Counting Counts: Overcoming Counting Challenges in Image Generation using Reinforcement Learning

Shaan Gill

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-19
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-19.html

April 24, 2024

Counting Counts: Overcoming Counting Challenges in Image Generation using Reinforcement Learning

by Shaan Gill

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Trevor Darrell
Research Advisor

4/19/2024

Date

* * * * * * *

[illegible]

Professor Sergey Levine
Second Reader

4/19/2024

Date

Counting Counts: Overcoming Counting Challenges in Image Generation using Reinforcement Learning

by

Shaan Gill

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Sergey Levine

Spring 2024

Abstract

Counting Counts: Overcoming Counting Challenges in Image Generation using Reinforcement Learning

by

Shaan Gill

Master of Science in Computer Science

University of California, Berkeley

Diffusion models have quickly become state-of-the-art for high-resolution image synthesis. With an iterative forward and subsequent reverse diffusion process, diffusion models serve as a flexible tool to sequentially generate outputs on downstream objectives. Past works such as denoising diffusion policy optimization (DDPO) by Black et al. [2] have employed deep Reinforcement Learning (RL) techniques to directly fine-tune diffusion models for downstream objectives. DDPO achieves success in optimizing text-to-image diffusion models for various objectives but presents challenges in ensuring reliable semantic alignment for specific subsets of tasks. Particularly, text-to-image Stable Diffusion models prompted with text prompts of the form "N objects" fail to produce images consistent with the expected count, even after DDPO prompt alignment fine-tuning. We term this as the N-objects counting problem, characterized by the mismatch of the expected count and the count of generated objects in the image. This research serves as progress towards solving the full class of N-objects problems by focusing on a subset of the format "N [color] balls on white background". We advanced toward solving this problem by implementing several vision-informed reward functions and training models using curriculum learning techniques. Our results demonstrated that fine-tuning Stable Diffusion models under our proposed reward functions improved fidelity to the counts. Our fine-tuning resolved many empirically observed issues with the baseline model, including the wide distribution of object counts among samples prompting for N<=5. After fine-tuning, sampled images yielded a count distribution that was tightly clustered around a normal distribution, with the mean closely aligned with the anticipated count. Our curriculum learning approach improved these results with a marked difference for the complex 7-balls case. Furthermore, our solution generalized to N-objects problems beyond the subset of counting tasks we focused on. Our results convey promise for this technique as a solution to the overarching N-objects problem and for prompt-image alignment for text-to-image diffusion models.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my heartfelt gratitude to a few remarkable individuals and groups who have played a crucial role in my academic journey and personal growth.

Firstly, I want to express my deepest appreciation to Professor Trevor Darrell. Thank you, Professor Darrell, for your support and for pushing me to be the best version of myself. Additionally, I would like to thank Xudong Wang for the invaluable opportunity to conduct research under his mentorship. Working with you significantly advanced my technical skills and deepened my understanding of our field, inspiring me to conduct my research. I am also very grateful to William Lin, who has provided significant support and contributed to my research efforts. I would also like to extend a thank you to Anisha Iyer for her contributions to this research.

To my family—Papo, Mumma, Noor, Amrita, and Robin—your unyielding support and confidence in me have been the foundation of my achievements. Thank you for your endless patience, guidance, and faith in my capabilities. I could not have reached this point without your love and encouragement. You all mean the world to me.

Finally, I am grateful to all my friends, peers, the SEED Scholars Honors Program, teaching assistants, graduate student instructors, and professors who have contributed to my growth. Thank you for helping me blossom into the person I am today.

Chapter 1

Introduction

1.1 Problem

Over the last two decades, the field of generative artificial intelligence has witnessed ground-breaking progress. The development of models and frameworks like generative pretrained transformers (GPT), generative adversarial networks (GANs), and variational autoencoders (VAEs) have significantly reshaped the landscape of generative artificial intelligence across domains such as text and image generation. The ability to generate realistic and accurate images through deep learning models has marked a significant milestone. Among these advancements, Stable Diffusion models have emerged as useful tools for generating images from textual prompts, offering unbounded creative potential. However, a challenge among Stable Diffusion models is ensuring the alignment between the attributes listed in text prompts and their visual representation in generated images. Specifically, the accurate representation of count remains a critical bottleneck in the overarching alignment challenge. It influences the usefulness and relevance of the images produced by Stable Diffusion models in fields, including automated content generation, educational resources, and creative settings. This challenge is demonstrated in Figure 1.1

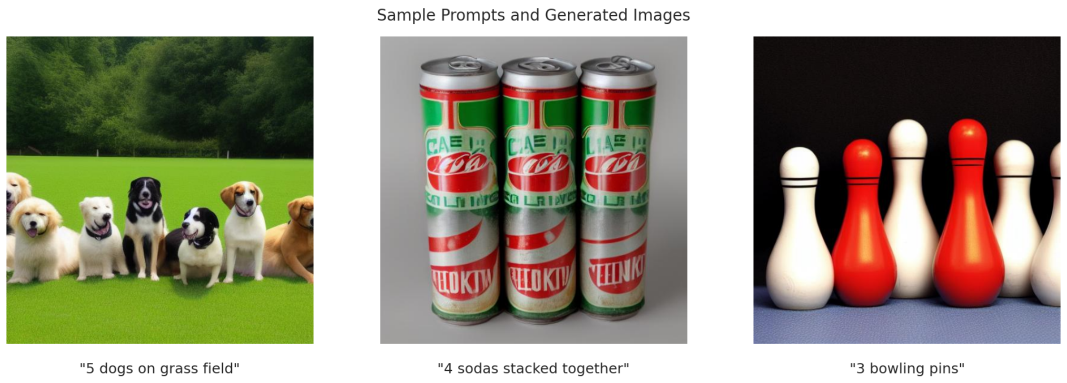

Sample Prompts and Generated Images
"5 dogs on grass field"
"4 sodas stacked together"
"3 bowling pins"

Figure 1.1: **Sample Prompts and Generated Images** The number of elements in the image does not align with the number requested in the prompt.

The seminal work by Black et al. [2] on applying Deep Reinforcement Learning (RL) to fine-tune Stable Diffusion processes constitutes a significant advancement towards solving the overall alignment challenge. This work allows for fine-tuning large Stable Diffusion models without having to curate new datasets. However, it revealed an intricate challenge: fine-tuning for prompts structured like "N [objects]" proved unsuccessful. This limitation, as identified by Black et al., stems from a mismatch in the number of objects requested in the prompt and those present in the generated image, posing a significant barrier to practical application. Black et al. noted that when optimizing for prompts following the form "N animals", denoising diffusion policy optimization (DDPO) exploits the reward function, which utilizes LLaVA (Liu et al. [9]) in conjunction with a BERT score (Devlin et al. [4]) metric, by rendering text related to the requested number in the image. As a result, the reward mean of samples increases due to the LLaVA and BERT model finding the generated images to be synonymous with the prompts, attributable to the text present in the images. This reward-hacking is depicted in Figure 1.2, where the generated images do not align with the prompts.

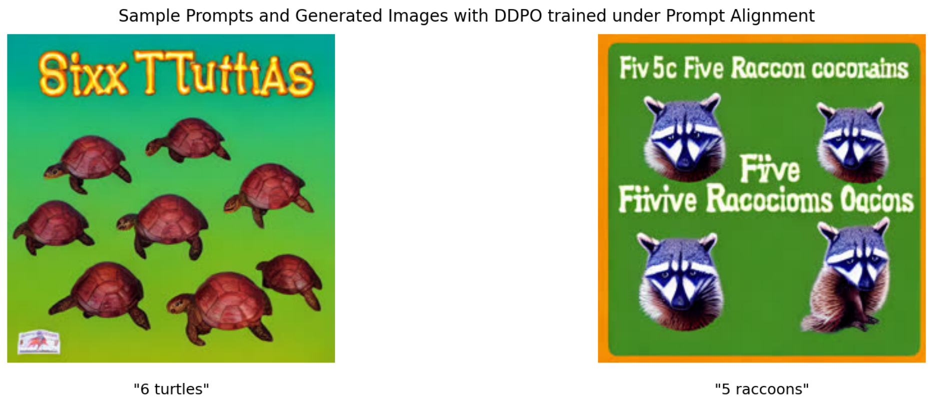

Sample Prompts and Generated Images with DDPO trained under Prompt Alignment
Sixx TTuttiAs
"6 turtles"
Fiv 5c Five Raccon cocorains
Fiive
Fiivive Racocioms Oacons
"5 raccoons"

Figure 1.2: **Sample prompts and images from Black et al. [2].** The number of elements in the image does not align with the number requested in the prompt, however, such images still attain a high reward because of the presence of words related to the requested count.

This research builds upon the foundation laid by Black et al. [2], focusing on refining Deep Reinforcement Learning approaches to address the task of object counting in image generation. By optimizing the accuracy of object representation, we aim to push the boundaries of what is achievable with Stable Diffusion models, extending their applicability, and enhancing their practical value in real-world applications.

While the focus of our research remains to utilize deep RL to overcome the limitations of denoising diffusion policy optimization, which failed on prompts of the form "N [objects]", we refined our scope to a more manageable subset. We concentrated our efforts on a more fixed subset of the general problem — specifically, the task of generating images in response to prompts of the form "N [color] balls on a white background," for N in the range of 1 to 7. Figures 1.3 and 1.4 illustrate the presence of the counting issue within this subset of prompts. This decision allowed for a more precise examination of the object-counting problem within a controlled environment, enhancing the relevance of our findings. We directly addressed the counting problem by formulating reward functions based on the number of objects in the image, inferred by an object detection module. We hypothesized that by formulating reward functions that penalize any discrepancies between the number of objects specified in the prompt and the number actually depicted in the image, we can correct the denoising process of Stable Diffusion to generate the correct number of objects, regardless of application.

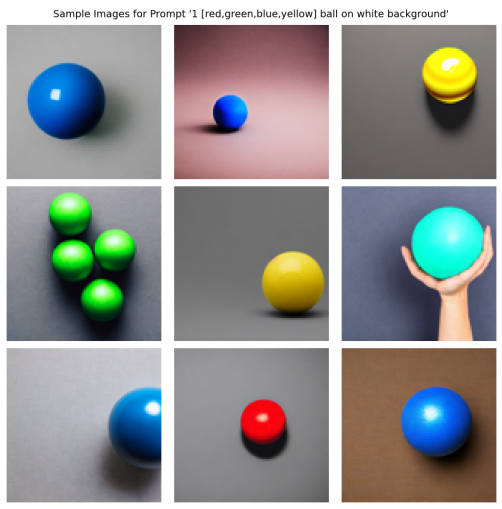

Sample Images for Prompt '1 [red,green,blue,yellow] ball on white background'

Figure 1.3: **Baseline 1 Ball Samples** Sample images generated with the baseline model for the prompt "1 [color] ball on white background."

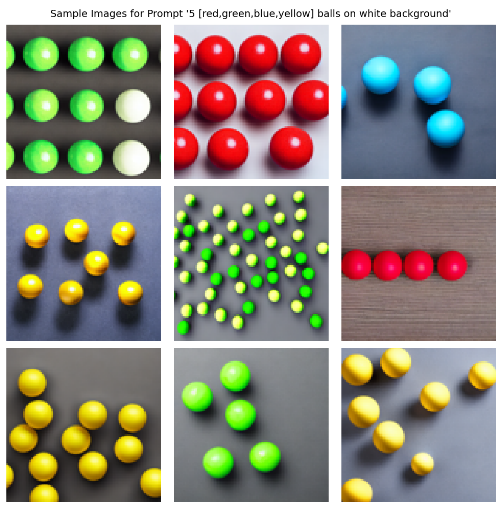

Sample Images for Prompt '5 [red,green,blue,yellow] balls on white background'

Figure 1.4: **Baseline 5 Balls Samples** Sample images generated with the baseline model for the prompt "5 [color] balls on white background."

We believe that progressing towards solving this simpler count problem is a significant contribution. The results demonstrated that fine-tuning count alignment results in more accurate image generation. The resultant image sets and their distribution are smaller in variance, higher in accuracy, and have a mean and median closer to the expected count. In practice, our approach partially generalized beyond simple shapes to more complex objects, in essence, serving as a source of transfer learning. Below is some preliminary information important to our work.

1.2 Reinforcement Learning

Black et al. [2] demonstrate how problems involving sequential decision-making, such as diffusion, can be addressed as Markov Decision Process (MDP) problems. An MDP is defined by (S, A, ρ_0, R), where each element stands for a different aspect of the decision-making environment: S is the state space, A refers to the action space, ρ_0 denotes the initial state distribution, and R represents the reward function. The Markov Property, the basis for MDPs, asserts that the transition to the next state s_{t+1} depends only on the current state s_t and the action a_t, mathematically expressed as $P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$. This property ensures that the environment's future is conditionally independent of its past. Under this framework, at a given timestep t, an agent observes the current state $s_t \in S$, executes an action $a_t \in A$, earns a reward r, based on the function $R(s_t, a_t)$, and moves to a subsequent state s_{t+1}, which is determined probabilistically as $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. A policy, denoted as $\pi(a|s)$, defines the RL agent's behavior in the environment and is a probability distribution over A, given the current state s_t. As the agent acts in the environment, the states and actions can be aggregated to formulate a trajectory, $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$. Under this setup, the goal of reinforcement learning is to maximize the expected cumulative reward over trajectories sampled from the agent's policy, written as $J_{RL}(\pi)$ (Black et al. [2]):

$$J_{RL}(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T} R(s_t, a_t) \right]$$

1.3 Diffusion Models

In this study, denoising diffusion probabilistic models (DDPM) are utilized to turn noise into context-aligned images (Ho, Jain, and Abbeel [6]). More specifically, as stated by Black et al. [2], we aim to learn a conditional diffusion probabilistic model (Sohl-Dickstein et al. [12], Ho, Jain, and Abbeel [6]), denoted as the distribution $p(x_0|c)$ "over a dataset of samples x_0 and corresponding contexts c" (Black et al. [2]). Conditional diffusion probabilistic models differ from diffusion probabilistic models because they utilize the additional context c, which allows for the generation of data given the context signal. In our use case, this allows images to be generated from text prompts. As explained by Black et al. [2], we aim to learn the distribution via denoising diffusion probabilistic models. Diffusion models function by having a forward process $q(x_t|x_{t-1})$, consisting of a Markov chain of sequential diffusion steps that add noise to data (Ho, Jain, and Abbeel [6]). Subsequently, "reversing the forward process can be accomplished by training a neural network $\mu_\theta(x_t, c, t)$, with the following objective:

$$L_{DDPM}(\theta) = \mathbb{E}_{(x_0,c) \sim p(x_0,c), t \sim \mathcal{U}\{0,T\}, x_t \sim q(x_t|x_0)} \left\| \tilde{\mu}(x_0, t) - \mu_\theta(x_t, c, t) \right\|^2$$

where $\tilde{\mu}$ is the posterior mean of the forward process, a weighted average of x_0 and x_t" (Black et al. [2]). The reverse process works by generating data from the noise iteratively across timesteps, T to 0. The goal is to optimize the objective $L_{DDPM}(\theta)$, which aims at "maximizing a variational lower bound on the log-likelihood of the data" (Black et al. [2]). The denoising process results in a sample distribution denoted as $p_\theta(x_0|c)$ (Black et al. [2]).

Consequently, the objective of "denoising diffusion reinforcement learning (DDRL) is to maximize a reward signal r defined on the samples and contexts:

$$J_{DDRL}(\theta) = \mathbb{E}_{c \sim p(c), x_0 \sim p_\theta(x_0|c)} [r(x_0, c)]$$

for some context distribution $p(c)$ of our choosing" (Black et al. [2]). We assume the presence of a pretrained diffusion model, using Stable Diffusion v1.4 (Rombach et al. [11]) as the base model for experiments. We chose to employ this model because it is "a latent text-to-image diffusion model capable of generating photo-realistic images given any text input"(Rombach et al.[11]). Our novel contribution to this objective comes from our vision-informed reward functions.

DDPO

Diffusion probabilistic models have been utilized in a variety of applications, ranging from image generation to video synthesis. Black et al. [2] introduced an innovative approach for fine-tuning such models, denoising diffusion policy optimization (DDPO). DDPO is tailored to train diffusion models to meet specific downstream objectives defined by the reward function. By conceptualizing the denoising process as a multi-step decision-making task, Black et al. [2] defined a class of policy gradient algorithms that enabled them to train denoising diffusion as an RL problem.

DDPO directly optimizes the RL objective using policy gradient estimators. Under DDPO, Black et al. [2] implemented two variants: the score function policy gradient estimator, commonly known as REINFORCE, and an importance sampling estimator. The REINFORCE variant computes gradients using data generated by the current model parameters, ideal for single-step optimizations. The importance sampling variant, on the other hand, is suited for multiple optimization steps, employing trajectories from prior model parameters for gradient estimation. It is defined as (Black et al. [2]):

$$\nabla_\theta J_{\text{DDRL}} = \mathbb{E}\left[\sum_{t=0}^{T} \frac{p_\theta(x_{t-1}|x_t,c)}{p_{\theta_{\text{old}}}(x_{t-1}|x_t,c)} \nabla_\theta \log p_\theta(x_{t-1}|x_t,c) r(x_0,c)\right] \qquad (\text{DDPO}_{\text{IS}})$$

Per Black et al. [2]'s analysis, they found the importance sampling variant "...to be the most effective algorithm...likely due to the increased number of optimization steps". Particularly, this led to the importance sampling variant to achieve high rewards among samples earlier in training. Accordingly, we utilized the importance sampling variant for our experiments as well.

In their work, Black et al. [2] applied this methodology to fine-tune text-to-image diffusion models, focusing on tasks such as image compressibility, alignment, and aesthetic quality. Building on the foundation they established, our research extends their work to specifically address the challenges of accurate object counting in images, an issue briefly mentioned in their appendix.

1.4 Curriculum and Transfer Learning

Curriculum and transfer learning serve as meta-algorithms to improve the training of machine learning algorithms. Both methods of learning can be applied to complete complex or out-of-domain tasks by first introducing the agent to subtasks related to the target task. Transfer learning entails training an agent on particular source tasks to accelerate the learning of a policy for a more complex target task (Zhuang et al. [14]). This method of learning is especially useful in environments where a source task can be formulated that is similar enough, but not as complex as the target task. When the target task is too complex for just a singular source task, curriculum learning is a more advantageous approach.

In practice, curriculum learning is similar to transfer learning as it relies on training on source tasks to gain coverage and knowledge to complete a target task. However, curriculum learning involves creating a directed acyclic graph of tasks (Narvekar et al. [10]). Each node represents a set of tasks, a module, and the edges determine the sequence of learning. Modules can range in complexity from exploration-level modules, focusing on exploring a given environment, to task-level modules. The formulation of a curriculum involves defining modules and training using some order of those modules. We define the simplest form as Sequential Curriculum Learning, where the modules across episodes of training are linearly related in complexity. Selecting an appropriate curriculum graph is critical for the effective transfer of learned behaviors and knowledge across task domains. In theory, curriculum learning enables the sequential transfer of policies and knowledge from the various subtasks. We found both transfer learning and curriculum learning to be relevant to our research experiments.

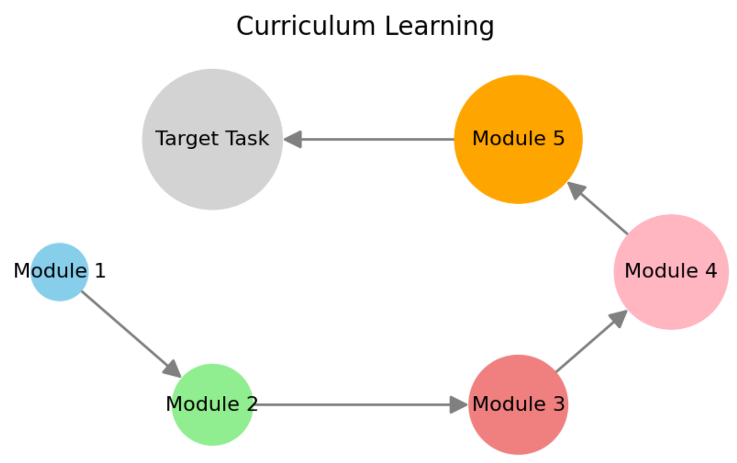


Figure 1.5: This diagram represents the sequence of learning modules in a curriculum learning plan. Each node represents a learning module, with the size of the node indicating the complexity of the module and directed edges representing the progression from one module to the next.

1.5 Object Detection and YOLO

Object detection in machine learning enables a variety of applications. Modern object detection methods rely on deep learning, and given an image, provide the user with a set of bounding boxes, class labels, and confidences. The task of object detection is typically a supervised learning problem but has a wide history of alternative approaches, including breakthroughs in unsupervised methods. The object detector we employed in our reward functions was YOLO: You Only Look Once (Fang et al. [5]). We also tested two other object detectors: CutLER (Wang et al. [13]) and DETR (Carion et al. [3]). The metrics we used for selecting our object detector were speed and accuracy. Speed was required as its use in reward functions requires that it be able to generate a proper reward very quickly in the training of the model. YOLO proved to be significantly faster than the other two methods. We assessed accuracy by creating a set of images, each generated under the prompt “5 [color] balls on a white background”. Using each object detector, we calculated the counts of balls in these images and compared the distribution of these detector-generated counts to a human-labeled count distribution. Once again, we found that YOLO performed the best.

The underlying architecture of YOLO utilizes a convolutional neural network that predicts both bounding boxes and corresponding class probabilities (Fang et al. [5]). This approach allows YOLO to look at the whole image only once and detect objects more efficiently than methods that predict objects individually or via multiple scans. YOLO divides the input image into a grid and each grid cell predicts bounding boxes and scores that indicate the possibility of objects of a given class (Kundu [8]). These scores are determined by how confident the model is that a box contains an object in addition to how accurate it thinks the bounding box is (Kundu [8]). "The model is trained using a 'bipartite matching loss' which compares the predicted classes and bounding boxes of each of the K=100 object queries to the ground truth annotations (so if an image only contains 4 objects, 96 annotations will just have a 'no object' as class and 'no bounding box' as bounding box). The Hungarian matching algorithm is used to create an optimal one-to-one mapping between each of the N queries and each of the N annotations. Next, standard cross-entropy (for the classes) and a linear combination of the L1 and generalized intersection over union loss (for the bounding boxes) are used to optimize the parameters of the model" (Abai [1], Carion et al. [3]).

We utilized a YOLO model fine-tuned for balloons using the Matterport Balloon Detection dataset (Abai [1]). The balloon fine-tuning enabled the model to detect spherical objects with higher accuracy. As a result, YOLO became our preferred choice for the object detection task.

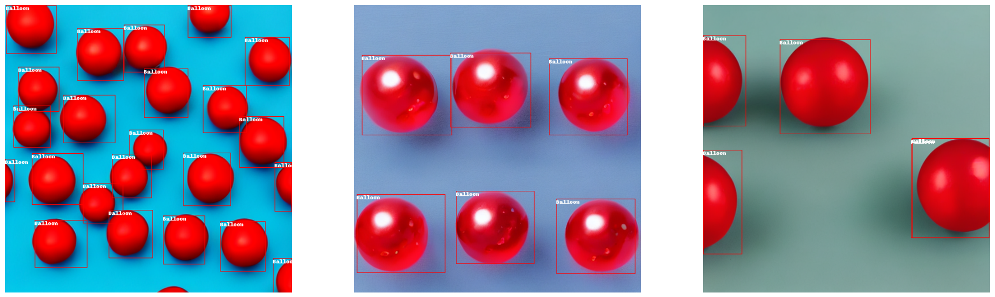


Figure 1.6: **Object Detection Results**. Results of object detection are evidenced by the rectangular outlines to identify detected balls of various sizes and fragments. We found these bounding boxes to be consistent with human interpretation.

Figures 1.7 and 1.8 show that the distribution of counts using YOLO and hand counts were similar, albeit with some minor differences. Despite these differences in outputs, these results determined that YOLO's performance was sufficient.

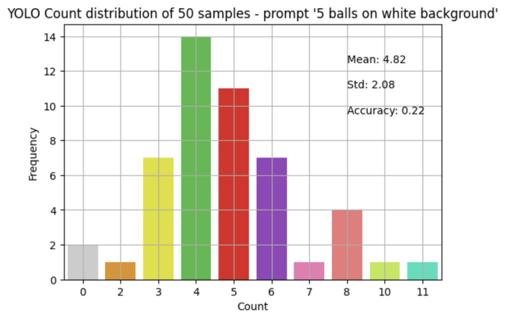


Figure 1.7: **YOLO Identification of Ball Counts** Distribution of ball counts identified by the YOLO object detection model.

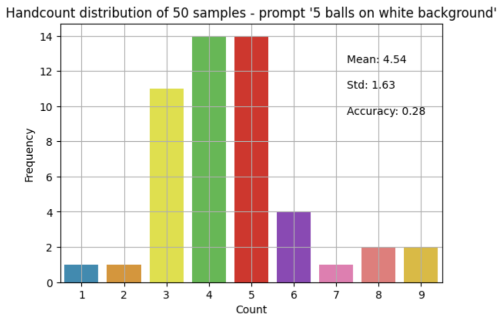


Figure 1.8: **Hand Counted Ball Identification** Distribution of ball counts identified through manual counting for comparison.

Object detection in our research served the purpose of identifying the number of balls in the images rendered by the diffusion model we fine-tuned. Object detection on generated images becomes an interesting problem, as Stable Diffusion can often generate "in-between" images that don't fully match any class. This issue meant that the counting problem for more complex image prompts, such as the prompts "N turtles" and "N raccoons" that we had seen previously, is unfeasible to approach with a local object detection-based reward function. Although approaches for detecting such objects and figures exist, such as Segment Anything (Kirillov et al. [7]), we were constrained by our computational resources. Thus we decided to focus on a subset of the problem, balls, aiming to find generalization among results. The detected number of balls and their locations were used in the reward function to compare the number of balls generated to the number of balls demanded by the prompt. Additionally, the bounding box and confidence information informed us of how reliable the ball prediction was. For instance, a ball with a bounding box deviating significantly from a square would not be considered a proper ball by our standards as it should not be likely for a spherical ball to be in a bounding box that has a significantly larger width than height or vice versa.

Chapter 2

Methods and Experiments

The main component of our study's RL approach was the design and implementation of specialized reward functions. These functions were formulated to align with our primary downstream objective: accurately producing the correct count of objects.

The reward functions consisted of two parts: a processing portion, which takes the output of the object detection network and calculates the number of identified balls based on the validation of bounding box dimensions, and a reward function portion that acts as a function of the expected count. The processing portion's main purpose was to take the list of bounding boxes that meet the confidence threshold criteria and remove significantly overlapping bounding boxes, which happened when the object detector erroneously identified one ball twice. We determined a confidence threshold of 0.90 to be appropriate for our experiments because we wanted to consider only reliable detections in our function to ensure the integrity of our detected count. This high confidence threshold helped minimize the chances of false positives, where the object detection network might have mistakenly identified non-ball objects, like shadows, as balls. The processing function then sorted the remaining balls into "good balls" and "bad balls". Good balls are the default while bad balls are those that only partially appear, for instance being cut off by the boundary of the image. We screened these by considering the aspect ratios of the bounding boxes provided by the YOLO network, and if a bounding box deviated significantly from 1 we considered the ball a bad ball. Accordingly, the processing portion returns a list of good balls and a list of bad balls. The reward calculation portion then relied on a function of the three inputs: the prompt, the list of good balls, and the list of bad balls. Our reward function framework was pivotal for testing our hypothesis, as it allowed us to train the model in varying ways. Much like the original DDPO implementation, these rewards were not used as-is, rather they were normalized for training.

We tested seven different reward function methods, all under the general framework of comparing the total number of balls (the sum of the number of good and bad balls) with the expected number. Each function had a maximum reward of 100, which indicated that the number of balls in the image aligns perfectly with the expected number of balls. Our experiments involved assigning models a specific prompt structure, "N [color] balls on a white background," and utilizing a reward function, from the seven we curated, for training. N was predetermined for each model as one of the following: 1, 3, 5, or 7. The colors used for these prompts were drawn at random from a set including red, green, blue, orange, and yellow. The rationale and differences between each of the seven reward functions will be discussed in the subsequent sections.

A challenge became apparent when prompting for images with larger values of N, particularly with N set to 7. When prompted with a smaller N, the model would generate counts that were mostly around the mean with reasonable variance. However, when N was greater than 5, the count distribution would have a much higher variance. Achieving accurate counts for larger Ns proved difficult, suggesting the natural progression in difficulty would be creating a larger number of balls. This insight led us to adopt multiple curriculum learning approaches. We structured our curriculums to start off with simple tasks, such as generating small counts or focusing on color, before moving to training on more complex tasks, such as generating larger counts. By methodically increasing the complexity of modules, we aimed to improve the model's capacity for accurate count generation. The curriculum learning approaches will be discussed in depth later in this section.

2.1 Baseline and Ablations

Our research aimed to show that RL fine-tuning for count is effective, hence we created a baseline to compare to. In Stable Diffusion models, the number of timesteps T plays a critical role in image generation and quality. We conducted several ablations of hyperparameters, specifically focusing on the effects of training diffusion timesteps on training, reward, and image quality. Figure 2.1 illustrates the reward mean of samples across epochs when training on the prompt "5 [color] balls on white background" for T=[10, 20, 35, 40, 50]. All timesteps performed sufficiently well for many epochs, illustrated through the increasing reward mean. We did notice that the T=50 case fails after 250 epochs, but we attributed this to too high of a learning rate. If timesteps were fundamental for generating prompted counts, we would expect to see differing reward mean curves as more timesteps would allow the model to generate more prompt-aligned images. In that case, the model with T=10 would never learn or increase in reward mean because it would never have enough timesteps to generate 5 ball instances. We would also expect the T=50 to start at a higher reward mean if timesteps were critical for count generation. This would imply that the count problem could be solved with more timesteps alone. However, this was not the case, and the ablation demonstrates that counts emerge very early in timestep enumeration, which is why all training curves start at a similar reward mean and are increasing. When looking at sample images from the various timestep models, we noticed that counts converged for all the models, meaning that many timesteps are not essential to solving the counting problem since the counting inconsistency is a result of the Stable Diffusion model's misunderstanding of count.

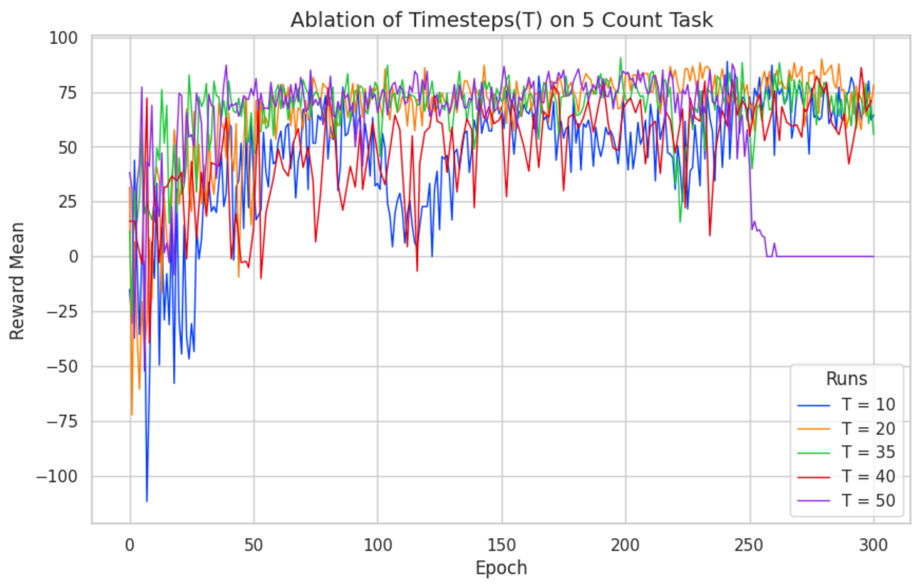


Figure 2.1: **Ablations** The plot above showcases the reward mean curves across epochs for our ablations experiment where we set diffusion timesteps T=[10,20,35,40,50]. The reward mean curves remain similar across this ablation. We utilized the Linear Reward Function for these ablation experiments.

This diffusion timestep ablation experiments demonstrated that the increase in reward mean is not a result of refinement from additional timesteps, but rather the impact of the fine-tuning on the Stable Diffusion model. We noticed that images generated with fewer timesteps had a lower resolution and no distinct foreground and background. Although fewer timesteps had a lower computational cost, because we were focused on both practical image generation and count, we found that 30 timesteps was a good balance between reward function performance, image quality, and computational cost.

As our final hyperparameters, we apply:

- Diffusion Timesteps: 30
- Learning rate: 3×10^{-5}
- Optimizer: AdamW
- Batch size: 3

- Sample Per Iteration: 9
- Epochs: 300
- DDPO Gradient updates per iteration: 1

Other hyperparameters such as the Clip Range and Optimizer β values were set to the values utilized by Black et al. [2]. Although we were unable to utilize multiple optimization steps during our training due to memory constraints, we found our model performance to be satisfactory. Our Stable Diffusion model of choice for all experiments was the CompVis Stable Diffusion v1.4 model (Rombach et al. [11]). We utilize reward mean across samples as an evaluation metric for training.

To empirically validate the effectiveness of our RL-based fine-tuning approach, we adopted a sampling-based evaluation strategy. In order to compare our fine-tuned results to the baseline model weights results, we generated two sets of 50 sample images, for prompts formatted as "N balls on white background" — one set before applying our fine-tuning training and one set afterward. These image sample sets facilitated the plotting of count frequency distributions, providing a direct means to observe and compare the impact of our fine-tuning process on the model's performance. We chose the distribution of count frequencies as our primary metric of comparison because it provides a comprehensive overview of the model's accuracy and consistency in object count generation. By comparing metrics like the variance, accuracy, and mean of the object counts across these two sets of images, we could examine how fine-tuning impacts the distribution. We empirically saw that the distributions for smaller Ns were relatively adequate, but N=5 resulted in a distribution with high variance, hence N=5 served as a significant baseline of comparison. This is evident in the top two plots of Figures 3.3, which illustrate the counts when prompted with 1 [color] ball and 5 [color] balls using the baseline model. The base Stable Diffusion model fails to generate the expected count reliably when N is larger than 1. Having established a baseline and demonstrated variability in our model's performance, we conducted experiments with several reward functions.

2.2 Reward Functions

Linear Reward Function

The Linear Reward Function was the first and best-performing reward function. This reward function begins with a base reward of 100, and linearly penalizes the absolute difference between the expected number of balls versus the total number of balls as follows: reward $\leftarrow$ reward $- 15 \times |$totalBoxes $-$ expectedCount$|$. This penalization adjusts the reward based on the difference between the total count of detected objects, *totalBoxes*, and the expected count specified in the prompt, *expectedCount*. Afterward, there incurs an additional penalty given on how many of the total balls were bad balls. This is written as reward $\leftarrow$ reward $-$ $5 \times |\text{badBoxes}|$. This part of the reward function attempts to penalize balls being generated on the edges of the image (ones that are half spheres). Thus an image that maximizes reward would be one that has the same number of balls as expected and all of those balls would be "good". We believe this function performed the best because it penalized deviations from the target count on a linear schedule, providing a direct gradient signal to the model. This reward structure may have simplified the optimization pathway, guiding the model towards higher rewards.

Ratio Reward Function

The Ratio Reward Function was intended as an analog for the linear reward function under the dynamic count curriculum learning approach. This again is a natural reward function, which is given by reward $\leftarrow$ reward $- 100 \times \left(\frac{|\text{totalBoxes - expectedCount}|}{\text{expectedCount}}\right)$. This introduces another scaling factor of the expected count. The logic behind this is that penalization should also have a dependence on how many balls were expected instead of just the absolute difference. This is because the difference between generating 2 balls when 1 was expected should be penalized more than the difference between generating 8 balls when 7 are expected. One represents a 100% increase in number of balls, while the other is only a 14%. Given this quality of the reward function, we studied this reward function under the dynamic count curriculum and regular fine-tuning. Under the dynamic count curriculum, instead of seeing a fixed number of prompted balls, we prompted for various quantities of balls per epoch of training. Similar to the Linear Reward function, we followed with an additional penalty for bad balls: reward $\leftarrow$ reward $- 5 \times |\text{badBoxes}|$.

Exponential Reward Function

We also attempted to make a reward that penalizes heavily when off by a small amount, and less severely when off by a significant margin. The penalization strategy is as follows: reward $\leftarrow$ reward $\times\ \alpha^{|\text{totalBoxes}-\text{expectedCount}|}$. Here, α is a constant less than 1, and its exponentiation by the absolute difference between *expectedCount* and *totalBoxes* forms the basis of our penalty calculation. We empirically found α=0.75 to work the best. The exponential function ensures that the biggest jump in penalty occurs when the count is off by 1, critically penalizing near-misses. Beyond being off by 1, each additional count error results in a smaller relative increase in penalty, understanding that large miscounts are more indicative of more systemic issues with the Stable Diffusion model. This penalization strategy highlights the fact that Stable Diffusion models are often off by a small count, thus penalties for small miscounts should be more significant.

Hybrid Reward Function

The Hybrid Reward Function represents a dynamic approach to penalizing deviations from the expected object count. This function was inspired by the Linear and Exponential Reward Functions. It is defined by a dual-phase penalization strategy, where for deviations larger than a certain threshold, a linear penalty is applied: $\text{reward} \leftarrow \text{reward} - 15 \times |\text{totalBoxes} - \text{expectedCount}|$. For smaller deviations, the penalty becomes exponential: $\text{reward} \leftarrow \text{reward} \times (0.75)^{|\text{totalBoxes}-\text{expectedCount}|}$. Based on empirical experiments on thresholds between 1 and 4, we established a threshold of 3. This methodology was designed to accurately reflect the severity of counting errors, applying harsher penalties for minor miscounts which seemed to be a bottleneck. The hybrid model thus adjusts its penalization mechanism based on the error magnitude, acknowledging that different types of errors have varied impacts on the model's performance.

Relative Reward Function

The Relative Reward Function is a variation of the Ratio Reward Function and focuses on penalizing the relative error in the detected object count compared to the expected count as well. The penalization formula is articulated as $\text{reward} \leftarrow \text{reward} - \beta \times \left(\frac{|\text{totalBoxes}-\text{expectedCount}|}{\text{expectedCount}}\right)$, introducing a variable β multiplier allowing for an increase or decrease in severity for deviation. We experimented with β=[80,70,50,30] and found that 50 performed the best. A more effective smaller multiplier suggested a more lenient approach to penalization, possibly to encourage exploration or to stabilize the model's training by limiting drastic jumps in reward. Additionally, the function incorporates a penalty for bad balls given by $\text{reward} \leftarrow \text{reward} - 5 \times \text{badBoxes} \times (1 + \text{relativeError})$. By scaling the penalty for badBoxes with $(1 + \text{relativeError})$, we made it such that penalties are not flat but instead grow with the relative error. This means that the more inaccurately the model counts (i.e., the higher the relative error), the more it is penalized for each bad detection. As such, this function penalizes inaccuracy and badBoxes more severely than the Ratio Reward Function.

Accuracy Reward Function

Emphasizing the importance of accurate object detection, the Accuracy Reward Function scales the reward based on the accuracy of the count. This function prioritizes exact match precision over relative error adjustments. It works on the principle that the reward should directly reflect the proportion of identified objects. The function works as follows: $\text{reward} \leftarrow \text{reward} \times \text{accuracy}$ and $\text{reward} \leftarrow \text{reward} - (\text{badRatio} \times \text{reward})$, where $\text{accuracy} = 1 - \frac{|totalBoxes-expectedCount|}{expectedCount}$, and badRatio reflects the proportion of inaccurately detected objects, $\frac{badBoxes}{\max(1, totalCount)}$. Unlike the other functions, which scale penalties based on the absolute or relative error, or adjust penalties based on deviation size, the Accuracy Function incentivizes precision in detection and ball consistency.

Adaptive Threshold Reward Function

This function adapts its threshold for penalization based on the model's historical performance: $\text{reward} \leftarrow \text{reward} - \text{adaptivePenalty} \times |\text{total} - \text{expectedCount}|$. This function calculates the deviation from the expected account, and using an adaptive threshold that gets updated across epochs, selects an adaptivePenalty. This allows the model to gradually refine its sensitivity to errors through feedback from its own performance. The reward function applies a larger penalty for discrepancies that exceed the threshold and a smaller penalty for count differences less than or equal to the threshold. This dynamic approach ensures that the model is penalized proportionally more when, after epochs of progressive learning, it produces images that resemble less refined or past fine-tuning stages. This policy is intuitive because because we expect the model's performance to consistently improve over time, without regressing to less accurate stages of image generation. This self-adjusting mechanism aims to calibrate the model's penalization threshold across epochs, encouraging a more efficient learning process tailored to the model's changing accuracy.

Emerging Patterns

Through our experiments, we observed that focusing on refining the count accuracy did not adversely affect the model's ability to accurately align with the specified colors and objects. This was a significant finding because it indicated that training for count refinement could continue without severely compromising holistic prompt alignment. However, we observed that the training under differing reward functions impacted the style of balls in the images, with some models generating more or less detailed balls. These variations will be further discussed in the Results and Discussion sections. Moreover, we did see that once we reached each training's "upper bound", subsequent training did not increase the reward mean and there seemed to be a degradation of image quality. To adjust for this, we utilized model checkpoints prior to the upper bound for our analysis and evaluation.

Pseudocode for the Reward functions

Algorithm 1 Counter Reward Function

```
Require: Object detection pipeline $obj\_detector$, Base reward $base\_reward$, Image list $images$, Prompt list $prompts$
Ensure: A list of scores corresponding to each image and prompt
  function POSTPROCESS($boundingBoxes$)
      Filter $boundingBoxes$ to remove duplicates and separate into $goodBoxes$ and $badBoxes$, $badBoxes$ being bounding boxes that have disproportionate aspect ratio
      return $goodBoxes$, $badBoxes$
  end function
  function REWARD($images$, $prompts$)
      Initialize $scores$
      for each $image$ in $images$ do
          $boundingBoxes \leftarrow obj\_detector(image)$        ▷ returns list of bounding boxes
          $goodBoxes$, $badBoxes \leftarrow$ POSTPROCESS($boundingBoxes$)
          $totalBoxes \leftarrow$ length of $goodBoxes$ + length of $badBoxes$
          $reward \leftarrow base\_reward$
          Determine $expectedCount$ based on $prompt$
          if linear then
              $reward \leftarrow reward - 15 \times |totalBoxes - expectedCount|$
              $reward \leftarrow reward - 5 \times$ length of $badBoxes$
          else if ratio then
              $reward \leftarrow reward - 100 \times \left(\frac{|totalBoxes - expectedCount|}{expectedCount}\right)$
              $reward \leftarrow reward - 5 \times$ length of $badBoxes$
          else if exponential then
              $reward \leftarrow reward \times \alpha^{|totalBoxes - expectedCount|}$
          end if
          Append updated $reward$ to $scores$
      end for
      return $scores$
  end function
```

Algorithm 2 Counter Reward Function Cont.

```
Require: Object detection pipeline $obj\_detector$, Base reward $base\_reward$, Image list
   $images$, Prompt list $prompts$, Threshold $adaptive\_threshold$
Ensure: A list of scores corresponding to each image and prompt
  function POSTPROCESS($boundingBoxes$)
      Filter $boundingBoxes$ to remove duplicates and separate into $goodBoxes$ and
  $badBoxes$, based on aspect ratio
      return $goodBoxes$, $badBoxes$
  end function
  function REWARD($images$, $prompts$)
      Initialize $scores$
      for each $image$ in $images$ do
          $boundingBoxes \leftarrow obj\_detector(image)$                ▷ Detect objects
          $goodBoxes, badBoxes \leftarrow$ POSTPROCESS($boundingBoxes$)
          $totalBoxes \leftarrow$ number of $goodBoxes$ + number of $badBoxes$
          $reward \leftarrow base\_reward$
          Determine $expectedCount$ based on $prompt$
          if hybrid then
              if $\text{abs}(totalBoxes - expectedCount) > 3$ then
                  $reward \leftarrow reward - 15 \times \text{abs}(totalBoxes - expectedCount)$
              else
                  $reward \leftarrow reward \times (0.75)^{\text{abs}(totalBoxes - expectedCount)}$
              end if
              $reward \leftarrow reward - 5 \times$ length of $badBoxes$
          else if relative then
              $relativeError \leftarrow \text{abs}(totalBoxes - expectedCount)/\max(expectedCount, 1)$
              $reward \leftarrow reward - \beta \times relativeError$
              $reward \leftarrow reward - 5 \times$ length of $badBoxes \times (1 + relativeError)$
          else if accuracy then
              $accuracy \leftarrow \max(0, 1 - \text{abs}(totalBoxes - expectedCount)/expectedCount)$
              $reward \leftarrow reward \times accuracy$
              $reward \leftarrow reward \times (1 -$ length of $badBoxes/\max(totalBoxes, 1))$
          else if adaptive then
              $deviation \leftarrow \text{abs}(totalBoxes - expectedCount)$
              if $deviation > adaptive\_threshold$ then
                  $reward \leftarrow reward - 20 \times deviation$
              else
                  $reward \leftarrow reward - 10 \times deviation$
              end if
              Update $adaptive\_threshold$ based on $deviation$
          end if
          Append updated $reward$ to $scores$
      end for
      return $scores$
  end function
```

2.3 Curriculum Learning

In our research, we observed a notable trend: as N increased, both our reward function and the Stable Diffusion model demonstrated diminishing results in generating the correct count. Specifically, our reward function failed to capture all appropriate bounding boxes for larger Ns, and the fine-tuned Stable Diffusion model began to generate extraneous image elements such as cameras, photography studios, and lighting fixtures. This observation led us to hypothesize that the complexity of generating a large number of objects might be the underlying issue. To address this, we employed curriculum learning strategies, theorizing that training with a gradual increase in task complexity could improve learning outcomes. We implemented:

- **Color-Based Curriculum**: This curriculum strategy involved a three-stage learning process focusing on color differentiation. The agent was first trained on the prompt “1 red ball on white background”, and then progressed to being trained on different colored 1 ball prompts. Lastly, it was trained on N=5 different colored balls. We conducted this curriculum learning experiment prior to training our models. It did not produce significant improvements compared to training directly on multiple different colored balls, indicating that color variance was not a bottleneck.

- **Dynamic Count Curriculum**: This curriculum involved dynamically altering the count of objects within a single training episode. Across each epoch, sample prompts had a random number of balls, between 1 and 7, and the reward function took this into account. In theory, each epoch served as a module with varying levels of complexity. However, this approach was not as effective as the learning from one epoch did not effectively transfer to the next, having no persistence of learned patterns.

- **Sequential Count Curriculum**: The most effective approach was a sequential curriculum learning strategy. In this method, we began training on a small count of multicolored balls (e.g., “1 [red, blue, green, yellow, orange] ball on white background”) and linearly increased the count across training episodes. We trained on 1, 3, 5, and finally 7 balls. This approach effectively facilitated the sequential transfer of learned knowledge across counts. The details and results of this approach are discussed in the results section.

For each curriculum learning approach, the model weights from the end of the previous training module were loaded when transitioning to the next module. It is important to note that we had to decrease the learning rate from 3e-5 to 3e-6 for adequate curriculum training for larger Ns. Our curriculum learning strategies emphasize the importance of thoroughly testing the sequence and complexity of tasks in a curriculum.

Chapter 3

Results and Discussion

3.1 Results

Our experiments achieved significant results in shrinking the distribution of generated counts for balls. Various versions of the seven reward functions were tested to see which performed the best for both our training and practical effectiveness metrics. Figures 3.1 and 3.2 depict the progression of the reward mean for sample images across training for the reward function variants.

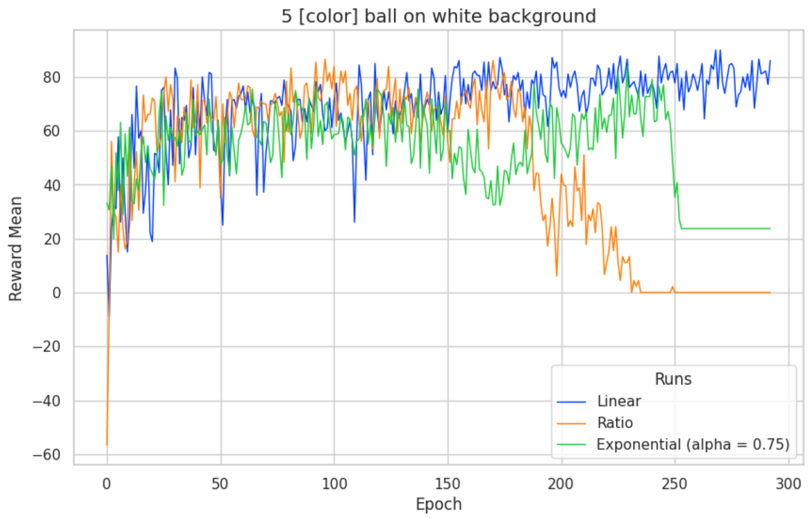


Figure 3.1: **Reward Functions Comparison** Comparison between Linear, Ratio, and Exponential Reward Functions.

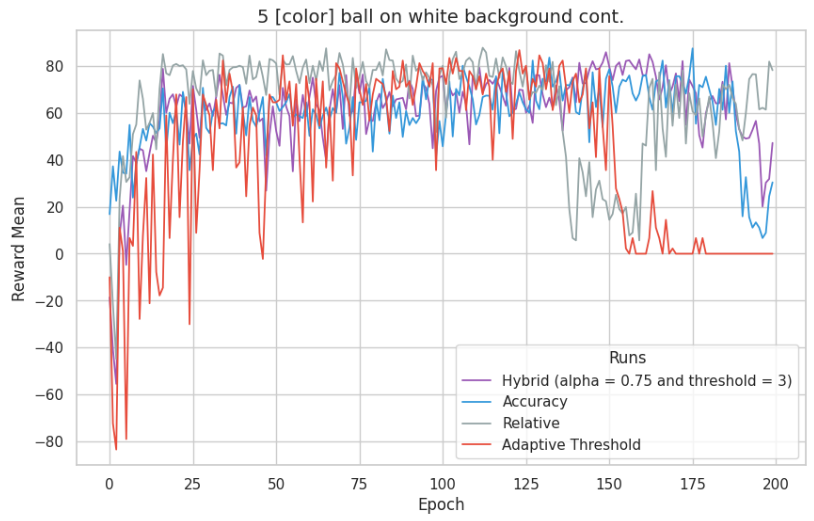


Figure 3.2: **Reward Functions Comparison** Comparison between Hybrid, Accuracy, Relative, and Adaptive Threshold Reward Functions.

Several functions facilitated constructive learning, evidenced by the increasing reward means, however, some functions performed much better than others. It is important to note that, because we had different penalization strategies across reward functions, the reward mean across epochs alone is not an adequate indicator that reinforcement learning is effective for resolving the N object problem. The sampling-based evaluation metrics and distribution serve as better indicators of the impact fine-tuning has on model performance.

Specifically, the mean, median, and variance are important statistics in evaluating our approach's performance. A mean or median that is close to the expected count demonstrates a model's ability to consistently generate images with accurate counts. In an ideally fine-tuned model for N=5, the mean and median would both be 5. This is because such a model should generate samples whose count distribution resembles a tight normal distribution. This would imply that, on average, the model successfully follows the numerical count given in the prompt. Furthermore, a low variance and standard deviation indicate stability in a model's performance, ensuring that the generated counts are not only accurate but also consistent upon multiple inferences. Our results demonstrate a decrease in mean and variance for both the N=1 and N=5 cases across several reward functions. Refer to Figure 3.3 and Table 3.1 for statistics for the N=5 case. We found that when fine-tuning for $N > 5$, model performance diminished significantly, hence we utilized a curriculum learning approach to achieve adequate results for N=7.

Table 3.1: Statistical Summary of Model Performance by Reward Function for Prompts with N=5

Function	Mean	Median	Variance	Standard Deviation	Accuracy
Baseline	9.64	7.50	79.54	8.92	4.00
Linear	4.74	5.00	2.56	1.60	32.00
Ratio	5.28	5.00	4.04	2.01	30.00
Exponential	3.24	3.00	3.53	1.88	10.00
Hybrid	8.22	4.00	435.32	20.86	10.00
Relative	4.58	4.00	4.70	2.17	18.00
Accuracy	9.42	4.50	436.29	20.89	14.00
Adaptive Threshold	4.24	4.00	4.88	2.21	12.00

Per our statistical evaluation, the Linear Reward Function proved to optimize evaluation criteria the best, massively shrinking down the baseline distribution and shifting the count mean from 9.64 to 4.74 and median from 7.5 to 5 when prompted to generate 5 balls. Additionally, the linear model displayed effective training, as demonstrated by a reward mean that remained stable and never seriously declined. The linear function also showcased the lowest variance among all tested functions. This low variance indicates that its learned representation of count best aligns with our expectations. We also note significant increases in accuracy from 82% to 98% for the 1-ball case and 4% to 32% for the 5-ball case when the linear function was utilized for fine-tuning. Refer to Figures 3.3, 3.4, and 3.5.

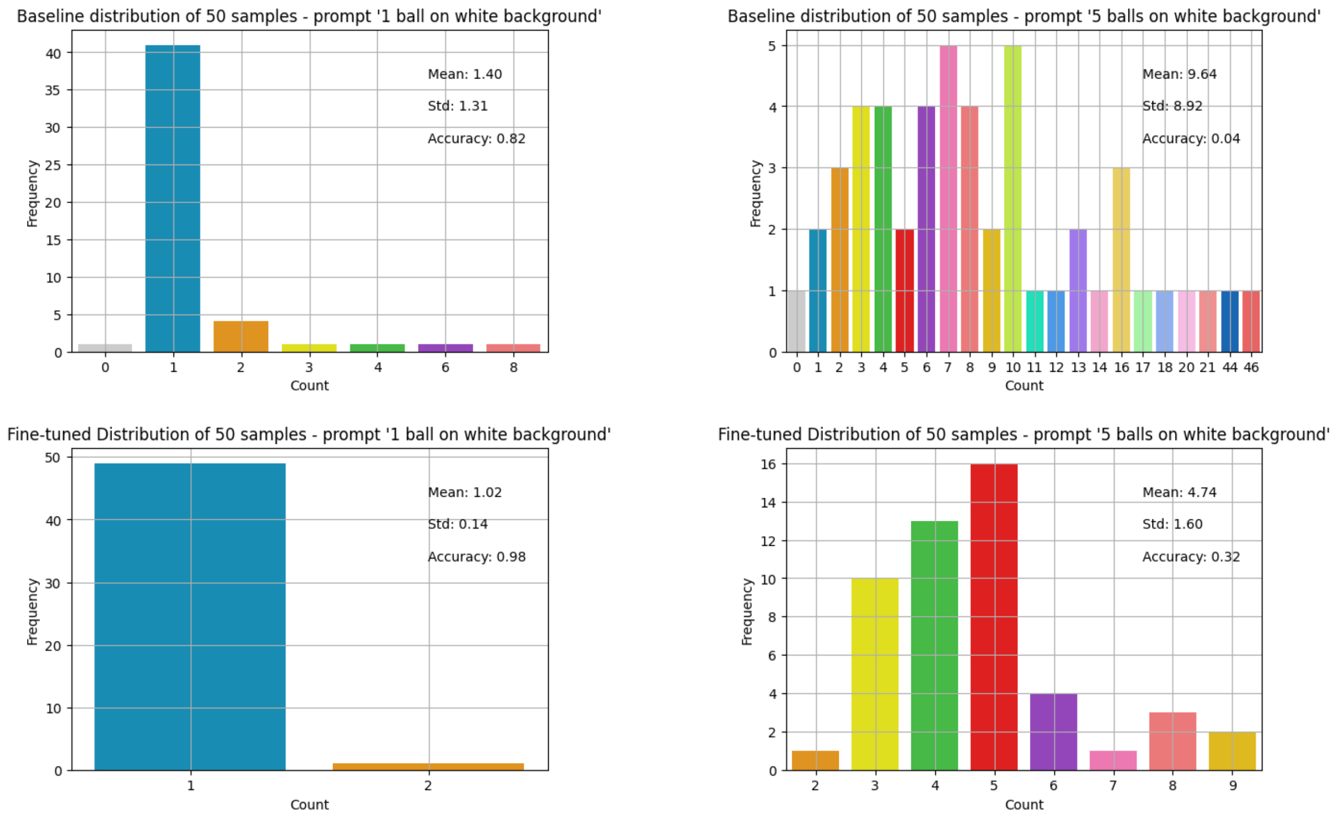


Figure 3.3: **Distribution of Ball Counts for Fine-tuned Model** Top: Distribution of ball counts in 50 samples from the baseline Stable Diffusion model for both the N=[1,5] cases. Bottom: Distribution of ball counts in 50 samples from the fine-tuned Stable Diffusion model for both the N=[1,5] cases. Results from the trained distribution have a lower variance and are closer to the expected mean.

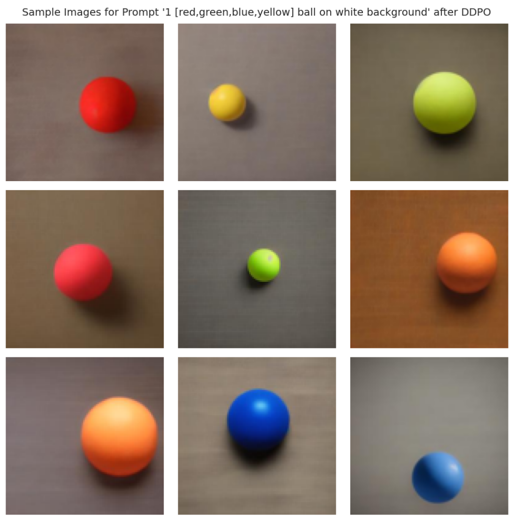

Sample Images for Prompt '1 [red,green,blue,yellow] ball on white background' after DDPO

Figure 3.4: **Fine-tuned 1 Ball Samples** Sample images generated with the fine-tuned model for the prompt "1 [color] ball on white background." We did notice that the background for many images had more of a brown or grey hue, but this was also persistent in the baseline model samples, hence we treat it as a constant.

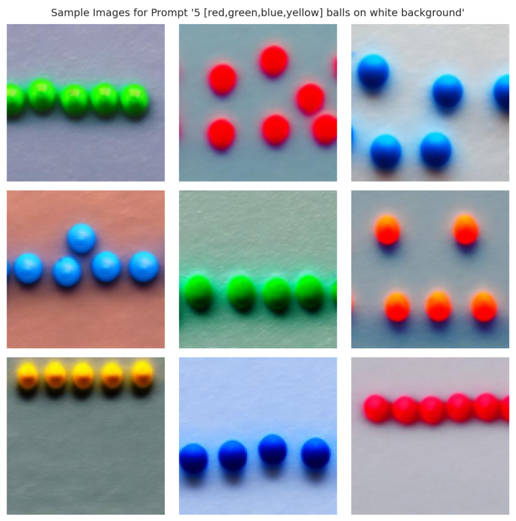

Sample Images for Prompt '5 [red,green,blue,yellow] balls on white background'

Figure 3.5: **Fine-tuned 5 Balls Samples** Sample images generated with the fine-tuned model for the prompt "5 [color] balls on white background." We did notice that the background of fine-tuned samples was more white in hue compared to the baseline.

Although the linear function performed the best statistically and was robust in training, we observed a slight haziness around each ball in the generated images. This did not undermine the efficacy of the reward function, but it prompted us to investigate potential causes for the haziness, particularly the influence of samples per epoch on image fidelity. In the training context, we focus on samples per epoch since our experiments utilized a single optimization step across the generated samples. In the original DDPO framework, a batch size of 64 was used with 4 optimization steps, resulting in 256 samples per epoch. Due to constraints on our computational resources, our experiments utilized 9 samples per epoch.

To assess whether the number of samples per epoch was significantly affecting training, especially in terms of image quality and alignment with the prompts, we conducted additional experimental runs. We discovered that too few samples per epoch resulted in the generation of undesirable shapes or details and a slower learning process. This was expected because the models lacked sufficient performance data, provided via comprehensive rewards, to establish a robust gradient for training, leading to erratic training and performance trajectories. For experiments with a greater number of samples per epoch, we focused on a model trained with 256 samples per epoch over 50 epochs. The training of the 256-sample variant was limited to 50 epochs due to resource and time constraints.

Figure 3.6 compares the count distribution for the 9 and 256 samples per epoch models. The results validated that the 256-sample model performed similarly to the 9-sample model in terms of statistical metrics and reward function effectiveness. Moreover, the 256 samples per epoch model resulted in better image fidelity, having no signs of haziness or glow. Samples images are located in the Appendix. A longer training duration with this configuration might yield results that may not only match but potentially exceed those of the 9 samples per epoch model in terms of both statistical metrics and visual alignment.

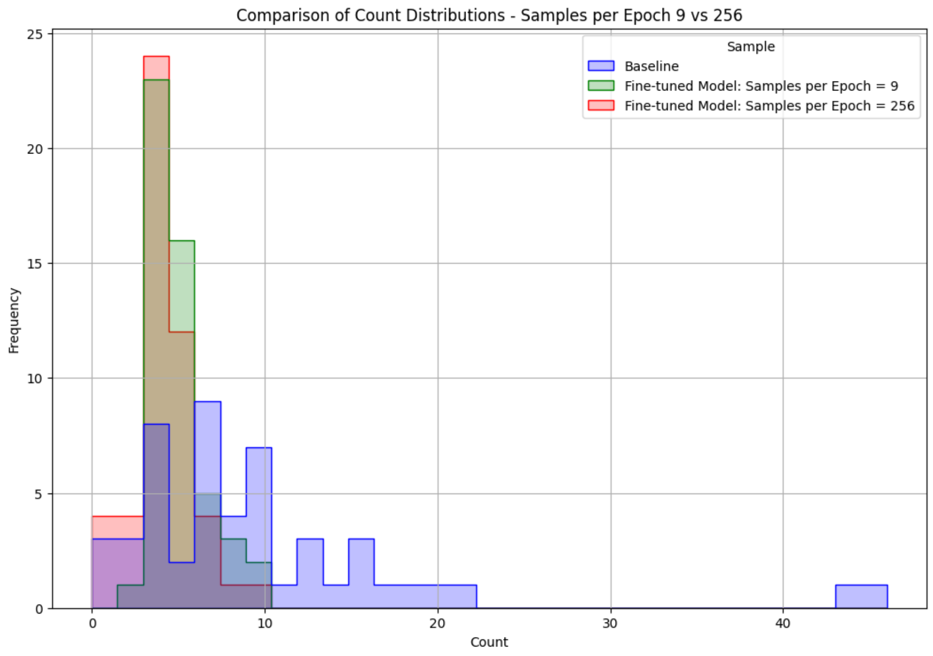


Figure 3.6: **Comparison of Count Distributions for Linear Training Using Varying Samples per Epoch** The plot above showcases that the 9 and 256 samples per epoch models trained using the Linear Reward Function performed similarly well. After being trained for 50 epochs, the large samples per epoch model attained a mean of 3.92, a median of 4.0, and a variance of 2.97. We hypothesize that with further epochs of training, the distribution will be comparable to or exceed the 9 samples per epoch model's distribution.

The Ratio Reward Function also performed well, generating the second-lowest variance and a mean close to 5. We originally trained the ratio model using the dynamic count curriculum, however, those experiments led to unsatisfactory results as the model would forget what it had learned across epochs. We postulate that the Ratio Reward Function did not perform well with the dynamic count curriculum because the ratio reward incentivizes generating more accurate counts for smaller Ns. Consequently, we believe that the Ratio Reward Function teaches the Stable Diffusion model to take less of the prompt into account for larger N as it understands the greatest priority is to minimize the penalty on the smaller count prompts since the difference of being off for a smaller count is more substantial. We then trained a model using the ratio function, but with prompts with constant N=5, like the linear models. This led to the ratio function results presented in Table 3.1. However, we discovered that the balls generated in sample images shared characteristics with golf balls. For example, many balls were covered in dimples and were small in size in relation to the entire image, much like golf balls. Refer to the Appendix for a sample image. This slight misalignment with the expectations of the prompt, which only instructed for balls of ambiguous type, in addition to the fact that its sample distribution is higher in variance and lower in accuracy, leads to its evaluation as the second-best reward function for the counting task.

We noticed that the training and sample images for the Exponential Reward Function performed poorly for all α values we tested, thus the results from those experiments are not noteworthy. Some intuition on why this function might have performed poorly is that the distribution before training had a high variance. In this case, when an image was generated with a significantly different number of balls than specified in the prompt, the increase in reward for slight improvements was marginal and did not provide as strong a signal as a more linear reward function would have.

For the Hybrid, Accuracy, Relative, and Adaptive Threshold Reward Functions, we noticed some interesting trends. The Relative and Adaptive Threshold Reward Functions had promising results, paralleling the performance of the linear and ratio functions in both training efficacy and statistical evaluation. This followed intuitively as the Relative approach was an adaptation of the Ratio Reward function, and the Adaptive Threshold Reward Function penalized on a linear schedule, much like the Linear Reward function. However, after 125 epochs, these models exhibited dramatic declines in performance; they essentially “forgot” how to generate balls correctly. Upon further training, only the Relative reward function managed to recover post-150 epochs, as evidenced by a rebound in the reward means in Figure 3.2. This suggests some resilience or ability to recalibrate the learning trajectory under DDPO.

On the other hand, the Hybrid and Accuracy reward functions performed very poorly under statistical evaluation. Both functions exhibited increasing reward means, plateauing near 80. However, when sampled, they generated images containing upwards of 120 balls, which significantly affected the variance and standard deviation metrics, as shown in Table 3.1. Moreover, they had surprisingly similar results in terms of generated sample images. Repeated sampling from these models produced nearly identical images, despite being trained under different reward functions. This phenomenon suggests that there might be some convergent learning behaviors or that similar penalization is inadvertently taking place across these models. This behavior was unexpected and starkly different from the other reward functions, which produced images with somewhat distinctive stylistic variations such as having different color tones, ball sizes, or image depths.

These observations raise questions about the underlying mechanisms of model training and hint at the possibility that certain reward configurations might lead to a homogenization of learning patterns. Further investigation into these phenomena could shed light on optimizing reward functions for not just accuracy and consistency for the counting task, but also for maintaining diversity across generated content for other objectives outside of just count.

The secondary focus of the reward functions was to see what function would work best for curriculum learning. We chose to employ the Linear Reward Function because it performed the best, both for the distribution comparison and the training reward mean. For our experiments, we utilized N=[1,3,5,7] for the sequential curriculum approach and loaded the weights from the previous module when conducting training for the current module. We hoped that training using this approach would lead to adequate results for prompts where $N > 5$, having chosen N=7 as our value of choice. Figures 3.9, 3.10, 3.11 demonstrate the positive effect curriculum learning had on training on counting larger Ns. Specifically, utilizing curriculum learning for N=[3,7] resulted in stark differences in the reward mean compared to training for such Ns without curriculum learning. As evident in Figures 3.9 and 3.11, the reward means start at a much higher point at epoch 0 and continually increase.

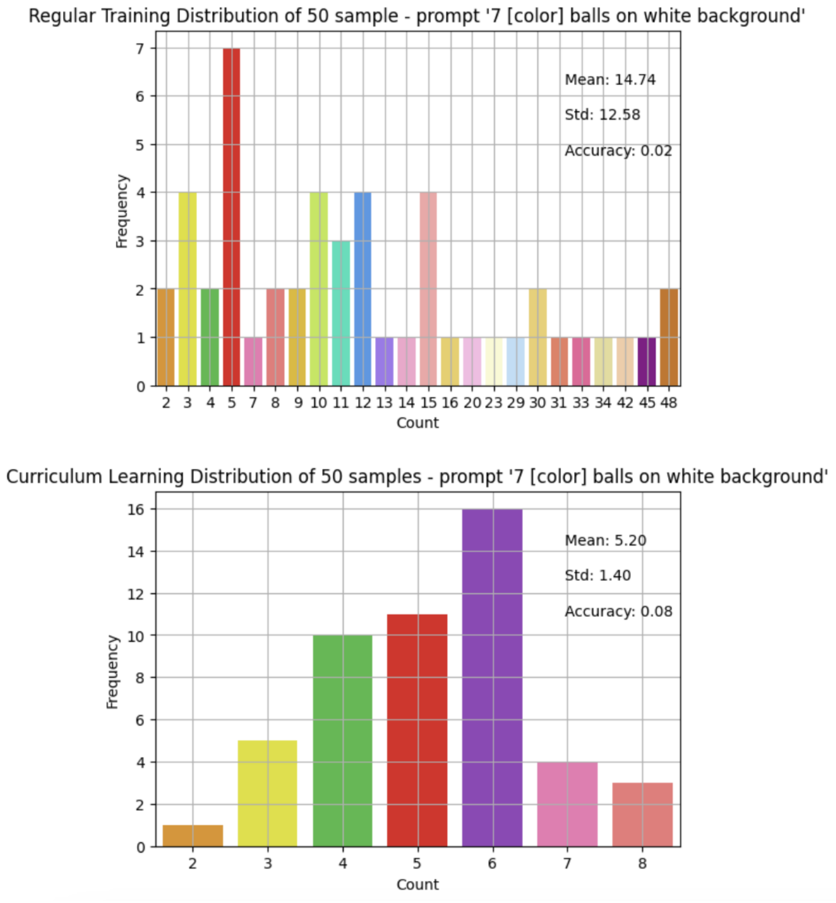


Figure 3.7: **Distribution of 7 Ball Counts for Fine-tuned Model** The top figure is the distribution of 50 samples after regular fine-tuning on the prompt "7 [color] balls on white background." The bottom figure is the distribution of 50 samples after curriculum fine-tuning on the same prompt. The distribution for the curriculum learning approach is better and has a lower variance than the regular approach.

Figure 3.7 showcases the distribution of counts for N=7 after regular training and after curriculum learning. Although not flawless, curriculum learning results in much better accuracy, mean, and standard deviation. The distribution with regular fine-tuning for the 7 balls looked similar to that of the baseline model, whereas the curriculum learning distribution seemed more adequate. In Figure 3.10, we observe that curriculum learning and regular fine-tuning achieve comparable results for 5 balls. However, at 7 balls, the distinction becomes much more clear: regular fine-tuning yields only marginal gains, whereas curriculum learning leads to significant improvements in the reward mean. We believe this can be attributed to the fact that training on previous modules in the curriculum formulates a generalizable policy that acquires knowledge and learns patterns for generating count. When transferred for larger Ns, this knowledge can be extrapolated to quickly learn how to generate the new prompted count. Our experiments shed light on the importance of curriculum learning for solving the counting problem.

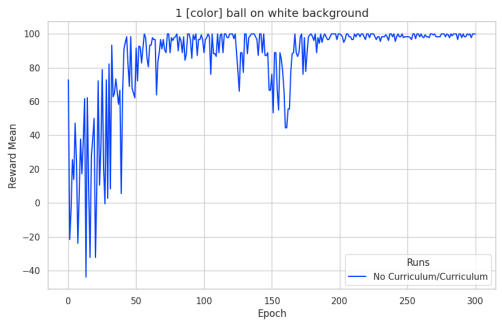


Figure 3.8: **Regular/Curriculum 1 Ball Reward Mean Plot** The plot demonstrates improvement in alignment to prompts for 1 ball prompts.

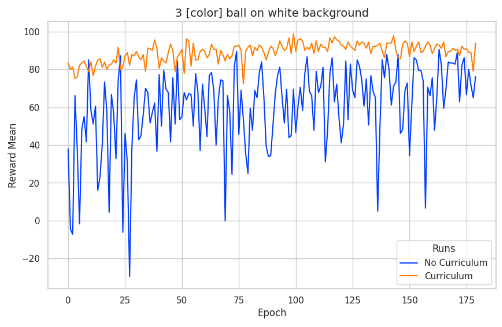


Figure 3.9: **3 Balls Curriculum vs. Regular Fine-tuning** Comparison of reward mean during curriculum learning vs regular fine-tuning for 3 balls.

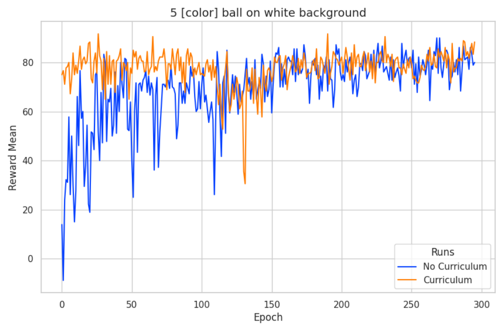


Figure 3.10: **5 Balls Curriculum vs. Regular Fine-tuning** Comparison of reward mean during curriculum learning vs regular fine-tuning for 5 balls.

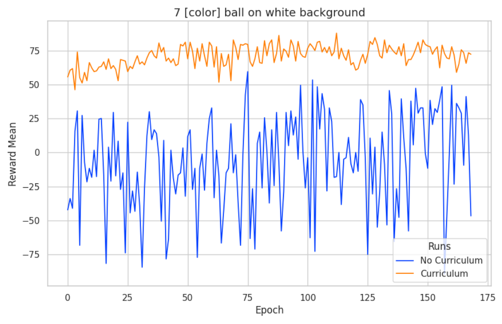
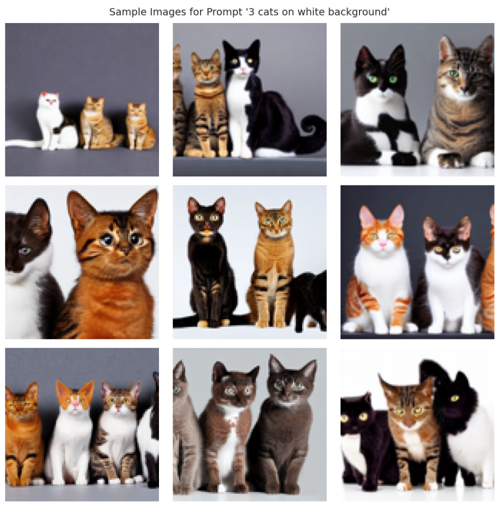


Figure 3.11: **7 Balls Curriculum vs. Regular Fine-tuning** Comparison of reward mean during curriculum learning vs regular fine-tuning for 7 balls.

The main hypothesis of the report was to establish if it was possible to fix the counting problem on balls using reinforcement learning, and if this would lead to sufficient generalization in providing more accurate counts in generating other kinds of images. Using transfer learning by loading the ball-count model weights from the curriculum learning approaches for N=[3,5], we wanted to see if the counting knowledge that was learned by the policy would generalize to the case of generating "[3,5] cats". Figure 3.17 shows an improvement in generating the proper number of cats when 50 sample images were generated from the baseline Stable Diffusion model versus the curriculum-trained model. Fortunately, this increase in count accuracy did not appear to lead to significant degradation of the image quality.



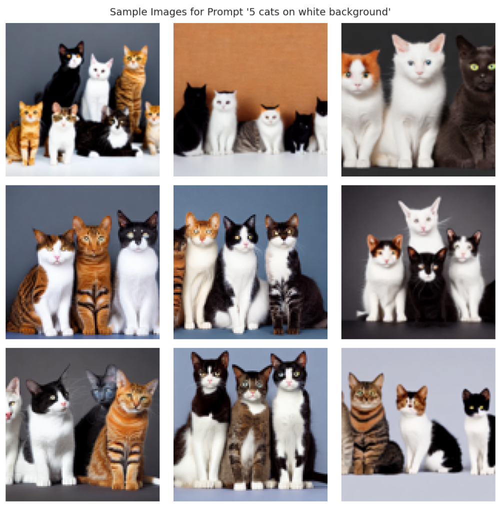
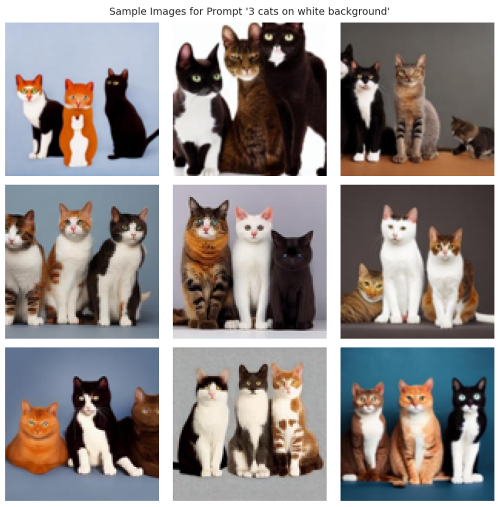


Figure 3.12: **3 Cat Samples Generated Using Baseline Weights** Note the frequency of 2 cats instead of the requested 3.

Figure 3.13: **5 Cat Samples Generated Using Baseline Weights** Note the frequency of 3 cats instead of the requested 5.



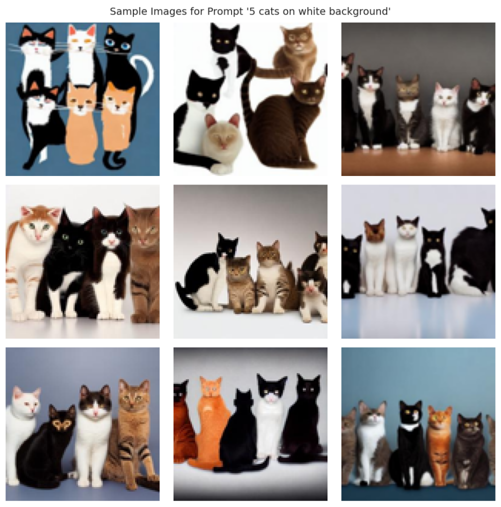


Figure 3.14: **3 Cat Samples Generated Using Curriculum 3 Weights** We see considerable improvements in the count in comparison to the baseline model.

Figure 3.15: **5 Cat Samples Generated Using Curriculum 5 Weights** We see considerable improvements in the count in comparison to the baseline model.

Figure 3.16: **Cat Sample Images** Note the quality of the image of cats is similar to the original.

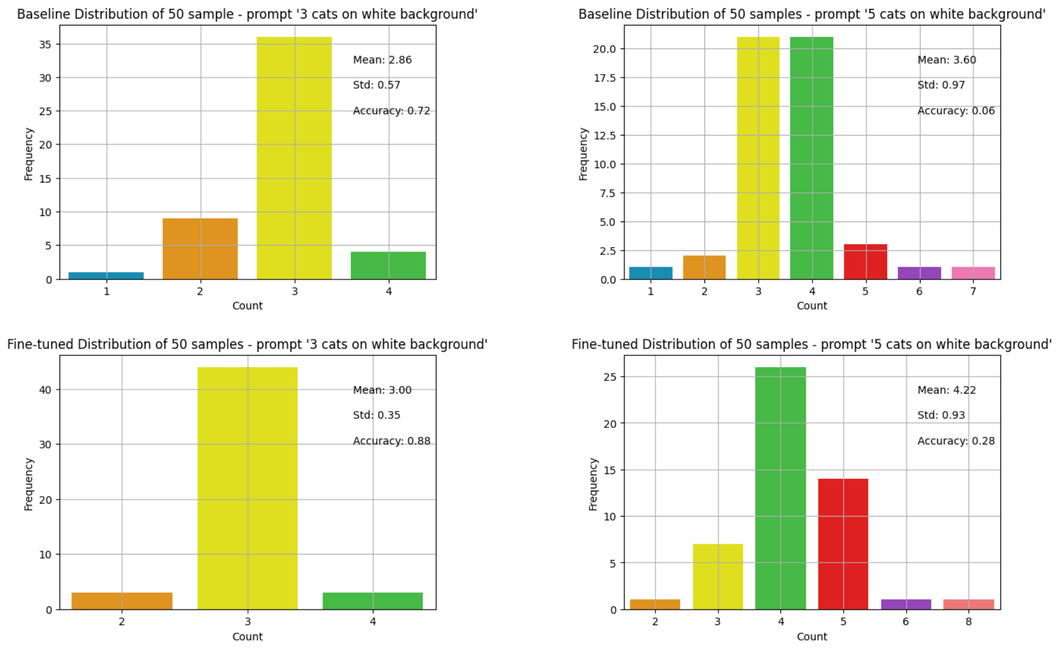


Figure 3.17: **Distribution of Cat Counts Given Input** Distribution of the number of cats generated by the model, comparing the baseline and curriculum fine-tuned models for the prompt "[3,5] cats."

In Figure 3.17, the yellow bars on the left plots denote the number of correct counts when prompted to generate 3 cats using both the baseline and curriculum fine-tuned weights. The increase in mean and accuracy demonstrates improved prompt alignment. On the right, the prompt was to generate 5 cats, and the correct amount of cats is denoted by the red bars. While this was not able to be optimized completely, compared to the baseline Stable Diffusion model, the fine-tuned model distribution has an accuracy of 400% more than the baseline and a mean closer to 5. This represents a significant improvement compared to the baseline, especially in the context of knowing that the model was solely trained on correcting the counts of balls. This demonstrates that the learning for counting balls has effectively generalized to include counting cats as well. This example of transfer learning shows that the counting strategy developed for simple ball scenarios is adaptable and can be applied to counting objects across different categories. These results hint at the notion that the model has an intrinsic representation of count, which enabled the transfer learning.

3.2 Discussion

The results from our reinforcement learning-based approach to address the counting problem in image generation have been encouraging, especially with the Linear Reward Function. The curriculum learning strategy demonstrated its efficacy by improving performance on more challenging prompts following $N > 5$, suggesting a practical way of training models on progressively complex tasks such as larger counts. Our findings confirm the potential of reinforcement learning not only in enhancing the accuracy of counts but also in maintaining the quality and prompt alignment of generated images. Additionally, the transfer learning results, specifically the generalization from counting balls to counting cats, provide a compelling proof of concept that the techniques developed for one domain can be extrapolated to another. This transfer-learning is a critical finding as it ensures that fine-tuned Stable Diffusion models can function across various tasks in creative or commercial settings.

3.3 Limitations

However, several limitations must be acknowledged:

- **Model Dependence on Samples per Epoch**: Our experiments showed a dependency on the number of samples per epoch, with larger sample sizes providing better image fidelity. This dependency can pose a challenge in computationally-constrained environments and might limit the scalability of our approach. Moreover, it is not inherently clear whether a greater number of samples per epoch results in equivalent refinement compared to the fewer samples we utilized.
- **Generalization Capability**: While we observed promising results in generalizing from balls to cats, this may not apply to all categories or more complex objects. We understood that the Stable Diffusion model was trained on images of colorful balls, but we do not know if the counting principle extrapolates for artificially generated objects, such as objects created by the user in the prompt. The generalization capabilities of the model are still an area that requires further exploration and validation.
- **Complexity of Prompts**: Our experiments utilized short prompts focusing on balls on a background. We do not know how more complex prompts may interfere with count performance, warranting further investigation.

3.4 Future Work

Future studies should focus on addressing these limitations. An intriguing area for future research would be to experiment with different reward functions within our curriculum learning framework. We only utilized the Linear Reward Function for our curriculum learning approach, so assessing the impact of different reward functions could provide valuable insight into training effectiveness and adaptation to complex tasks.

In conclusion, our study highlights the potential of using reinforcement learning to enhance Stable Diffusion image generation, for tasks such as object counting, while also shedding light on the complexities and challenges that coincide with these tasks.

Bibliography

[1] Zoheb Abai. *zoheb/yolos-small-balloon*. `https://huggingface.co/zoheb/yolos-small-balloon`. Accessed: 2024-04-18.

[2] Kevin Black et al. *Training Diffusion Models with Reinforcement Learning*. 2023. arXiv: `2305.13301 [cs.LG]`.

[3] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: `2005.12872 [cs.CV]`.

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: `1810.04805 [cs.CL]`.

[5] Yuxin Fang et al. *You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection*. 2021. arXiv: `2106.00666`. URL: `https://arxiv.org/abs/2106.00666`.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: `2006.11239 [cs.LG]`.

[7] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: `2304.02643 [cs.CV]`.

[8] Rohit Kundu. *YOLO: Algorithm for Object Detection Explained*. `https://www.v7labs.com/blog/yolo-object-detection`. Accessed: 2024-04-18.

[9] Haotian Liu et al. *Visual Instruction Tuning*. 2023. arXiv: `2304.08485 [cs.LG]`.

[10] Sanmit Narvekar et al. *Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey*. 2020. arXiv: `2003.04960 [cs.LG]`.

[11] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: `2112.10752 [cs.CV]`.

[12] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: `1503.03585 [cs.LG]`.

[13] Xudong Wang et al. *Cut and Learn for Unsupervised Object Detection and Instance Segmentation*. 2023. arXiv: `2301.11320 [cs.CV]`.

[14] Fuzhen Zhuang et al. *A Comprehensive Survey on Transfer Learning*. 2020. arXiv: `1911.02685 [cs.LG]`.

Appendix A

Sample Images Across Reward Functions

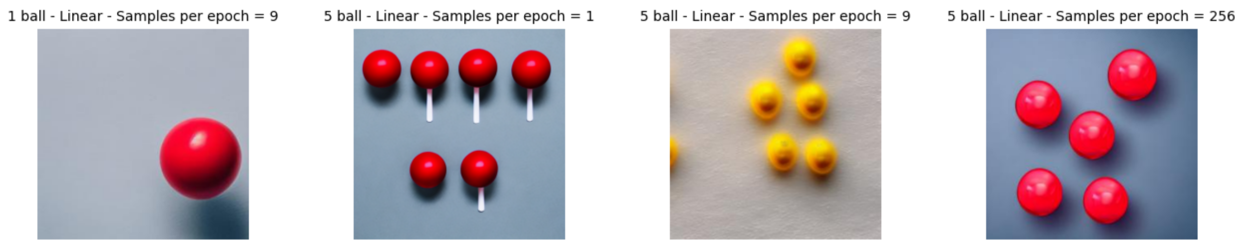

1 ball - Linear - Samples per epoch = 9
5 ball - Linear - Samples per epoch = 1
5 ball - Linear - Samples per epoch = 9
5 ball - Linear - Samples per epoch = 256

Figure A.1: **Sample Images from Linear Reward Function Experiments** We found variation among the style of generated images using this reward function. For example, the 1 sample per epoch model was prompted to generate an image of 5 red balls but generated 6 balls on popsicle sticks instead. Moreover, the 9 samples per epoch balls are more hazy than the 256 samples per epoch balls.

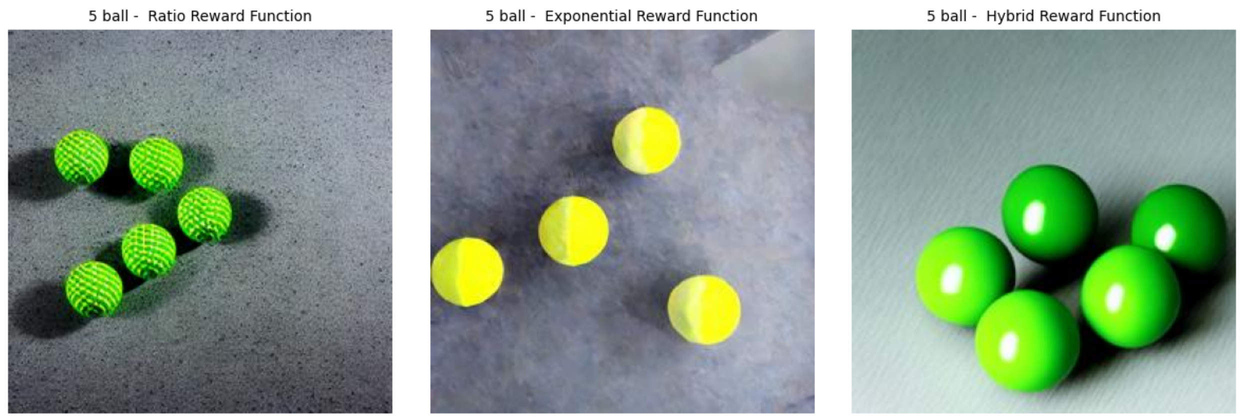

5 ball - Ratio Reward Function
5 ball - Exponential Reward Function
5 ball - Hybrid Reward Function

Figure A.2: **Sample Images from Ratio, Exponential, and Hybrid Reward Function Experiments** Notice the slight variations of style. For example, the ratio balls share characteristics with golf balls.

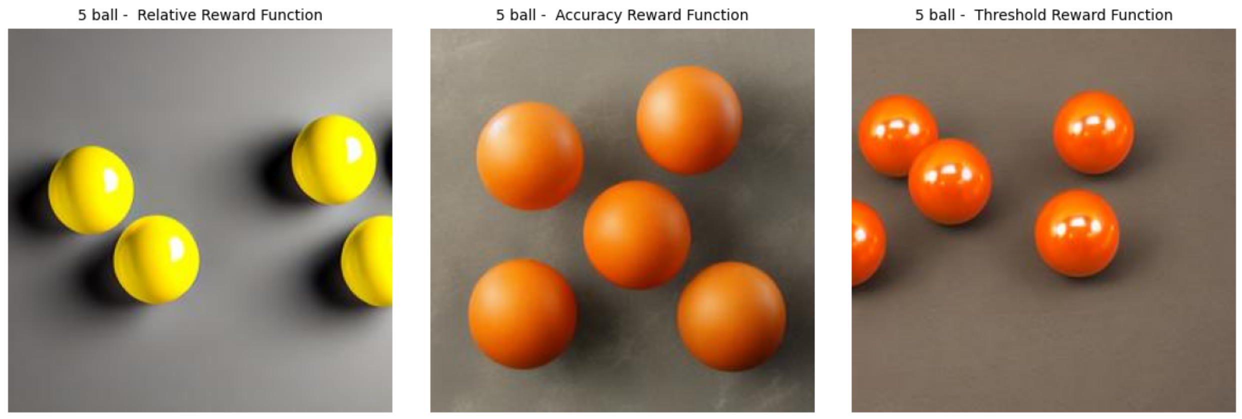

5 ball - Relative Reward Function
5 ball - Accuracy Reward Function
5 ball - Threshold Reward Function

Figure A.3: **Sample Images from Relative, Accuracy, and Adaptive Threshold Reward Function Experiments** Notice the slight variations of style. For example, the accuracy threshold balls are more matte than the relative balls.