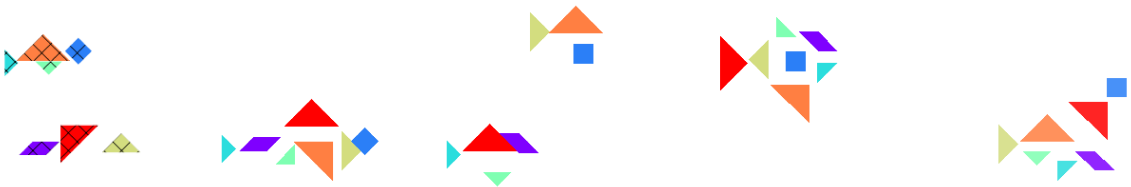


Figure I.12: Fishes on Tangram.