COUNTING COUNTS: OVERCOMING COUNTING CHALLENGES IN IMAGE GENERATION USING RL

William Lin, Shaan Gill, Anisha Iyer

EXTENDED ABSTRACT

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. As such, the problem of image synthesis can be framed as a multi-step decision making problem. Past works such as denoising diffusion policy optimization (DDPO) have employed deep Reinforcement Learning (RL) techniques to directly optimize diffusion models for specific objectives. DDPO achieves success in optimizing text-to-image diffusion models for specialized objectives, but presents an unsolved problem in reliable semantic alignment for a subset of tasks [Black et al., 2023].

Despite the success of DDPO on RL-based diffusion model optimization, there exists a bottleneck in the recurring failure of these methods for text prompts that include counts. Text prompts of the structure "N objects" fail to reliably produce images consistent with the prompts. We term this the N-objects counting problem, characterized by the mismatch of the prompted count and the true value of generated objects in the final image. To close this gap in text-image diffusion model optimization, we include an object detection module that dynamically modifies the reward distribution based on the detection of generated objects.

This paper serves as progress towards solving the full class of N-objects problems by limiting this counting problem to a subset of the format "N [color] balls on white background". We additionally demonstrate that our optimization of text-to-image diffusion models generalizes to domains outside of the training context. We advance toward solving this problem by implementing multiple vision-informed reward functions and training policies using curriculum learning techniques.

Through preliminary experiments with multiple object detection networks, we selected YOLO to create three reward functions penalizing a base reward based on vision-informed criteria. Our approach presents an object detection module that infers the number of objects in the image, compares this to the prompted N value, and returns a modification to the reward. We employ linear, ratio, and exponential penalties in three separate reward functions. Our linear reward function linearly penalizes the difference between the prompted N value and the count of generated objects. Our preliminary experiments demonstrated that the error distribution for object counts varied for different N values when prompting different n values on the same model.To address this, we employ a ratio penalty which infers the appropriate penalty based on the prompted N value. Our third reward function employs an exponential penalty to more significantly penalize mismatches of lesser magnitude to address the high frequency of off-by-one mistakes as compared to off-by-multiple mistakes in our experiments.

Among the three reward functions, the linear reward function demonstrated the most significant increase in reward mean, while both the ratio and exponential reward functions were unstable during training. A training schedule using curriculum learning further improved the success of the linear reward function. Scheduled increases in prompted N values from 1 to 7 achieved higher reward means with curriculum learning than with the baseline training approach. We performed ablations on hyperparameters and isolated the number of training timesteps as beneficial for image quality but inconsequential for model performance for the counting task. We found an optima at 30 timesteps for training and sample generation. We developed a metric to evaluate performance on the counting problem by comparing the statistical features of the distributions of object counts before and after training with the prompted N value.

Our results demonstrated that our reward functions improved fidelity to the counts. Our training paradigm resolved many empirically observed issues in baseline models including the wide distribution of resulting object counts for prompting for five balls. Our approach produced a narrow distribution of counts of generated balls clustering closely around five without significant outliers. Our curriculum learning approach further improves these results with a marked difference for the complex 7-balls case. Moreover, our solution generalizes to N-objects problems beyond the subset of counting tasks we focused on. We plotted the distribution of counts from varying N-count prompts for animals. We observed narrower distributions of counts that were more closely centered around the prompted N-value, which constitutes a significant improvement as a result of Transfer Learning. Our results demonstrate significant promise for this technique as a solution to the broader N-objects problem and for prompt-image alignment for text-to-image diffusion models on the whole.

1 INTRODUCTION

The seminal work by Black et al. (2023) on applying Deep Reinforcement Learning (RL) to finetune Stable Diffusion processes constitutes a significant advancement. This work allows for finetuning on large Stable Diffusion models without having to curate new datasets. However, it revealed an intricate challenge: finetuning for prompt structures 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", DDPO exploits the reward function, which utilizes the LLaVA in conjunction with a BERT score metric, by rendering text related to the requested number in the image. As such, the reward mean increases due to the BERT model finding the generated images to be synonymous with the prompts, attributable to the text present in the images. In reality, the generated images do not align with the prompt.

This research paper builds upon the foundation laid by Black et al., aiming to address this specific issue of accurate object counting in Stable Diffusion image generation. Our original goal was to utilize deep RL to overcome the limitations of Denoising Diffusion Policy Optimization (DDPO), which fails on prompts of the form "*N* [objects]". While the focus of our thesis remains aligned with the initial problem, we have refined our scope to a more manageable subset. We limited the range from all problems with a prompt of the form "*N* [objects]" to specifically "*N* [color] balls on white background," with *N* ranging between 1 and 7. We directly address the counting problem by defining reward functions based on the number of objects in the image, inferred by an object detection module. We hypothesized that a targeted reward function that penalizes mismatched counts would correct the sequential denoising process of Stable Diffusion to generate the correct number of objects specified in the prompt beyond just balls.

We believe that solving this simpler count problem is a significant contribution. The results demonstrate that finetuning count alignment results in more accurate image generation. When sampling 50 images of an "*N* [objects]" prompt, the resultant images and their distribution are smaller in variance and have a mean closer to the expected count. In practice, our approach partially generalizes beyond simple shapes to more complex objects, in essence serving as a source of Transfer Learning.

2 PRELIMINARIES

2.1 PRELIMINARIES

A Markov Decision Process (MDP) serves as a formal representation of problems involving sequential decision-making. It is characterized by the tuple (S, A, ρ_0, P, R), where S represents the state space, A is the action space, ρ_0 is the distribution of initial states, P is the transition kernel, and R is the reward function. At each timestep t, the agent observes a state $s_t \in S$, takes an action $a_t \in A$, receives a reward $R(s_t, a_t)$, and transitions to a new state $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. The agent's behavior is governed by a policy $\pi(a|s)$.

In the course of interacting with the MDP, the agent generates trajectories, sequences of states and actions $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$. The primary objective of RL is for the agent to maximize $J_{RL}(\pi)$, the expected cumulative reward over trajectories sampled from its policy:

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

We explore conditional diffusion probabilistic models Sohl-Dickstein et al. (2015); Ho et al. (2020), denoted as $p(x_0|c)$, which reverse a Markovian forward process $q(x_t|x_{t-1})$ using a trained neural network $\mu_\theta(x_t, c, t)$. The objective $L_{DDPM}(\theta)$ maximizes a variational lower bound on the log-likelihood Ho et al. (2020). The denoising diffusion probabilistic model (DDPM) objective is given by:

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

The sampling process starts with $x_T \sim \mathcal{N}(0, I)$, and the reverse process $p_\theta(x_{t-1}|x_t, c)$ produces a trajectory $\{x_T, x_{T-1}, \ldots, x_0\}$ ending with x_0. Common samplers use an isotropic Gaussian reverse process with fixed timestep-dependent variance:

$$p_\theta(x_{t-1}|x_t, c) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t, c, t), \sigma_t^2 I)$$

We assume the presence of a pre-existing diffusion model, using Stable Diffusion v1.4 (Rombach et al., 2022) as the base model for experiments. Given a fixed sampler, the diffusion model establishes a sample distribution denoted as $p_\theta(x_0|c)$. The objective of denoising diffusion reinforcement learning (DDRL) is to maximize a reward signal, denoted as r, defined on the samples and contexts:

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

Here, $p(c)$ represents a context distribution of our choice. Our novel contribution to this objective come from our vision-informed reward functions, with linear, ratio, and exponential rewards functions.

2.2 DDPO

Diffusion probabilistic models represent a significant advancement for image synthesis. These models have been effectively utilized in a variety of applications, ranging from image and video synthesis to continuous control. The essential concept behind these models involves transforming a simple prior distribution into a more complex target distribution through a sequential denoising process, conceptualized as a maximum likelihood estimation problem. Black et al. introduces an innovative approach for finetuning such models, denoising diffusion policy optimization (DDPO). By reconceptualizing the denoising process as a multi-step decision-making task, Black et al. define a class of policy gradient algorithms that enable them to train diffusion as an RL problem. By recasting the denoising procedure as a multi-step Markov Decision Process, DDPO directly optimizes the RL objective using policy gradient estimators.

Within DDPO, Black et al. implement 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. This importance sampling approach allows for more accurate model updates. DDPO is tailored to train diffusion models to meet specific downstream objectives defined by the reward function. Black et al. apply this methodology to finetuning text-to-image diffusion models, focusing on tasks such as image compressibility and aesthetic quality.

2.3 Curriculum 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. (2020). 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 tasks. 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. (2020). 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, where a node corresponds to a distinct task. The formulation of a curriculum involves defining modules and training using some order of those modules. The simplest form is Sequence 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.

2.4 Object Detection

Object detection in machine learning enables a variety of applications. Standard object detection methods rely on deep learning, and given an image, provide the user with a set of bounding boxes, labels, and confidences. The task of object detection is a supervised learning problem and has a wide history of different methods, including breakthroughs in unsupervised methods. The object detector we decided on using in our reward function was based on YOLO: You Only Look Once. We also tested two other object detectors: CutLER and DETR. The metrics we used for selecting our object detector were speed and accuracy. Speed was required as its use in the reward function requires that it be able to generate a proper reward very quickly in the training of the model. YOLO ended up performing much quicker than the other two methods. Accuracy was measured by comparing the distribution generated under images with a human-labeled distribution, and we found again that YOLO performed better. Furthermore, YOLO was able to be finetuned for balloons, which led to it being able to detect spherical objects with higher accuracy. Because of this, it ended up being the object detector of choice for our problem.

Figures 2 and 3 show that the distribution of counts using YOLO and hand counts were similar despite some differences. Despite these minute differences in outputs, these results determined that YOLO's performance was sufficient.

Object detection in our problem served the purpose of identifying the number of balls in the images rendered by the diffusion model we trained. The number of balls and their locations are then used in the reward function to ensure that the number of balls generated is close to the number of balls demanded by the prompt. The information on the bounding box and the

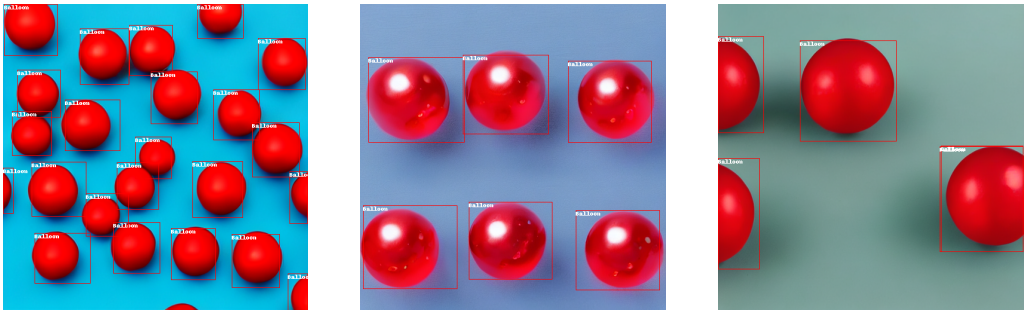

Figure 1: **Object Detection Results**. Results of object detection are evidenced by the rectangular outlines to identify detected balls of various sizes and fragments.

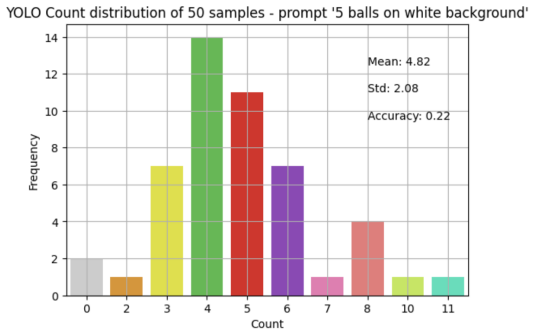

Figure 2: **YOLO Identification of Ball Counts** Distribution of ball counts identified by the YOLO object detection model.

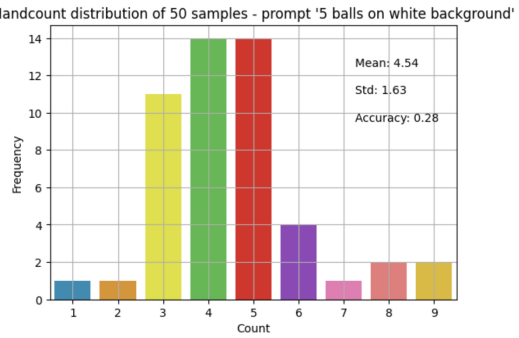

Figure 3: **Hand Counted Ball Identification** Distribution of ball counts identified through manual counting for comparison.

confidence informs us of how reliable the ball is as well, for instance, a ball with a shape deviating significantly from a sphere would not be considered a ball by our standards. The precise code for the reward function will be accessible in the appendix. However, 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 ”monkeys” and ”cats” that we have seen previously is unfeasible to approach with an object detection-based reward function. Thus we decided to focus on a subset of the problem, balls, aiming to find generalization among results.

3 Methods

The original DDPO paper by Black et al. provided a framework for optimizing a Stable Diffusion model with RL. The main component of our study's RL approach was the design and implementation of a specialized reward function. This function was formulated to align with our primary downstream objective: accurately producing the correct count of objects.

The reward function consists of two parts: a postprocess portion, which takes the output of the object detection network and gives the number of identified balls and ensures they have the proper shapes, and a reward function portion that acts as a function of the expected count. The postprocess function's main purpose is to take the list of bounding boxes that meet the confidence threshold criteria, and remove significantly overlapping bounding boxes, which happens when the object detector erroneously identifies one ball twice, and then sorts 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 half cut off by the boundary of the image. We screen these by considering the aspect ratio of the bounding box provided by the YOLO network, and if this deviates significantly from 1 we consider the ball a bad ball. The postprocess portion then returns a list of good balls and a list of bad balls, which are given by their bounding boxes. Then the total reward function relies on a function of the three inputs: the prompt, list of good balls and list of bad balls. Our reward function framework was pivotal for testing our hypothesis, as it allowed us to adjust the model in varying ways.

We tested three 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 boxes). The code for the reward function is under Appendix 1. We performed multiple experiments with each reward function. The most basic was where each model was given a prompt of the form ”N [color] balls on white background”, where N was fixed for each model between either 1, 3, 5, or 7, and colors from the

set [red, green, blue, orange, yellow]. We also implemented a curriculum learning approach, where we trained the model on perfecting generating images with smaller N before moving on to larger values. Empirically we found that the models had difficulties generating the proper count for larger numbers, so it seems that the natural progression in difficulty would be creating a larger number of balls. The results of these experiments will be discussed below and in the results section.

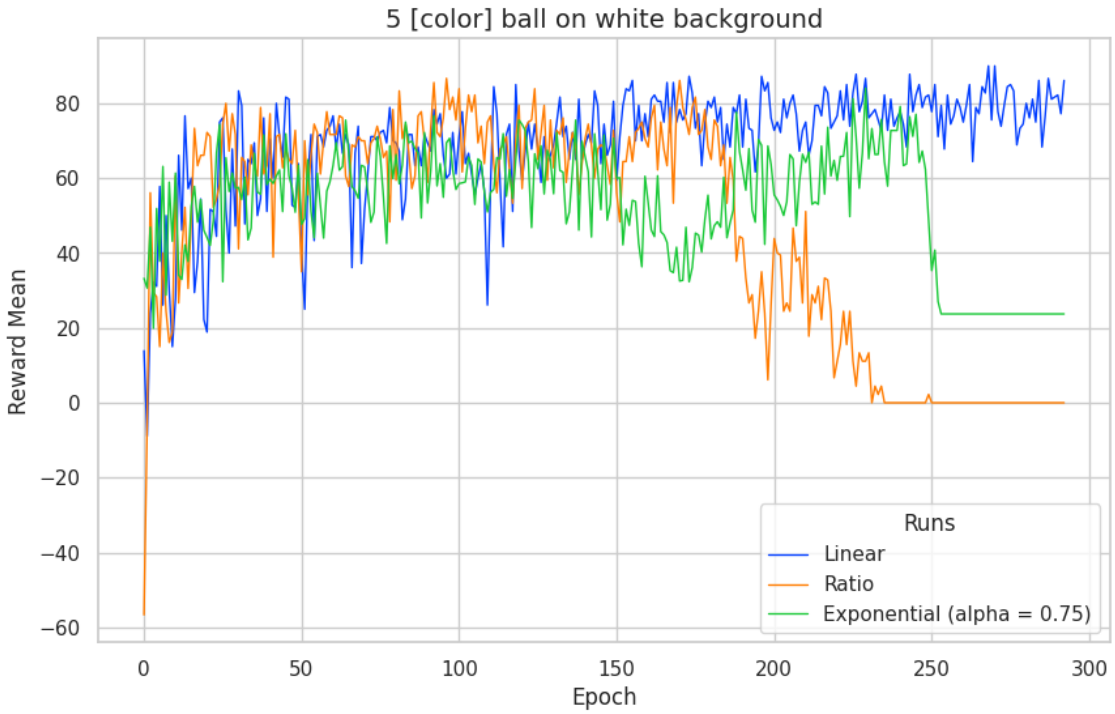

Figure 4: **Reward Functions Comparison** Comparison between Linear, Ratio, and Exponential Reward Functions for curriculum learning.

3.1 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 $- 15 \times |\text{totalCount} - \text{expectedCount}|$. Afterward, this 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". Again, this is the most basic reward function and ended up performing quite well compared to the others.

3.2 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%. Thus when we no longer see a fixed number of balls and instead see a mixture of different numbers of balls, we have to take into account the expected number as well. Similarly, we follow with the additional penalty of badly created balls: reward $\leftarrow$ reward $- 5 \times |\text{badBoxes}|$. This remains a fixed 5 reward penalty.

3.3 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 lot. The penalization strategy is as follows: reward $\leftarrow$ reward $\times \alpha^{|\text{totalCount} - \text{expectedCount}|}$. This penalization dynamically adjusts

the reward based on the difference between the total count of detected objects, *totalCount*, and the expected count specified in the prompt, *expectedCount*, in an exponential manner. Here, α is a constant less than 1, and its exponentiation by the absolute difference between *expectedCount* and *totalCount* 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 penalizing for small miscounts should be more significant.

3.4 Curriculum Learning

In our research, we observed a notable trend: as the number *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 *N*s, and the 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 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", then progressed to being trained on different colored 1 ball prompts. Lastly, it was trained on N=5 different colored balls. This approach 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 9, 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 low count of multi-colored objects (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.

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. Moreover, the weights and policy were loaded from the end of the past training modules when moving to the next. Our curriculum learning strategies emphasize the importance of thoroughly testing the sequence and complexity of tasks in training.

4 Baseline and Ablations

Our research aims to show that RL finetuning 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 ablation of hyperparameters, specifically focusing the effect of training timesteps on training, reward, and image quality. Figure 22 illustrates the reward mean 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, but we attribute 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 since more timesteps would allow the model to generate more prompt-aligned images. In that case, the model with T=10 would never train or increase in reward mean because it would never have enough timesteps to generate 5 ball instances. In essence, this would mean that the count problem could be solved with more timesteps alone. However, it looks like that is not the case, and that counts emerge very early in timestep enumeration, which is why all training curves are increasing. When looking at samples for the various Ts we notice that counts merge for all ablations, meaning that many timesteps are not essential to solve the counting problem since the counting inconsistency is a result of the Stable Diffusion model's misunderstanding of count.

These hyperparameter ablation experiments for training demonstrates that the increase in reward mean is not a result of timesteps and refinement from additional timesteps, but rather the impact of the finetuning on the Stable Diffusion model. However, 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 are focused on both practical image generation and count, we found that 30 timesteps are a good balance between reward function performance, image quality, and computational cost.

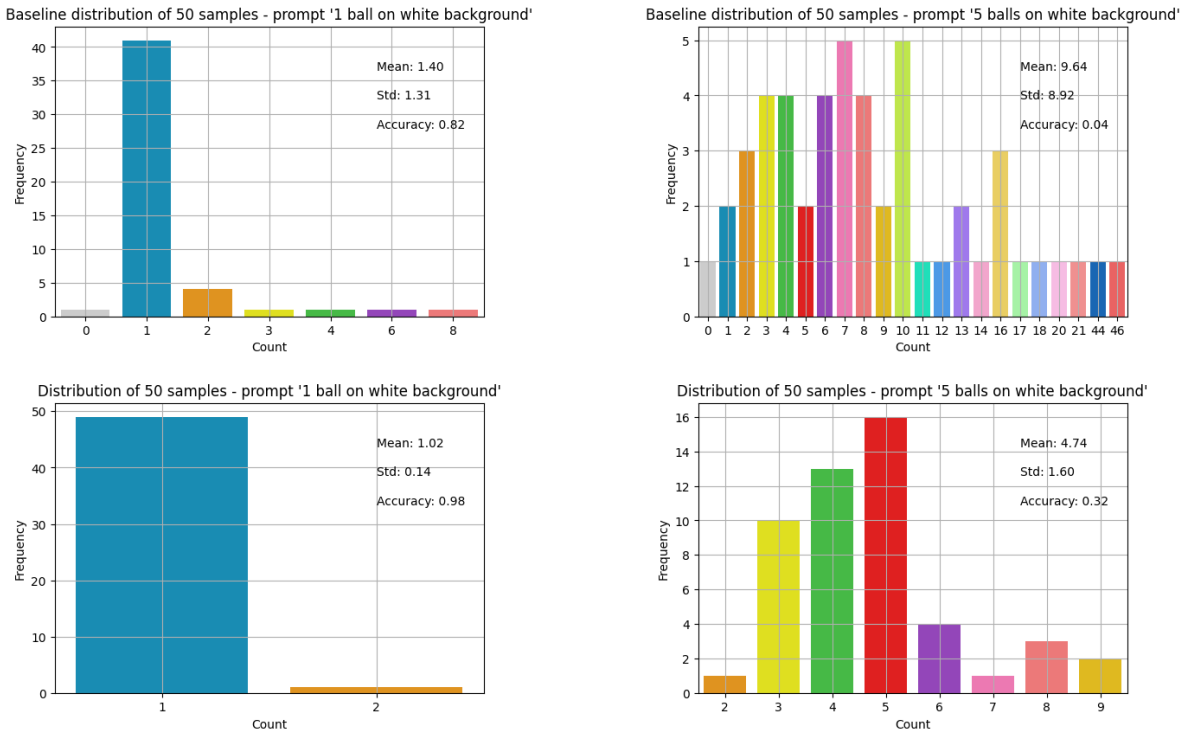

Figure 5: **Distribution of Ball Counts in Finetuned Model** Top: Distribution of ball counts in 50 samples from the untrained Stable Diffusion model. Bottom: Distribution of ball counts in 50 samples from the finetuned Stable Diffusion model using the prompt "5 [color] balls on white background." Results from the trained distribution have a lower variance and less distance from the expected mean.

As our final hyperparameters, we apply a constant learning rate of 3e-5, 30 timesteps, batch size of 9, and 300 epochs across training episodes. Our Stable Diffusion model of choice for all experiments was the CompVis Stable Diffusion v1.4 model. We utilize reward mean as an evaluation metric for training. To assess the practical effectiveness of the counting reward function, we generated 50 sample images before training and 50 samples following training using the prompt "5 [color] balls on white background" and compared the count distribution. We chose this prompt because we empirically saw that the distributions for smaller Ns were relatively adequate, but N = 5 resulted in a distribution with high variance. This is evident in the top two plots of Figure 5, 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. Accordingly, we wanted to see if we could improve the post-training distribution. As shown in Figure 5, we plot and compare the distribution of the counts from the baseline and finetuned model. Additionally, we consider metrics like mean, standard deviation, and accuracy to determine model performance.

5 Results

As shown in the above section, the paper has achieved significant results in shrinking the distribution of generated counts for balls. Multiple different variations of the three reward functions were tested to see which performed the best for both our training and practical effectiveness metrics. The linear reward function proved to optimize both evaluation criteria the best as compared to the baseline, massively shrinking down the baseline distribution and changing the count mean from 9.64 to 4.74 when prompted to generate 5 balls. We can also note significant increases in accuracy from 0.82 to 0.98 for the 1-ball case and 0.04 to 0.32 for the 5-ball case. Refer to Figure 5. 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 may perform poorly is that the distribution before training has a high count variance. In this case, when an image is generated with a significantly different number of balls than in the prompt, the increase in reward by improving slightly is marginal and does not provide as strong a signal as a linear reward function would. The ratio reward function, which was primarily tested for the dynamic count curriculum approach, performed similarly poorly. We postulate that this is because the ratio reward incentivizes generating more accurate counts for smaller Ns, taking 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.

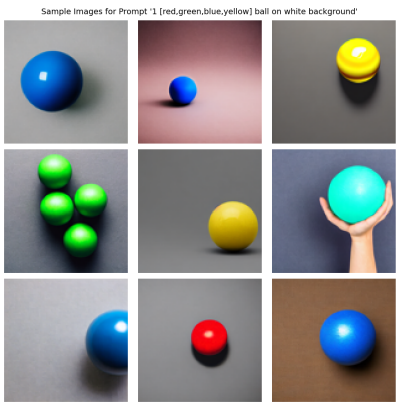

Figure 6: **Baseline 1 Ball Samples** Sample images generated with the baseline model for the prompt "1 [color] ball on white background."

Figure 7: **Baseline 5 Balls Samples** Sample images generated with the baseline model for the prompt "5 [color] balls on white background."

Figure 8: **Finetuned 1 Ball Samples** Sample images generated with the finetuned model for the prompt "1 [color] ball on white background."

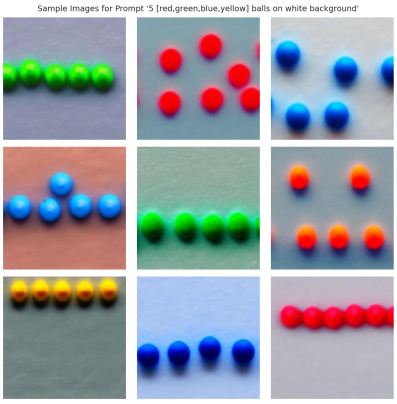

Figure 9: **Finetuned 5 Balls Samples** Sample images generated with the finetuned model for the prompt "5 [color] balls on white background."

Figure 10: **Improvement in Alignment to Prompt for Ball Counts** Visual representation of the significant improvement in alignment to the prompt for different counts of balls.

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 performs the best, both for the baseline 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 and policy from the previous modules when conducting training for the next module. Figures 12, 13, and 14 demonstrate the positive effect curriculum learning has 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 the figure, the reward means starts off at a much higher point at epoch 0 and continually increases. Figure 21 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. While we see that for 5 balls, curriculum learning and baseline achieve similar scores, there is a marked difference at 7 balls, as regular finetuning with reward functions achieves at most marginal gain but curriculum learning is actually able to make gains. Thus the distribution with regular finetuning for the 7 balls looks similar to as if it was not trained at all. 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 a 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.

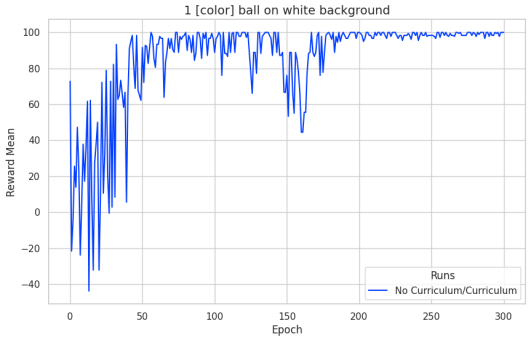

Figure 11: **Baseline/Curriculum 1 Ball Reward Mean Plot** Improvement in Alignment to Prompt for 1 Ball Prompts.

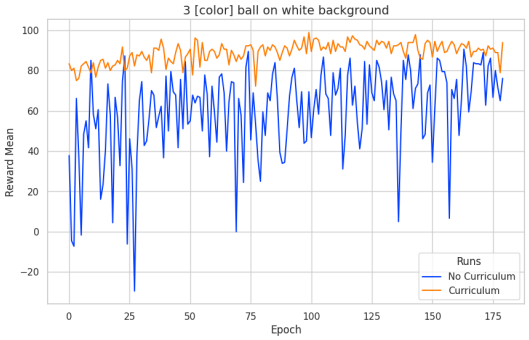

Figure 12: **3 Balls Curriculum vs. Baseline** Comparison of reward mean in curriculum learning for 3 balls versus the baseline model.

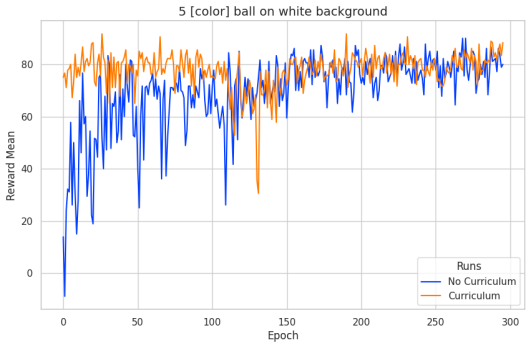

Figure 13: **5 Balls Curriculum vs. Baseline** Comparison of reward mean in curriculum learning for 5 balls versus the baseline model.

Figure 14: **7 Balls Curriculum vs. Baseline** Comparison of reward mean in curriculum learning for 7 balls versus the baseline model.

The main hypothesis of the paper was to establish if it was possible to fix the counting problem on balls using RL, 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 and policy, we wanted to see if the counting knowledge that was learned by the police would generalize to the case of generating "[3,5] cats". Figure 15 shows an improvement in generating the proper number of cats as counted when 50 sample images were generated from the default Stable Diffusion model versus the model further trained with RL. We note that this increase in the accuracy of count does not appear to lead to degradation of the original image quality, which is fortunate. Some sample images illustrating the different pictures are contained in the appendix.

In Figure 15, we have that on the left, the yellow bar denotes the number of 3s when prompted to generate 3 cats, and shows gains in accuracy. On the right, the prompt is to generate 5 cats, and the correct amount of cats is denoted by the red bar. While this wasn't able to optimize completely, compared to the baseline Stable Diffusion model, the finetuned model distribution has an accuracy of more than quadruple the baseline and a mean that is closer to 5. This represents a significant improvement compared to before the training, especially as the model was solely trained on correcting the counts on balls. This means correcting the counts on balls has generalized properly to correcting counts on cats. This transfer learning experiment demonstrates that the counting policy that was learned for the simple ball case can be transferred to generate counts of objects from other domains.

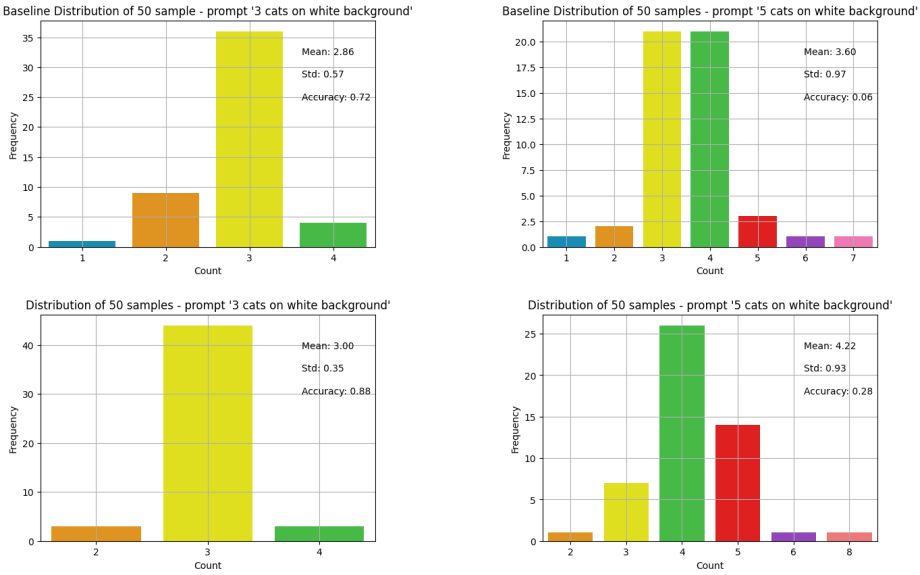

Figure 15: **Distribution of Cat Counts Given Input** Distribution of the number of cats generated by the model, comparing the baseline and finetuned models for the prompt "[3,5] cats."

6 Discussion and Limitations

One significant challenge we noticed is under certain reward functions, the ratio and exponential ones especially. Sometimes the reward would go from quite high to instantly dropping around to 0. The images that emerge from this resemble gaussian noise, and seem to be affected by the learning rate. A possible explanation is that the model gets stuck in a local minimum of reward and then cannot create any possible image that would have a positive reward. We have that this is different from the over-optimization that occurred in the original DDPO paper. In some of the reward functions in the original DDPO paper, such as the aesthetic and compressibility functions, there wasn't any increase in reward for staying aligned with the original prompt. Thus, these led to natural optimizations away from the original prompt. However, our reward function is naturally tied to the prompt. Thus this seems to be a different form of collapse than occurred previously.

Some paths for further research might include an additional curriculum learning where instead of varying the complexity of the task, e.g. increasing the number of balls, it varies the reward function. For instance, we could initially work to finetune the model using the linear variant of the reward function, which would easily penalize wide distributions, and then follow this by using the exponential reward function which is better for finetuning small changes in the distribution. Perhaps this would be a method that could break the plateau of around 80 reward that the linear reward function maintains.

Author Contributions

William Lin: I contributed to the project by testing and working on the object detection methods with balls and finally selecting the YOLOs object detector. Using this I built the postprocess function to take the outputs of the object detection and process it to a better form and designed the linear and ratio reward functions from which Shaan tested. Along with Shaan I made many plots and figures ensuring our results worked, e.g. comparing the pretraining and post training distributions for balls and monkeys. For the project report, I contributed to the extended abstract, and introduction, and wrote the object detection, methods, linear reward function, ratio reward function, results, and discussion and limitations sections.

Shaan Gill: I contributed to this project by running and debugging the DDPO algorithm for our use case. I had access to my lab's GPU server so I was responsible for running experiments my teammates and I created. I also tested the CutLER objection detection network. I was heavily involved in creating all the plots and figures for this project, which involved generating the samples discussed in the paper. In regard to the particular research, I tested debugged any reward functions my teammates made, created and tested the exponential reward function, and conducted all the curriculum learning and ablations experiments. In regards to the report, I contributed to the extended abstract, Introduction, DDPO, Curriculum Learning, Baseline and Ablations, Methods: Exponential and Curriculum Learning, and Results sections.

Anisha Iyer: I spent a significant portion of this project working on getting llava-server running via gunicorn, Flask, and ultimately a Docker image. This was because semantic mismatch in the production of images from number prompts was

only evident in the prompt-image alignment objective from DDPO, which used LLaVA. I spent several weeks rewriting the code for the Flask app and ultimately switching to running LLaVA on an external server and proxying it to the remote host that I have access to. Ultimately, these experiments were not included in the final report. I contributed significantly to earlier discussions about the project trajectory based on past literature in related areas. In regards to the report, I worked on the full extended abstract and preliminaries as well as contributing to the related works, methods, and baseline and ablations, and discussion sections in addition to all figure annotations.

REFERENCES

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning, 2023.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

Xudong Wang, Rohit Girdhar, Stella X. Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation, 2023a.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning, 2023b.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2023.

Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models, 2021.

Fuzhen Zhuang, Zhiyuan Qi, Dongbo Xi Keyu Duan, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.

A Appendix

A.1 Pseudocode for the reward function

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

A.2 Example Cat pictures Before and After Finetuning

Figure 16: 3 Cat Samples Generated Using Baseline Weights

Figure 17: 5 Cat Samples Generated Using Baseline Weights

Figure 18: 3 Cat Samples Generated Using Curriculum 3 Weights

Figure 19: 5 Cat Samples Generated Using Curriculum 5 Weights

Figure 20: Higher accuracy for number of cats showing generalization. Note the quality of the image of cats is similar to the original.

A.3 CURRICULUM VS BASIC TRAINING FOR 7 BALLS

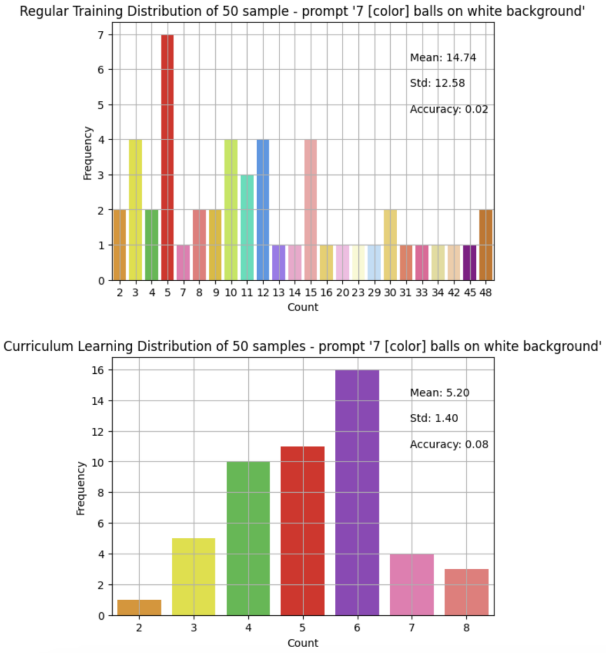

Figure 21: The top figure is the distribution of 50 samples after regular finetuning on the prompt "7 [color] balls on white background." The bottom figure is the distribution of 50 samples after curriculum finetuning on the same prompt. The distribution for the curriculum learning approach is better and has a lower variance than the regular approach.

A.4 ABLATIONS EXPERIMENT

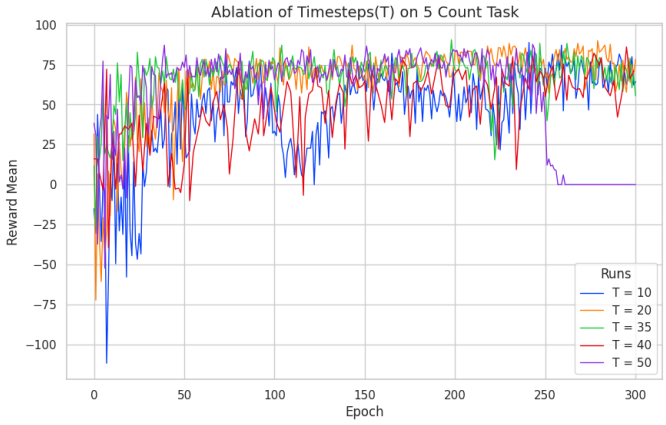

Figure 22: The plot above showcases the reward mean curves across epochs for our ablations experiment where we set diffusion timesteps T = [10,20,35,40,50].

A.5 GITHUB LINK: HTTPS://GITHUB.COM/SHAANSINGHGILL/DDPO-PYTORCH