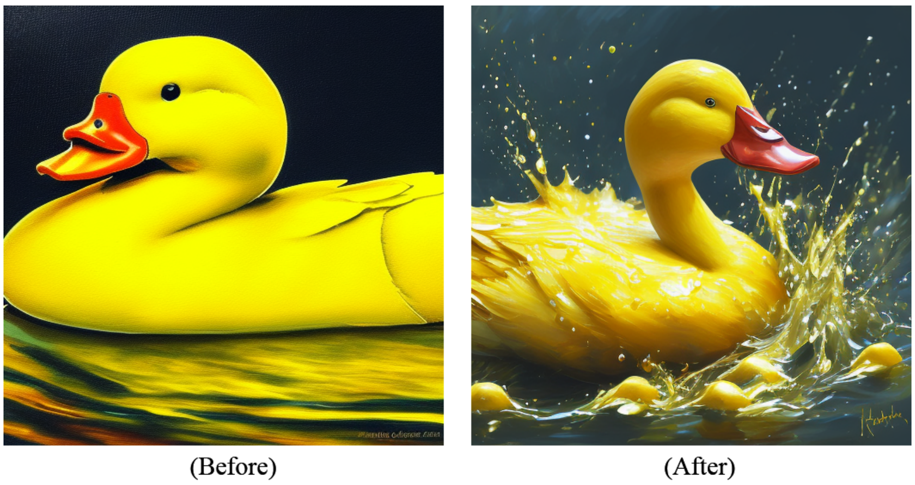
Magnus Kristiansen and Magnus Morud Vågen

Aligning Diffusion-Based Text-to-Image Models using Reinforcement Learning from Human Feedback

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2023

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science


(Before) (After)

"yellow duck, colored, splash, high detailed, painting"

Magnus Kristiansen and Magnus Morud Vågen

Aligning Diffusion-Based Text-to-Image Models using Reinforcement Learning from Human Feedback

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

This thesis explores the intersection of deep generative models and reinforcement learning, focusing on the alignment of diffusion-based text-to-image models using reinforcement learning from human feedback (RLHF). Text-to-image models have attracted significant attention due to their potential to generate high-fidelity and semantically coherent images from natural language descriptions. Despite their impressive capabilities, current models' performance is limited by their inability to fully capture human preferences, as they are trained using objectives that merely maximize the likelihood of training data, rather than the ease-of-use of human users' interactions with the models or the users' expected or preferred outputs.

Motivated by the recent success of aligning large language models using RLHF, we develop a corresponding methodology for text-to-image models based on diffusion models. We frame the reverse diffusion process as a Markov decision problem, allowing policy gradient methods to optimize the model using reward signals from human feedback. By training a reward model to serve as a proxy for human preferences during reinforcement learning, we enable the alignment of state-of-the-art text-to-image models with human values and preferences. Furthermore, we develop a novel reward model by repurposing parts of the diffusion model itself, ensuring a common understanding of the underlying data distribution between the model being optimized and the model guiding the optimization.

Our key contributions are fourfold. Firstly, we propose a systematic methodology for aligning diffusion models with human preferences using reinforcement learning from human feedback. Secondly, we validate the effectiveness of our approach through a series of experiments that showcase the improvement in image quality and text-image alignment. Thirdly, we provide insights into the challenges and limitations of our approach, informing future research in this domain. Lastly, we demonstrate the real-world applicability of our method, as it successfully aligns a state-of-the-art text-to-image model with general human preferences, underscoring its potential impact across various fields, such as art, education, and entertainment.

Through the development and validation of this novel RLHF methodology for text-to-image models, this thesis paves the way for more user-centric and safer models, enhancing their performance and unlocking their full potential in numerous real-world applications.

Sammendrag

Denne avhandlingen utforsker krysningen mellom dype generative modeller og forsterkningslæring, med søkelys på tilpasningen av diffusjonsbaserte tekst-til-bilde-modeller ved hjelp av forsterkningslæring fra menneskelige tilbakemeldinger (RLHF). Tekst-til-bilde-modeller har fått betydelig oppmerksomhet på grunn av deres potensiale til å generere høyoppløselige og semantisk sammenhengende bilder fra naturlige språkbeskrivelser. Til tross for deres imponerende egenskaper, er dagens modellers ytelse begrenset av deres manglende evne til fullt modellere menneskelige preferanser, da de blir trent ved å maksimere sannsynligheten for treningsdataene, heller enn brukervennligheten til menneskelige interaksjoner med modellene eller de forventede eller foretrukne resultatene fra brukerne.

Motivert av den nylige suksessen med å tilpasse store språkmodeller ved hjelp av RLHF, utvikler vi en tilsvarende metodikk for tekst-til-bilde-modeller basert på diffusjonsmodeller. Vi omformulerer den reverserte diffusjonsprosessen som et Markov-beslutningsproblem, noe som tillater at gradientbaserte metoder optimaliserer modellen ved hjelp av belønningssignaler fra menneskelige tilbakemeldinger. Ved å trene en belønningsmodell for å fungere som en proxy for menneskelige preferanser under forsterkningslæring, gjør vi det mulig å tilpasse toppmoderne tekst-til-bilde-modeller til menneskelige verdier og preferanser. Videre utvikler vi en ny belønningsmodell ved å gjenbruke deler av diffusjonsmodellen selv, for å sikre en felles forståelse av den underliggende datadistribusjonen mellom modellen som blir optimalisert og modellen som veileder optimaliseringen.

Våre hovedbidrag er firedelte. For det første foreslår vi en systematisk metodikk for å tilpasse diffusjonsmodeller til menneskelige preferanser ved hjelp av forsterkningslæring fra menneskelig tilbakemelding. For det andre validerer vi effektiviteten av vår tilnærming gjennom en serie eksperimenter som viser forbedringen i bildekvalitet og tekst-bilde-sammenheng. For det tredje gir vi innsikt i utfordringene og begrensningene ved vår tilnærming, noe som informerer fremtidig forskning på dette området. Til slutt demonstrerer vi den reelle anvendeligheten av vår metode, da den med hell tilpasser en toppmoderne tekst-til-bilde-modell til generelle menneskelige preferanser, og understreker dens mulige betydning i en rekke ulike felt, som kunst, pedagogikk og underholdning.

Gjennom utvikling og validering av denne nye RLHF-metoden for tekst-til-bilde-modeller, baner denne avhandlingen vei for mer brukersentrerte og sikrere modeller, forbedrer deres ytelse og frigjør deres fulle potensial for en rekke virkelige applikasjoner.

Preface

This Thesis serves as the primary work required to obtain a Master's degree in Computer Science at the Norwegian University of Science and Technology (NTNU). It explores the intersection between text-to-image models and human preference alignment, in the field of deep learning. The work was supervised by Professor Björn Gambäck.

Some sections of the Background chapter are adapted from the unpublished specialization project conducted by the same authors during fall 2022, as a preliminary study to this Thesis. Sections that are reproduced with no or minor modifications are marked with the †-symbol. Sections that are reproduced with moderate to significant modifications are marked with the ‡-symbol.

We would like to give special thanks to OpenAI for their seminal work on aligning Large Language Models (LLMs) using Reinforcement Learning from Human Feedback (RLHF), a foundation upon which this thesis is constructed, their significant contributions to the literature on diffusion and text-to-image models, and their generosity in open-sourcing both their diffusion model implementations and their initial RLHF implementation for LLMs, including providing pre-trained diffusion model checkpoints. Appreciation is also extended to the CompVis research group at LMU Munich, Runway, and Stability AI for the open-source text-to-image model Stable Diffusion, which was instrumental in evaluating our method. Our gratitude goes out to the PyTorch Foundation and the Hugging Face team for providing the tools that were essential in our method implementation. Special acknowledgment is due to our supervisor, Professor Björn Gambäck, for his encouragement to pursue our own research interests, for letting us work independently, and for providing valuable feedback on this thesis. Lastly, we are especially grateful to the High-Performance Computing (HPC) group at NTNU for granting us access to the IDUN cluster (Själander et al., 2019), including a full-node reservation for part of our project duration, which greatly facilitated our research.

Magnus Kristiansen and Magnus Morud Vågen

Trondheim, 26th June 2023

Contents

List of Figures

List of Tables

Acronyms

AE Autoencoder.

BPE Byte-Pair Encoding.

CLIP Contrastive Language-Image Pretraining.

CNN Convolutional Neural Network.

CoCa Contrastive Captioners.

DDIM Denoising Diffusion Implicit Models.

DDP Distributed Data Parallel.

DDPM Denoising Diffusion Probabilistic Model.

DDPO Denoising Diffusion Policy Optimization.

DPOK Diffusion Policy Optimization with KL regularization.

dVAE discrete Variational Autoencoder.

ELBO Evidence Lower Bound.

FID Fréchet Inception Distance.

GAE Generalized Advantage Estimation.

GAN Generative Adversarial Network.

GPU Graphics Processing Unit.

HPC High-Performance Computing.

HPS Human Preference Score.

IS Inception Score.

KL-divergence	Kullback-Leibler divergence.
LLM	Large Language Model.
LoRA	Low-Rank Adaptation.
MDP	Markov Decision Process.
MLE	Maximum Likelihood Estimation.
MLP	Multilayer Perceptron.
MS-COCO Captions	Microsoft Common Objects in Context Captions.
NLP	Natural Language Processing.
NTNU	Norwegian University of Science and Technology.
PCA	Principal Component Analysis.
PPO	Proximal Policy Optimization.
RLAIF	Reinforcement Learning from AI Feedback.
RLHF	Reinforcement Learning from Human Feedback.
RMSE	Root-Mean-Square Error.
SDE	Stochastic Differential Equation.
SGM	Score-based Generative Model.
SOTA	State-of-the-Art.
TPU	Tensor Processing Unit.
TRPO	Trust Region Policy Optimization.
VAE	Variational Autoencoder.
ViT	Vision Transformer.
ViT-VQGAN	Vision Transformer Vector Quantized Generative Adversarial Network.
VQ	Vector Quantized.
VQGAN	Vector Quantized Generative Adversarial Network.
VQVAE	Vector Quantized Variational Autoencoder.

1. Introduction

In this Thesis, we propose a novel approach for aligning diffusion-based text-to-image models using Reinforcement Learning from Human Feedback (RLHF). Diffusion models are a class of generative models that can produce high-quality and diverse images by reversing a stochastic diffusion process. RLHF is a technique that leverages human feedback as a reward signal to optimize a model's behavior using reinforcement learning. By combining these two techniques, we aim to improve the performance and alignment of text-to-image models.

1.1. Motivation

Text-to-image models aim to synthesize realistic and diverse images from natural language descriptions, enabling novel applications in art, content creation, education, entertainment, and more. They have attracted a lot of attention in recent years, thanks to advancements in transformers for Natural Language Processing (NLP), diffusion models for high-quality image synthesis, and large-scale datasets for multimodal training. Some of the most impressive text-to-image models include DALL-E 2 (Ramesh et al., 2022), which shocked the world with its ability to create surreal and humorous images from complex prompts; Midjourney[1], which offers a paid service for high-quality image generation with easy-to-use prompts; and Stable Diffusion (Rombach et al., 2022), which is a free and open-source model that can be run locally and has a huge community of users training custom models and constantly adding new features.

While diffusion-based text-to-image models have come a long way since their initial introduction, they still face some limitations and challenges. One of them is that the generated images are often misaligned with human preferences. For example, human users may prefer more creative, realistic, coherent, diverse, or ethical images than the diffusion model can produce by default. Another is that they can be difficult to use and control, requiring careful prompt engineering and iterative refinement to achieve desired results.

These limitations stem from the way the models are constructed and trained. The quality and diversity of the generated images depend largely on the training data used as well as the chosen text encoder (Saharia et al., 2022a). The training data used to train large text-to-image models is usually scraped from the internet, with varying quality of

[1]Available at: `https://www.midjourney.com/`.

(image, text) pairs. Furthermore, human preferences are subjective and hard to define and capture with predefined loss functions or metrics. The standard training objective for diffusion models is based on Maximum Likelihood Estimation (MLE), which tries to optimize the generative likelihood of the training data and does not seek to capture the nuances, subtleties, and complexity of human preferences. These inefficiencies in training can lead to unsatisfactory or even harmful generations. Therefore, it would be desirable to have a mechanism that can directly optimize the text-to-image model using human feedback.

Inspired by the recent success of RLHF in aligning language models (Ouyang et al., 2022; Schulman et al., 2022), we propose to apply RLHF to align diffusion models for text-to-image generation. We hypothesize that RLHF can improve the quality and diversity of the images generated by diffusion models, as well as make them more aligned with human preferences and expectations.

From a practical perspective, text-to-image generation using diffusion models has many potential applications that can benefit from RLHF. For example, RLHF can enhance the creativity and expressiveness of text-to-image systems by allowing users to provide feedback and guide the generation process. RLHF can also improve the safety and reliability of text-to-image systems by preventing or correcting undesirable or harmful outputs. Furthermore, RLHF can increase user satisfaction and trust in text-to-image systems by aligning them with user values and goals.

1.2. Thesis Goal and Research Questions

This Thesis aims to investigate the potential of RLHF in improving the alignment of diffusion-based text-to-image models. By drawing inspiration from recent advancements in natural language processing and related fields, we aspire to create a novel approach for aligning the text-to-image generation process. The primary goal of this Thesis is:

Thesis Goal: Develop an approach to align diffusion-based text-to-image models using RLHF for generating high-fidelity and semantically consistent images.

To accomplish this goal, the Thesis will address the following research questions:

Research questions:

- *RQ1: How can diffusion-based text-to-image models be mapped to a policy gradient environment that enables RLHF?*
- *RQ2: How can we design and train a reward model that effectively captures human preferences and provides rewards for diffusion-based text-to-image models?*

- *RQ3: How can we evaluate the performance and alignment of text-to-image models trained with RLHF?*

The first research question (RQ1) aims to explore the applicability of policy gradient methods, such as Proximal Policy Optimization (PPO), in the context of RLHF for text-to-image generation. We will investigate the challenges and potential solutions in adapting diffusion models to reinforcement learning environments.

The second research question (RQ2) focuses on the design and training of a reward model capable of learning from human feedback. This question will delve into various reward modeling approaches and their suitability for RLHF in the context of text-to-image generation, as well as how to effectively collect human preferences.

The third research question (RQ3) aims to establish evaluation metrics and methods for assessing the performance and alignment of diffusion models trained using RLHF. We will compare these models with suitable baselines, considering both their image fidelity and image-text alignment.

1.3. High-Level Methodology

This section provides a high-level overview of the methodology proposed in this Thesis; it should serve as a reference to the reader.

The method is an adaptation of OpenAI's RLHF approach for aligning Large Language Models (LLMs) (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) to the domain of diffusion-based text-to-image models. The method leverages a pre-trained diffusion model (e.g., Stable Diffusion; Rombach et al., 2022) and consists of four main steps, outlined below:

Step 1: Curating a Labeling Dataset. The first step in the methodology involves curating a dataset consisting of text prompts and associated image samples generated from the text-to-image model. Following a similar approach to the concurrent work of Lee et al. (2023), simple prompts are used to test specific capabilities of the text-to-image model, such as generating objects with certain properties (e.g., a lion wearing pink sunglasses). Multiple image samples are generated for each text prompt to ensure diverse representations for comparison.

Step 2: Collecting Human Feedback. The second step involves collecting human feedback on the curated labeling dataset. Human labelers are presented with pairs of image samples generated from the same text prompt. These labelers are tasked with selecting which image sample better aligns with the corresponding text prompt.

Step 3: Training a Reward Model. The collected human feedback is used to train a reward model via supervised learning. This model learns to predict preference rankings for pairs of responses, where each pair has a "winner" and a "loser" according to human preference. In order to effectively learn the relationships between texts and images in this context, the encoder component of the diffusion model is repurposed as the reward model architecture. This approach offers an additional benefit by pre-aligning the reward model with the underlying image generation process in terms of understanding text and image content.

Step 4: Optimizing Policy with Reinforcement Learning. Once the reward model is trained, the next step is to optimize a diffusion policy using reinforcement learning algorithms, specifically policy gradient methods (e.g., PPO). The objective is to optimize the policy against human preferences by employing the reward model as a proxy.

To further enhance performance and alignment, this methodology allows for continuous iteration of steps 2 and 3, which involves collecting additional comparison data on the current best policy and training new reward models and policies accordingly. This multi-iterated training process (Gao et al., 2022) addresses potential misalignments that may arise as the optimized policy deviates from the initial policy and, consequently, from the reward model trained on samples generated by the initial policy.

A comprehensive exploration of the methodology outlined above will be covered in two main chapters:

Chapter 4: Human Data Collection This chapter will discuss the process of curating a labeling dataset, *Step 1*, and collecting human feedback on this dataset, *Step 2*. It will elaborate on the selection of text prompts, image sample generation, and the procedure and interface for obtaining pairwise comparisons from human evaluators. Furthermore, criteria used by evaluators to determine which image samples better align with the given text prompts will be explored.

Chapter 5: Reinforcement Learning for Fine-tuning Diffusion Models The focus of this chapter will be on *Step 3* and *Step 4*, which involve training a reward model using human feedback and detailing the optimization of a policy against the reward model using reinforcement learning methods. The supervised training objective employed will be presented, and a discussion about the reward model architecture derived from the encoder part of the diffusion model will follow. Additionally, it will be explained how this approach allows for effective learning of text-image relationships. In the latter part of the chapter emphasis will be placed on the mapping of a diffusion model to a reinforcement learning environment for use as a policy to be optimized. Followed by our PPO setup.

1.4. Key Contributions

The key contributions of this Thesis are:

1. Development of a methodology for aligning diffusion models with human preferences: The Thesis presents a systematic methodology that addresses a specific gap in the field by proposing a novel approach to align diffusion models with human preferences using reinforcement learning. The methodology includes steps for data collection, reward model training, and policy optimization, allowing for the alignment of state-of-the-art text-to-image models.

2. Validation of the methodology through a series of experiments: The Thesis includes a series of experiments that progressively increase in complexity and demonstrate the effectiveness of the proposed methodology. These experiments involve simple diffusion models trained on restricted datasets and aligned using fixed reward functions, as well as more complex text-to-image models trained on large-scale datasets and aligned using reward models trained using human feedback. The results show that the methodology can effectively align diffusion models using a wide variety of reward schemes to improve the quality of generated images.

3. Insights into the challenges and limitations of aligning diffusion models with human preferences: The experiments highlight various challenges and limitations encountered during the alignment process. These include issues related to reward model training, mode collapse, degradation of image quality, and loss of diversity. These insights provide valuable knowledge for guiding future research in this area.

4. Proof-of-concept of aligning models with general human preferences: The experiments conducted in the Thesis demonstrate the scalability and real-world applicability of the methodology. By successfully aligning a state-of-the-art text-to-image model with general human preferences, the Thesis establishes the methodology as a powerful tool for generating images that are better aligned with human expectations and preferences. This highlights the potential impact of the methodology in various domains, including art, education, and entertainment, enabling the development of more sophisticated and user-centric text-to-image models.

1.5. Thesis Structure

The Thesis is structured as follows:

Chapter 2 introduces the background and preliminaries necessary to understand current text-to-image models, such as transformers and diffusion models. It also gives an introduction to reinforcement learning techniques which are used in RLHF. Other topics, such as key datasets for training text-to-image models and key metrics for evaluation, are also covered.

Chapter 3 provides an overview of the related work on aligning generative models using human preferences, primarily in the domain of language models. It also covers some of the early explorations into aligning text-to-image models using human preferences, focusing on reward modeling.

Chapters 4-5 present our proposed method for aligning diffusion-based text-to-image models using RLHF, the human data collection scheme, the reward model architecture, and the training procedures for our reward model and text-to-image model.

Chapter 6 describes the series of experiments conducted to develop and evaluate the methodology presented in the two prior chapters. The experiments gradually increase in complexity, building up to the alignment of a text-to-image model using our proposed RLHF framework. It covers 7 experiments in total, including their motivation, experimental setup, results, and a per-experiment evaluation.

Chapter 7 discusses the experiment series, reviews our research questions in light of our work, and discusses the implications and limitations of our approach.

Chapter 8 establishes future directions for work on the alignment of text-to-image models in light of our methodology and findings, and concludes the Thesis by summarizing our main contributions.

2. Background Theory

This chapter introduces key deep-learning architectures and concepts that are fundamental to understanding State-of-the-Art (SOTA) text-to-image models and alignment techniques. It assumes that the reader has a general understanding of machine learning and deep learning concepts, especially artificial neural networks. Some knowledge of probabilistic notation used in variational Bayesian methods is also valuable when reading the sections on generative models, Variational Autoencoders (VAEs), and diffusion models. ‡

We will begin by discussing generative models and encoder-decoder architectures, including VAEs, which form the basis of many other generative architectures. Next, we cover the transformer encoder, which serves as the textual understanding module for many text-to-image models. We will also cover U-Nets, which are commonly used with diffusion models, and the diffusion models themselves, which have proved themselves to be competent generative models in systems such as DALL-E 2 (Ramesh et al., 2022), Imagen (Saharia et al., 2022a) and Stable Diffusion (Rombach et al., 2022). Next, we will discuss Contrastive Language-Image Pretraining (CLIP), a joint image-text encoder that is used in Stable Diffusion and other text-to-image systems. Then, reinforcement learning and Proximal Policy Optimization (PPO), which serves as the alignment technique of the methodology presented in this Thesis, will be explained. Finally, we will provide an overview of the datasets used to train the diffusion models in the experiments in chapter 6. ‡

2.1. Generative Modeling

Generative modeling constitutes a fundamental approach within the domains of statistical modeling and deep learning, and it can be differentiated from its counterpart, discriminative modeling. While discriminative models primarily concentrate on learning the boundaries between data classes, or $p(y \mid x)$, generative models' main objective is to learn the underlying data distribution, represented as $p(x)$. Due to their ability to learn the data distribution itself, generative models have the capacity to generate novel samples that resemble the training data. Some noteworthy instances of generative models include Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Denoising Diffusion Probabilistic Models (DDPMs).

Formally, Considering a dataset $X = \{x_1, x_2, \ldots, x_n\}$ consisting of n samples; the goal of generative modeling is to approximate the true data distribution $p(x)$ with a learned distribution $p_\theta(x)$ that captures the joint distribution of the data. In the case of deep learning, θ represents the parameters of an artificial neural network.

Within the framework of text-to-image synthesis, generative models can be employed to learn the joint distribution of text and images, $p(y, x)$. Consequently, the resulting text-to-image model can be used for generating synthetic images x that are conditioned on specific text descriptions y.

2.1.1. The Principle of Maximum Likelihood Estimation

A central concept in parameter estimation for generative models is Maximum Likelihood Estimation (MLE). MLE seeks to find the parameters θ that maximize the likelihood of the observed data X given the model $p_\theta(x)$. In other words, MLE aims to find the model parameters that make the observed data as probable as possible.

Mathematically, MLE involves finding $\hat{\theta}$ such that:

$$\hat{\theta} = \arg\max_\theta \prod_{i=1}^{n} p_\theta(x_i),$$

In practice, it is often more convenient to work with the log-likelihood, which transforms the product into a sum, as follows:

$$\hat{\theta} = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(x_i).$$

The transformation to log-likelihood preserves the optimization objective since the logarithm is a monotonic function. Furthermore, it can help in avoiding numerical issues resulting from extremely small likelihood values.

2.1.2. Variational Inference

There are instances when we require approximate inference techniques due to the intractability of computing exact posteriors or MLEs directly. Variational Inference (VI) presents a viable solution, particularly in the context of Bayesian statistics, by approximating the true posterior distribution $p_\theta(z \mid x)$ with a tractable distribution $q_\phi(z \mid x)$, which is often referred to as the variational posterior.

Bayes' rule states that the posterior distribution can be computed as follows:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z) p_\theta(z)}{p_\theta(x)},$$

where $p_\theta(z \mid x)$ is the posterior, $p_\theta(x \mid z)$ is the likelihood, $p_\theta(z)$ is the prior, and $p_\theta(x)$ is the marginal likelihood (or evidence). In generative modeling, we are particularly interested in learning $p_\theta(x \mid z)$ and $p_\theta(z)$, since they allow us to sample new data points from the learned distribution. However, computing the posterior $p_\theta(z \mid x)$ or the marginal likelihood $p_\theta(x)$ can be intractable for complex models, necessitating the use of VI for approximation.

VI proposes to find an approximate posterior distribution $q_\phi(z \mid x)$ within a family of tractable distributions $\mathcal{Q}$ parametrized by ϕ. This is achieved by minimizing a certain divergence measure between the true posterior and its approximation. Among several measures of divergence, Kullback-Leibler (KL) divergence is most commonly used in VI.

2.1.3. The Evidence Lower Bound (ELBO)

A crucial concept in Variational Inference is the Evidence Lower Bound (ELBO), which provides a lower bound for the marginal log-likelihood. Optimizing the ELBO facilitates the performance of inference in complex models where direct computation of the posterior is intractable.

The ELBO can be mathematically expressed as:

$$\mathcal{L}(\theta, \phi, x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)].$$

By maximizing the ELBO with respect to both the model parameters θ and the variational parameters ϕ, we aim to approximate the true posterior distribution as closely as possible. It is important to note that the ELBO is a lower bound on the marginal log-likelihood, i.e., $\mathcal{L}(\theta, \phi, x) \leq \log p_\theta(x)$.

2.1.4. Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence serves as a measure of the difference between two probability distributions, and plays a key role in Variational Inference. KL divergence quantifies the difference between the approximating distribution $q_\phi(z \mid x)$ and the true posterior distribution $p_\theta(z \mid x)$.

The KL divergence between two distributions P and Q is defined as:

$$\mathrm{KL}(P \| Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx.$$

In Variational Inference, we minimize the KL divergence between $q_\phi(z \mid x)$ and $p_\theta(z \mid x)$ to improve our approximation of the true posterior. This is directly related to the ELBO optimization, as the difference between the marginal log-likelihood and the ELBO is the KL divergence between the true and approximate posteriors:

$$\log p_\theta(x) - \mathcal{L}(\theta, \phi, x) = \mathrm{KL}(q_\phi(z \mid x) \| p_\theta(z \mid x)).$$

Thus, maximizing the ELBO with respect to θ and ϕ is equivalent to minimizing the KL divergence between the true and approximate posteriors.

2.2. Encoder-Decoder Architectures †

The encoder-decoder architecture is a versatile deep learning framework that has been successfully employed in a variety of tasks, ranging from machine translation (Sutskever et al., 2014), image captioning (Vinyals et al., 2015), and speech recognition (Chan et al., 2016). It consists of two main components: an encoder $q(z|x)$ and a decoder $p(x|z)$. The former maps an input x to a latent space representation z, while the latter reconstructs the original data given z.

2.2.1. Autoencoders †

An autoencoder (Hinton and Salakhutdinov, 2006) is a specific type of encoder-decoder that is trained to reconstruct its own inputs. Autoencoders are commonly used for dimensionality reduction, as the encoder compresses the input into a lower-dimensional latent space. This compressed representation, or embedding, can then be used for downstream tasks such as classification or clustering. Autoencoders are typically trained on unlabeled datasets via supervised learning techniques.

2.2.2. Variational Autoencoders †

Variational Autoencoders (VAEs) (Kingma and Welling, 2013) extend autoencoders by introducing a prior distribution $p(z)$ over the latent space. This prior is often chosen to be a standard Gaussian distribution, and the encoder is trained to approximate the posterior distribution over the latent space given the input. This is achieved through variational inference (see subsection 2.1.2).

The use of a prior distribution over the latent space allows VAEs to be used as generative models. In particular, by sampling from the prior and passing these samples through the decoder, it is possible to generate new samples that resemble the original input data. This makes VAEs a powerful tool for tasks such as text-to-image synthesis, as they provide a probabilistic framework for generating novel images from text descriptions.

2.3. Transformers †

The transformer was first introduced in 2017 by Vaswani et al. as an alternative way to perform sequence-to-sequence modeling. Their architecture forgoes recurrence and convolutions entirely, the mechanisms behind the SOTA models at the time, in favor of an approach based solely on attention (Vaswani et al., 2017). As a result, the transformer is more parallelizable and sees significant speedups in training time. Since its inception, the transformer has received a lot of attention from researchers and numerous architectural updates have been introduced to improve performance on many tasks (Vaswani et al., 2017), including Natural Language Processing (NLP) tasks with the introduction of Large Language Models (LLMs) and vision understanding tasks with the Vision Transformer (ViT). Of interest to this Thesis is its use for natural language understanding in text-to-image models. Because of this, only the encoder stack will be explained.

2.3.1. Tokens

The transformer operates on tokens. Tokens usually encapsulate symbolic data like that of a word or a phrase, but they can generally be used as a representation for any discrete value. The data encapsulated by a set of tokens make up the transformer's vocabulary, and is encoded and decoded by mapping to and from the index of tokens that make up the vocabulary.

Tokenizers

While there is no inherent semantic relationship between the set of index-integers representing the tokens in the transformer, it is important to select tokens that represent information with meaningful relations where such relationships can be learned. This is the job of a tokenizer.

Byte-Pair Encoding

Byte-Pair Encoding (BPE) (Sennrich et al., 2015) is a popular tokenization technique for text, used, among others, by text generation models GPT (Radford and Narasimhan, 2018) and GPT-2 (Radford et al., 2019), and image-text understanding models such as CLIP (presented in section 2.6). Commonly, BPE pretokenizes the data by splitting the text on spaces. After pretokenization, BPE forms a base vocabulary of tokens made up of all the characters that appear in the training data. This base vocabulary is then iteratively grown to include new tokens merged from the most frequent pairs of two tokens already in the vocabulary (Sennrich et al., 2015). BPE grows the vocabulary until it hits a predefined size limit or has encapsulated every pair of tokens in the data.

2.3.2. Embeddings

To learn the semantic relationships between tokens, the transformer uses embeddings (Vaswani et al., 2017). Instead of processing the tokens as unordered integers, the transformer projects them onto a lower-dimensional space as dense vectors, where their semantic relationship is represented by their distance in the embedding space (Lin et al., 2017). By operating in the embedding space, the transformer can learn relations between points in the embedding space rather than between indices in the vocabulary. Since the embedding space supports semantic closeness, the relations learned about a point in the embedding space translates to the points near it, meaning the transformer can generalize across groups of words rather than just across words (Mikolov et al., 2013; Dar et al., 2022).

2.3.3. Attention

Attention was first introduced in 2014 by Bahdanau et al. as a way to perform neural machine translation. Since a source sentence from one language often follow a different structure and can have a different length to that of its translated counterpart in another language, it was not enough to naively match one word in the source sentence to one word in the translation (Bahdanau et al., 2014). To overcome this, Bahdanau et al. proposed what would later be known as attention: to compute a weight for each word in the source sentence for every word in the target sentence. Bahdanau et al. could then attend to the words according to the calculated weight.

Mathematically, this attention process is as simple as two matrices multiplied together, where the first matrix serves as a filter for the second. Perhaps more intuitively, attention can be thought of as the process of selective masking, where a filter weighs the importance of each element in a sequence, and thus how much the element should be attended to, or if it should not be attended to at all.

Scaled Dot-Product Attention

Scaled dot-product attention was introduced with the transformer and builds on the concept of attention (Vaswani et al., 2017). It is natural to look at the scaled dot-product attention mechanism through the eyes of information retrieval theory. In information retrieval, given a query and a set of keys and values, where each key is mapped to a value, the goal is to find the key that best represents the given query, under the assumption that this key's corresponding value is the most representative value to that of the query.

In scaled dot-product attention, shown in Equation 2.1 where Q is the queries, K the keys and V the values matrices, and d_k is the number of dimensions in Q and K, this is done by calculating the similarity between the queries and the keys and then using that similarity score to weigh how much of each value to retrieve (or attend).

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.1}$$

First, the similarity between the queries and the keys are calculated by taking the dot product of the two matrices. When the similarity score between the queries and the keys are calculated, the resulting similarity matrix experiences a shift in variance from that of the input matrices. This is a result of the dot product operation used to calculate the similarity (Vaswani et al., 2017). If this variance shift is not corrected, the values grow out of control as the attention process is repeated. This makes gradient descent unstable and prohibits model learning. In scaled dot-product attention, it is therefore scaled back by the square root of the number of dimensions in the queries and keys matrices, $\sqrt{d_k}$, which makes up the variance after the dot product.

Before the final filtering step, the similarity matrix is passed through the softmax (Bridle, 1989) function. The softmax function normalizes the values between zero and one, and makes sure that their sum equals one, creating a probability distribution over how much attention to pay to each element in the values sequence.

The final step of scaled dot-product attention is then to attend the values matrix with the similarity matrix using another dot product operation.

Masked Scaled Dot-Product Attention

The transformer is often shown tokens it should not be aware of, either as an optimization technique during training or from padding-tokens just used to fill the remaining space in the fixed-length input sequence. These "filler" tokens need to be filtered out of the scaled dot-production attention before the similarity matrix attends to the values. To effectively mask these tokens out of the sequence their similarity score is set to "negative infinity" just before the softmax-step. This makes their softmax score in the succeeding step zero, showing the final step of the attention mechanism that there is absolutely no similarity between the selected queries and keys, and likewise to give no attention to their corresponding values.

Multi-Head Attention

Since the similarity matrix in scaled dot-product attention is a probability distribution, where the total attention sums to one, one such instance alone is unable to focus on many things at the same time without having to focus less on everything. Running multiple instances of scaled dot-product attention at the same time, or "heads", allows each head to attend a small part of the sequence, and the collective of heads to attend to everything important. This is the concept behind multi-head attention (Vaswani et al., 2017).

Like with scaled dot-product attention, the input of the multi-head attention comprises the queries, Q, the keys, K and the values, V, matrices. Then, for each head in the attention mechanism (Vaswani et al. used eight), the embedded matrices are projected onto another lower-dimensional space. It is these transformations that allow the individual heads to process the input differently from each other and learn different concepts, while the dimensionality reduction is just another tool to speed up the computation. Scaled dot-product attention is then applied to the transformed matrices for Q, K and V. The results from each head are then concatenated together and projected back into the embedding space using another learned matrix.

Self-Attention and Cross-Attention

In the transformer, multi-head attention takes two forms: self-attention and cross-attention. Self-attention is used to describe multi-head attention where the input queries, keys and values matrices are the same. Here, the sequence effectively attends to a projection of itself, hence the name. In cross-attention, the input keys and values, which are the same, come from a different embedding, usually produced by an encoder (described further in subsection 2.3.7), while the input queries comes from an earlier layer similar to that in self-attention.

2.3.4. Skip Connections

Surrounding each multi-head attention and Multilayer Perceptron (MLP) block in the transformer are skip connections. Skip connections work by adding the input of a module to its output. This is sometimes called a residual block, and a deep neural network with residual blocks is often referred to as a deep residual network (He et al., 2016).

The primary purpose of a skip connection is to smooth the gradient landscape for backpropagation (Furusho and Ikeda, 2020). Because the softmax function in each attention head returns mostly near-zero values (they all have to add to one across the entire context length), most of the input is filtered out, which artificially flattens the gradients and slows down backpropagation. By adding the original input to the final output, artificial saddle points and ridges in the gradients are smoothed out, while the important features of the gradient landscape from the attention processes are preserved (Furusho and Ikeda, 2020).

Skip connections also serve a secondary purpose. In the case that none of the attention heads in the multi-head attention module attends to an element in the input sequence, this element will be forgotten in the output. Because of the transformer's multi-layer nature (further described in subsection 2.3.7), this means that each layer, which works on the output of the previous layer, might filter out more and more of the signal until all auxiliary information is lost. By adding the input to the output, the skip connection preserves the original sequence for later layers to process.

However, while skip connections help smooth the gradient landscape, they also introduce unwarranted variance to the output (He et al., 2016). This variance grows exponentially with the depth of the residual network, and will lead to a gradient explosion in deep residual networks unless treated.

2.3.5. Layer Normalization

To prevent gradient explosion from skip connections, the output from each skip connection is normalized and scaled to a mean of zero and a standard deviation of one using layer normalization (He et al., 2016).

Layer normalization has empirically been shown to improve performance and help model convergence (Ba et al., 2016). The high-level intuition is that maintaining a consistent distribution across the layers of the model, and throughout training, makes it easier for the model to learn, since neural networks are inherently nonlinear and therefore sensitive to such distribution changes (Ba et al., 2016).

2.3.6. Positional Encoding

Sequence structure is important for many sequence-to-sequence modeling tasks like language understanding and vision. However, the attention module as introduced with the transformer has no inherent understanding of how the tokens in a sequence positionally relate to each other. To make the attention modules positionally aware, the original transformer proposed to encode positional information directly into the input embeddings using absolute positional encoding (Vaswani et al., 2017).

To do this, a positional embedding of the same size is added to the input embedding, moving the points around in the embedding space according to the encoded positional information of the token. The positional embedding is made using sinusoidal positional encoding (Vaswani et al., 2017), as shown in Equation 2.2, where p is the position of a token in the sequence, i is the index of the embedding-dimension and d_{model} is the size of the embedding space.

$$\begin{aligned}\text{PositionalEncoding}(p, 2i) &= \sin(p/10000^{2i/d_{\text{model}}})\\ \text{PositionalEncoding}(p, 2i+1) &= \cos(p/10000^{2i/d_{\text{model}}})\end{aligned} \tag{2.2}$$

Sinusoidal positional encoding applies a sinusoidal and a cosine signal alternatingly to the dimensions in the encoding, with the frequency of the wave in a dimension decreasing as the index of the dimension increases. This means that despite the positional embedding being the same size as the input embedding, only the first few dimensions of the embedding are actually used to encode the positional information. This is important, since it leaves the rest of the dimensions of the input embedding positionally undisturbed.

2.3.7. Encoder Stack

The goal of the transformer encoder as originally proposed by (Vaswani et al., 2017) was to make processed embeddings for the transformer decoder. In the context of text-to-image models, the goal is very similar: make processed embeddings for image-text understanding models such as CLIP. As such, the encoder is fairly simple, consisting of just a list of encoder layers iteratively working on the output from the prior layer, with the first layer working on the positionally encoded embedding of the input sequence. Each layer is made up of a self-attention module (with an accompanying skip connection and layer normalization) and an MLP with one hidden layer (again with an accompanying skip connection and layer normalization; Vaswani et al., 2017).

2.4. U-Nets †

U-Nets are a class of Convolutional Neural Networks (CNNs) that were originally developed for biomedical image segmentation (Ronneberger et al., 2015), but they have since found use in other domains as well, such as image restoration.

A U-Net is an encoder-decoder architecture consisting of a contracting path (the encoder) and an expanding path (the decoder). The contracting path is typically a series of convolutional and pooling layers that reduce the spatial size of the input, leading to a bottleneck that forces the network to learn a more compressed representation of the input. The expanding path then uses this learned representation to upsample the input back to its original size, typically using a series of convolutional and upsampling layers.

The two paths are connected by a series of skip, or residual, connections that concatenate the feature maps from the contracting path with the feature maps from the expanding path. The skip connections let information flow more easily through the network and give gradients a more direct backward path, which makes training easier and more stable. This allows the network to learn complex spatial relationships between the input and the output.

Training a U-Net is usually done in an end-to-end supervised fashion with a regression-based loss. The network inputs are images while the targets are e.g. segmentation masks or denoised versions of the inputs.

2.5. Diffusion Models ‡

Diffusion models are a class of latent variable generative models that rival VAEs and Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) on image generation tasks; They outperform VAEs on sample quality and GANs on mode coverage (Xiao et al., 2021). At the time of writing, they are considered to be the SOTA in image generation and have received a surge in research attention, as they play a key role in the latest SOTA text-to-image models.

The first work on diffusion models was published by Sohl-Dickstein et al. (2015); taking inspiration from non-equilibrium thermodynamics. However, diffusion models did not gain much traction within the field of image generation until the seminal paper by Ho et al. (2020), which introduced Denoising Diffusion Probabilistic Models (DDPMs). Song and Ermon (2019) and Song et al. (2020b) also made important theoretical contributions with their work on Score-based Generative Models (SGMs), which can be considered a generalization of DDPMs within the framework of stochastic differential equations. While DDPMs and SGMs are regarded by some to be two different paradigms within the diffusion field (Yang et al., 2022), practitioners generally do not distinguish between the two. As such, the theory is introduced under the shared term of *Diffusion Models*.

The main concept of diffusion models is to gradually add noise to samples x_0 from our data distribution $q(x)$ through a forward diffusion process $q(x_t \mid x_{t-1})$, resulting in a Markov chain of latent variables $x_1 \dots x_T$, of the same dimensionality as x_0, that get progressively more noisy until x_T can be approximated by random noise. We then try to remove the noise to recover our samples by learning a reverse diffusion process $p_\theta(x_{t-1} \mid x_t)$, parametrized by θ. By adding only a small amount of noise at each step, the problem of learning to undo a single step becomes tractable.

The following sections will provide details on the forward and reverse processes and will cover how to train and sample from the diffusion model they define. These sections are entirely based on the DDPM theory presented by (Ho et al., 2020). Lastly, we will discuss different schemes for conditioning these models on class labels and text captions.

2.5.1. Forward Process

The forward process $q(x_t \mid x_{t-1})$ is usually modeled as a fixed Gaussian process with linear noise schedule $\alpha_1, \dots, \alpha_T$

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, (1 - \alpha_t)\mathbf{I}) \tag{2.3}$$

A nice property of this forward process is that we can sample x_t at an arbitrary timestep t, by using the reparameterization $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \tag{2.4}$$

where $(1 - \bar{\alpha}_t) \to 1$ as $t \to T$, and hence, $q(x_T) \sim \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$. That is, we gradually add small amounts of Gaussian noise to our samples, until they become indistinguishable from standard Gaussian noise. Scaling the data by $\sqrt{\alpha_t}$ ensures that the variance of the data does not grow when adding the noise, which is beneficial, as it provides the deep neural networks in the reverse process with consistently scaled inputs (Ho et al., 2020). Ho et al. propose using $\alpha_1 = 1 - 10^{-4}$, $\alpha_T = 0.98$, and $T = 1000$.

The forward process presented here is fixed, i.e. we do not need to learn it. This allows us to focus our parameter resources on the reverse process, which is what will be used during generation. Note however that it is possible to define a learnable forward process. While this is more complicated, it has been shown to increase performance Nichol and Dhariwal (2021).

2.5.2. Reverse Process

The reverse process $p_\theta(x_{t-1} \mid x_t)$ is a learnable Gaussian process

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.5}$$

where $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are parameterized by deep neural networks, instead of being fixed like in the forward process. That is, we attempt to remove the small amount of noise that was added during one step of the forward process by estimating the denoised mean and variance of the noisy sample using complex non-linear function approximators.

To simplify the process, $\Sigma_\theta(x_t, t)$ is usually set to $\sigma_t^2\mathbf{I}$, where σ_t^2 are time-dependent constants. Ho et al. propose using $\sigma_t^2 = 1 - \alpha_t$. Thus, we only need to model $\mu_\theta(x_t, t)$.

Furthermore, $\mu_\theta(x_t, t)$ is usually reparameterized to

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) \tag{2.6}$$

where $\epsilon_\theta(x_t, t)$ is a denoising network that is trained to predict the noise added to x_t at timestep t. The key takeaway here is that $\epsilon_\theta(x_t, t)$ is the main workhorse of the diffusion model.

2.5.3. Training

While the diffusion processes presented above are somewhat complicated, they essentially make up a training scheme for a distribution-specific denoising network. The forward process generates noisy training data at several noise levels and the reverse process uses this data in a supervised fashion to train the denoising network. The end product is a robust denoising network that is able denoise images, independent of the amount of noise present, by using the knowledge it has learned about the structure of the data distribution it is trained on.

To train the denoising network we choose a random sample x_0 from the dataset and a random timestep from $1, \ldots, T$. In accordance with Equation 2.4, we can add noise to x_0 until it becomes x_t by computing

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{2.7}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ represents the noise added during the timestep (and that we are trying to predict). The denoising network output is then given by $\epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)$.

A loss function such as mean squared error is then used to compute the difference between the predicted noise and the actual noise, and gradient descent is used to optimize this loss. This training scheme is repeated until convergence. The loss function proposed by Ho et al. is given below:

$$L(\theta) = \mathbb{E}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right] \tag{2.8}$$

2.5.4. Sampling

To sample a single reverse step $x_{t-1} \sim p_\theta(x_{t-1} \mid x_t)$ we need to compute

$$x_t = \mu_\theta(x_t, t) + \sqrt{\Sigma_\theta(x_t, t)}z \tag{2.9}$$

where $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. As per subsection 2.5.2, this can be done by computing

$$x_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t z \tag{2.10}$$

To generate a novel sample from the original data distribution $x_0 \sim q(x)$, we repeat the process above for T timesteps, starting from $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. However, in practice, only a fraction of the timesteps are actually employed. For example, sampling using only 20 denoising steps could use timesteps $t \in \{1000, 950, \ldots, 100, 50\}$. This greatly speeds up inference time, at a slight cost to fidelity. This sampling technique is generally referred to as DDPM sampling.

The development of more advanced samplers, which do not require retraining the diffusion model and which provide different tradeoffs between speed and fidelity, is an active area of research. Other notable sampling techniques include Denoising Diffusion Implicit Models (DDIM) (Song et al., 2020a) and the Stochastic Differential Equation (SDE) solvers by Karras et al. (2022).

2.5.5. Denoising Network

The denoising network used in diffusion models is usually a modified U-Net (Ronneberger et al., 2015). To this end, a timestep embedding is added to each convolutional residual block. Ho et al. (2020) use sinusoidal positional encoding as proposed by Vaswani et al. (2017) in the transformer, for their timestep embeddings. This embedding allows the diffusion model to learn to treat different timesteps of the denoising process differently. This is advantageous both from a practical and a theoretical standpoint; it allows the parameters of a single U-Net to be shared across the whole diffusion process, which greatly simplifies training and sampling from the model, and it leverages the fact that there is significant overlap between the denoising tasks of different timesteps. However, the use of separate U-Nets for different parts of the diffusion process is also possible and has been shown to increase model performance, at a great cost to parameter efficiency (Balaji et al., 2022).

To improve the modeling performance of the original U-Net architecture, Ho et al. (2020) introduce a self-attention block with a single attention head between the two convolutional residual blocks at the 16x16 resolution level of the U-Net. This is inspired by the success of transformer architectures (Vaswani et al., 2017). Dhariwal and Nichol (2021) further explored the introduction of the attention mechanism through a series of ablation studies, and found that using self-attention blocks between the residual blocks at the 32x32, 16x16, and 8x8 resolution levels, instead of at just the 16x16 resolution level, and increasing the number of attention heads in the self-attention block, further improves the performance of the U-Net.

2.5.6. Latent Diffusion Models

Latent diffusion models are a variant of diffusion models that operate within the latent space of a frozen VAE (Rombach et al., 2022). The VAE performs perceptual compression with little effect on image quality and semantic content. Compared to pixel-space models, latent diffusion models present several advantages. They are more parameter efficient, requiring smaller U-Nets to achieve comparable resolution and performance to pixel-space models. Additionally, latent diffusion models enable faster training and sampling due to the reduced computational cost of operating in a compressed latent space. Furthermore, these models can create more semantically cohesive images with fewer timesteps during sampling, because the VAE decoder smoothes and interpolates the noisy latent samples. They also facilitate high-resolution synthesis without the need for a cascade of diffusion models to upsample the generated base image, such as in DALL-E 2 (Ramesh et al., 2022) and Imagen (Saharia et al., 2022a), since the VAE decoder upsamples the latent samples. Lastly, these models increase the accessibility of diffusion models as they can be run locally by users and researchers in academia with limited computational resources. Stable Diffusion is an example of a popular free and open-source latent diffusion model (Rombach et al., 2022).

To train a latent diffusion model, one must first train a VAE on the dataset and freeze its parameters. Next, the encoder network of the VAE is used to map data samples x_0 to latent vectors z_0. The forward process $q(z_t \mid z_{t-1})$ is applied to these latent vectors, resulting in a Markov chain of noisy latent vectors $z_1 \dots z_T$. The reverse process $p_\theta(z_{t-1} \mid z_t)$ is then trained on these noisy latent vectors, using the same denoising network architecture, training setup, and loss function as with pixel-space models (see subsection 2.5.3).

Sampling from a latent diffusion model involves first sampling a random noise vector $z_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse process $p_\theta(z_{t-1} \mid z_t)$ is then applied for T timesteps, starting from z_T, using the same sampling technique as in pixel-space models (see subsection 2.5.4).

Finally, the decoder network of the VAE is used to map the final latent vector z_0 to pixel space, resulting in a novel image sample x_0.

2.5.7. Conditional Diffusion Models

While unconditional diffusion models excel at generating high-quality image samples (Dhariwal and Nichol, 2021), their lack of control limits their applicability in real-world applications. Conditional diffusion models, denoted as $p_\theta(x_{t-1} \mid x_t, c)$, significantly enhance the capabilities of diffusion models by integrating additional conditional information (e.g., class labels, text descriptions, and reference images) into the diffusion process. This extends their use to a wider range of tasks, including class-conditional image synthesis, text-to-image synthesis, style transfer, and image interpolation.

Class Conditioning

Class conditioning is a straightforward approach to conditional generation, wherein the diffusion model is conditioned on a singular class label. In practice, this conditioning is implemented by appending an embedding of the class label to the timestep embedding already provided to the denoising network (see subsection 2.5.5). This embedding can be acquired either via a learned embedding layer or a static embedding scheme such as one-hot encoding (Dhariwal and Nichol, 2021).

Despite its ease of implementation, class conditioning presents a significant limitation in terms of its expressiveness due to its reliance on the classes on which the model is trained. Consequently, while it can generate images for classes like “dog” or more specific categories such as “german shepherd”, it falls short when tasked with generating images that require the composition of concepts from different classes. This limitation restricts the generation of images with complex features, such as “a photo of an astronaut riding a horse on the moon”, which would require conditioning on text descriptions (see subsubsection 2.5.7).

Classifier Guidance

An alternative or supplementary approach to class-conditional generation is classifier guidance, proposed by Dhariwal and Nichol in 2021. This technique involves training a separate classifier to categorize the noisy image samples generated during the sampling process. The sampling process is then guided by incorporating gradients from the classifier, scaled by a guidance scale factor, into the mean of the reverse diffusion process. The perturbed mean, $\hat{\mu}_\theta(x_t \mid t, c)$, is computed as:

$$\hat{\mu}_\theta(x_t \mid t, c) = \mu_\theta(x_t \mid t, c) + s \cdot \Sigma_\theta(x_t \mid t, c) \nabla_{x_t} \log p_\phi(c \mid x_t, t) \tag{2.11}$$

where s is the guidance scale and $p_\phi(y \mid t, c)$ is the predicted probability of the target class y by the classifier. Dhariwal and Nichol (2021) observed that increasing the guidance scale enhances the quality of the generated samples, albeit at the expense of sample diversity.

Classifier-Free Guidance

Motivated by the inconvenience of training a separate classifier and the benefits of a unified architecture, Ho and Salimans (2022) proposed a method known as classifier-free guidance. This technique bypasses the requirement of a separate classifier, instead leveraging an implicit classifier defined by the model itself.

Classifier-free guidance operates by occasionally replacing the conditional information with a null value during training. The conditional and unconditional outputs of the model are then combined during sampling using a classifier-free guidance scale, which is similar to the guidance scale used in classifier guidance. The modified prediction, $\hat{\epsilon}_\theta(x_t \mid t, c)$, is computed as:

$$\hat{\epsilon}_\theta(x_t \mid t, c) = \epsilon_\theta(x_t \mid t, \emptyset) + s \cdot (\epsilon_\theta(x_t \mid t, c) - \epsilon_\theta(x_t \mid t, \emptyset)) \tag{2.12}$$

where $s \geq 1$ is the guidance scale. Classifier-free guidance offers two key advantages over classifier guidance: firstly, it enables the diffusion model to utilize its own knowledge during guidance instead of relying on a separate classification model; secondly, it enables guidance using conditional information that is not well-suited for prediction by a classifier (e.g., text descriptions; Ho and Salimans, 2022). However, a disadvantage is that it requires two full forward passes of the denoising network per timestep.

Upsampling Diffusion Models

Image upsampling refers to the process of enhancing the resolution of an image, typically by upsampling a low-resolution version of the image. This can be achieved through various techniques, including conditional diffusion models.

Upsampling diffusion models condition on the low-resolution input image in addition to the noisy latent image and timestep embedding. Usually, the low-resolution image will be upsampled to the target resolution using simple interpolation (e.g., bilinear), before being concatenated channel-wise to the noisy latent image. This allows the model to improve the resolution of the image while preserving its overall structure and content.

Research has shown that diffusion models can achieve impressive results on upsampling tasks (Dhariwal and Nichol, 2021; Saharia et al., 2022b), outperforming other methods in terms of metrics such as Fréchet Inception Distance (FID) and Inception Score (IS) (introduced in section 2.12) as well as human comparison scores.

Upsampling diffusion models are often used in combination with other diffusion models to create a cascade of diffusion models (Ho et al., 2022), where the output of one model is used as the input to the next, allowing for the generation of increasingly detailed images. The simplest example of this is to first generate a low-resolution base image using a regular diffusion model, then generate a final high-resolution image from the base image using the upsampling diffusion model.

Text-Conditional Diffusion Models

Text-conditional diffusion models present a substantial advancement in the domain of image generation by enabling the creation of images based on textual descriptions. The primary challenge involves converting text into a meaningful representation that can be effectively integrated into the denoising network. To achieve this, transformer-based text encoders, such as T5 (Saharia et al., 2022a) or CLIP (introduced in section 2.6), are commonly used to transform textual descriptions into embeddings (explained in section 2.3). These embeddings are then incorporated into the denoising network using techniques like cross-attention (Rombach et al., 2022) or concatenation before the self-attention layers of the U-Net (Nichol et al., 2021), or even into the timestep embedding akin to the class-conditional case (see subsubsection 2.5.7). This approach yields increased flexibility and expressiveness in generating diverse images based on text descriptions (Nichol et al., 2021).

Several popular text-to-image diffusion models have emerged and captured the attention of the general public in recent years, including DALL-E 2, Stable Diffusion, Imagen, and Midjourney. These models are trained on extensive datasets of (image, text) pairs and utilize large pre-trained text encoders to accurately capture the semantic information required for image generation. All of these models employ classifier-free guidance capabilities to enhance the quality of the generated images. Some, such as DALL-E 2 and Imagen, rely on cascaded upsamplers to upscale the generated images to achieve higher resolutions, while others, like Stable Diffusion and Midjourney, utilize latent diffusion models for efficient high-resolution generation.

These advancements in text-to-image diffusion models showcase the potential of combining powerful text encoders with diffusion models for generating visually compelling images that accurately reflect the provided textual descriptions. By employing techniques such as classifier-free guidance, cascaded upsamplers, and latent diffusion, these models have pushed the boundaries of what is possible in conditional image synthesis.

2.6. Contrastive Language-Image Pretraining (CLIP) †

Contrastive Language-Image Pretraining (CLIP) (Radford et al., 2021) was proposed by OpenAI as a way to align related image and text embeddings. The method is capable of zero-shot classification, outperforming SOTA models on various tasks such as visual question answering and natural language inference.

The CLIP model consists of two components: an image encoder that uses either a ResNet-50 (He et al., 2016) or ViT based architecture, and a text encoder using a transformer based architecture. To train the model from scratch, Radford et al. created a dataset consisting of 400M (image, text) pairs collected from the internet.

The overall model flow includes four steps: firstly, feature representations are extracted from each modality; secondly, joint multimodal embedding vectors are calculated; thirdly scaled pairwise cosine similarities between all possible pairs in both modalities are computed; fourthly, these similarities are used to optimize the parameters with a symmetric loss function such as cross-entropy loss.

In summary, CLIP is an effective approach to learning joint representations of images and texts in an unsupervised manner by leveraging contrastive learning principles for better alignment between related embeddings across different modalities. This makes it particularly suitable for zero-shot classification tasks where labels may not be available during training time but can be inferred at test time using only information provided by both visual and textual data points. Furthermore, since CLIP embeddings capture the semantic information of text and images well, they are also good candidates for use in text-to-image models.

2.7. Reinforcement Learning

Reinforcement learning is a branch of artificial intelligence that focuses on training agents to make decisions in an uncertain environment, given limited information. The central problem reinforcement learning aims to solve is to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward in a given environment. It is important in the field of artificial intelligence as it enables the development of agents that can learn and adapt to new situations autonomously. Applications of reinforcement learning span across various domains, including robotics, finance, healthcare, gaming, and natural language processing, among others.

2.8. Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are the foundational framework for modeling decision-making problems in reinforcement learning. MDPs represent the dynamics of an environment, accounting for uncertainty in both state transitions and rewards.

2.8.1. Concept of MDPs

An MDP is defined by a tuple (S, A, P, R), where:

- S denotes the set of states in the environment.
- A denotes the set of actions available to the agent.
- P denotes the transition probability function $P(s'|s,a)$, which represents the probability of transitioning from state s to state s' when taking action a. The transition function must satisfy the Markov property, which states that the future dynamics of the system depend only on the current state and action and not on prior history.
- R denotes the reward function $R(s,a)$ or $R(s,a,s')$, which represents the immediate reward received after taking action a in state s, or after transitioning from state s to state s' via action a, respectively.

2.8.2. Policies and Value Functions

A policy, denoted by $\pi(a|s)$, is a mapping from states to actions that determines the agent's behavior in the environment. Policies can be deterministic, where the agent chooses a specific action given a state, or stochastic, where the agent chooses actions according to a probability distribution. Policies can also be discrete, where the action space is finite, or continuous, where the action space is infinite.

Value functions are used to estimate the expected cumulative reward an agent can obtain from a given state or state-action pair under a specific policy. The state-value function $V^\pi(s)$ is defined as the expected cumulative reward when starting from state s and following policy π:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s \right],$$

where $\gamma \in [0,1)$ is a discount factor that controls the relative importance of immediate versus future rewards, and R_t denotes the reward at time step t. Value functions are crucial in MDPs as they help evaluate and compare different policies.

2.8.3. Bellman Equation

The Bellman equation is a recursive relationship that connects the value function of a state with the value functions of its successor states. It can be expressed as:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma V^\pi(s') \right].$$

This equation states that the value of a state under a policy is equal to the expected immediate reward plus the expected discounted value of the next state when following that policy. The Bellman equation forms the basis for various solution methods in reinforcement learning, such as value iteration, policy iteration, and Q-learning.

2.9. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a modern policy optimization technique that aims to address the challenges of traditional policy gradient methods. In this section, we discuss PPO and its advantages over vanilla policy gradient methods.

2.9.1. Policy Gradient Methods

Policy gradient methods are a class of reinforcement learning algorithms that directly optimize the policy by computing an estimate of the policy gradient and updating the policy parameters using stochastic gradient ascent. The most commonly used policy gradient estimator has the form:

$$\hat{g} = \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right],$$

where π_θ is a stochastic policy parameterized by θ, $\hat{A}_t$ is an estimator of the advantage function at time step t, and the expectation $\mathbb{E}_t[\dots]$ denotes the empirical average over a finite batch of samples.

The vanilla policy gradient method, often called REINFORCE, suffers from high variance in gradient estimates, leading to slow convergence and unstable learning. Additionally, traditional policy gradient methods struggle to handle continuous action spaces effectively. Policy gradient methods can handle continuous action spaces by learning a policy that outputs a probability distribution over actions, instead of directly outputting actions.

2.9.2. PPO and its Advantages

PPO improves upon the basic policy gradient method by using a surrogate objective function to limit the change in policy at each update. PPO is similar to Trust Region Policy Optimization (TRPO), which maximizes a surrogate objective subject to a constraint on the size of the policy update. However, PPO simplifies the optimization problem by avoiding the complicated KL-divergence constraint used in TRPO.

The primary advantage of PPO over traditional policy gradient methods is its stability and robustness in learning. PPO addresses the issue of excessively large policy updates by introducing a clipping mechanism that prevents the policy from deviating too much from the previous policy at each update. This enables PPO to achieve better performance in a wide range of tasks with varying complexity and dynamics.

2.9.3. Algorithm

The PPO algorithm consists of several components, including the clipping technique and multiple epochs of mini-batch updates. The main objective function used in PPO is defined as:

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and ϵ is a hyperparameter (e.g., $\epsilon = 0.2$). The objective function combines the clipped and unclipped surrogate objectives, effectively providing a lower bound (or pessimistic bound) on the unclipped objective while penalizing changes to the policy that move $r_t(\theta)$ away from 1.

The PPO algorithm alternates between sampling trajectories from the environment using the current policy and updating the policy using stochastic gradient ascent on the clipped surrogate objective function. The policy is updated using multiple epochs of mini-batch updates to improve sample efficiency and stability.

In summary, PPO offers a more stable and robust learning algorithm compared to traditional policy gradient methods by introducing a surrogate objective function that limits the change in policy at each update. This enables PPO to achieve competitive performance across a wide range of reinforcement learning tasks.

2.10. Generalized Advantage Estimation (GAE)

Generalized Advantage Estimation (GAE) (Schulman et al., 2015) is a technique for estimating the advantage function used in policy gradient methods. It is closely related to both MDPs and PPO, as it aims to improve the stability and efficiency of policy optimization algorithms.

2.10.1. Importance of Advantage Estimation

Advantage estimation plays a critical role in policy gradient methods, as it helps reduce variance in the gradient estimates and improves learning efficiency. The basic idea behind advantage estimation is to subtract a baseline from the policy gradient estimator:

$$\hat{g} = \mathbb{E}_t\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right].$$

Here, $\hat{A}_t$ is an estimator of the advantage function, which measures how much better an action is compared to the average action in a given state.

2.10.2. Limitations of Existing Methods

Existing advantage estimation methods often face a trade-off between bias and variance. High-variance estimators, such as the Monte Carlo returns, can lead to slow convergence and unstable learning. On the other hand, low-variance estimators, such as TD(λ) returns, can suffer from bias due to bootstrapping, which may result in suboptimal policies.

2.10.3. Introduction to GAE

Generalized Advantage Estimation (GAE) addresses the bias-variance trade-off by introducing a weighting scheme that combines multiple estimators with different bias-variance characteristics. GAE is defined as:

$$GAE(\gamma, \lambda) = \sum_{t=0}^{\infty} (\gamma\lambda)^t \delta_t,$$

where δ_t denotes the temporal difference error at time step t, γ is the discount factor, and λ is a parameter that controls the trade-off between bias and variance.

By adjusting the parameter λ, GAE allows for flexible control over the bias-variance trade-off, resulting in more stable and efficient learning.

2.10.4. Algorithm

The GAE algorithm is integrated within the PPO training loop, as it provides an improved advantage function estimator for policy gradient updates. The algorithm consists of the following steps:

1. Collect a batch of trajectories using the current policy. 2. For each trajectory, compute the temporal difference errors δ_t. 3. Calculate the generalized advantage estimates $GAE(\gamma, \lambda)$ for each time step. 4. Update the policy using the PPO algorithm with the computed GAE values as advantage estimates.

In conclusion, GAE provides a flexible and efficient method for estimating the advantage function in policy gradient methods, addressing the inherent bias-variance trade-off and improving the overall stability and performance of algorithms like PPO.

2.11. Datasets

This section describes the datasets used to train the base policies for the experiments presented in chapter 6 and some of the datasets discussed in chapter 7.

2.11.1. MNIST

The MNIST (Deng, 2012) dataset is a collection of 70,000 images of handwritten digits (0-9). The dataset is divided into 60,000 training and 10,000 test samples (Deng, 2012). The images are grayscale, 28x28 pixel representations. Due its simplistic nature, MNIST is commonly used as a development dataset, where model and algorithm implementations can be tested and validated to ensure that everything is working correctly before the complexity is increased.

2.11.2. ImageNet

ImageNet (Deng et al., 2009) is a popular dataset for image classification. The dataset is structured into WordNet (Fellbaum, 1998) concepts, where each class, often called a "synset", can be described by one or more words or phrases. The goal with ImageNet is to represent each synset with at least 1,000 quality-controlled, human-annotated images. The full dataset has more than 100,000 synsets (Russakovsky et al., 2014), while ImageNet-1k, the dataset used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2014) and the most common subset of ImageNet, has 1000 synsets.

2.11.3. LAION-5B ‡

LAION-5B (Schuhmann et al., 2022) is a large-scale dataset of 5.85 billion English image-text pairs that is open and freely accessible. The dataset was assembled by extracting image links and their accompanying ALT-text from metadata files in Common Crawl web data. These image-text pairs were then automatically filtered using CLIP to remove suspected illegal content or images with bad image-text alignment. LAION-5B is therefore a non-curated dataset not meant for real world use, but provides an opportunity for researchers to test model training on a larger scale (Schuhmann et al., 2022). A subset of LAION-5B was used to train text-to-image model Stable Diffusion (Rombach et al., 2022).

2.11.4. MS-COCO Captions †

Microsoft Common Objects in Context Captions (MS-COCO Captions) (Chen et al., 2015), or more commonly MS-COCO or just COCO, is a dataset with image-text pairs often used to train and evaluate text-to-image models. MS-COCO Captions contains over 330,000 images, each with five human-generated captions. The images themselves were gathered by searching for common object pairs and scenes on popular image sharing site Flickr (Lin et al., 2014).

2.12. Evaluation Metrics †

This section presents Inception Score and Fréchet Inception Distance, the evaluation metrics most commonly used to automatically evaluate text-to-image models and their image fidelity.

2.12.1. Inception Score

The Inception Score (IS) (Salimans et al., 2016) is a metric used to evaluate the quality of generated images in generative models. The IS is calculated by first using an Inception v3 model (Szegedy et al., 2016) trained on the ImageNet (Deng et al., 2009) dataset to classify the generated images and assign them probabilities of belonging to the different classes the Inception model was trained on. The idea is that the more confident the Inception model is in its classification, e.g. by predicting just one class with high confidence, the higher quality the generated image is, and a low confidence prediction, i.e. a predicted class distribution that approaches a uniform distribution, would indicate a lower quality generation (Salimans et al., 2016). The IS is therefore calculated as the average of the Kullback-Leibler divergence (KL-divergence) between the predicted class probabilities and a uniform distribution for all images, with a lower IS indicating a higher quality model.

The reliance on predicting a set of predefined classes made it difficult to assess the quality of generative models trained on other datasets than ImageNet, and the metric fell out of favor as researchers sought more accurate ways to automatically evaluate generative models (Barratt and Sharma, 2018).

2.12.2. Fréchet Inception Distance

Proposed in 2017 by Heusel et al., the Fréchet Inception Distance (FID) is an automatic evaluation metric that attempts to solve some of the problems seen with the IS. Instead of measuring image fidelity based on the confidence of a fixed classifier, like the IS does, FID calculates the Fréchet distance between the statistics from a set of real images to that from a set of generated images (Heusel et al., 2017). Like with the IS, the Inception v3 model is used. But instead of looking at the output of the final classification, the activations from the final pooling layer are used to form two distributions, one for each of the set of images. The FID score is then the Fréchet distance between these distributions, where a lower distance means the two sets of images are more similar.

The FID-30K score is the FID score calculated when 30,000 random images from a dataset are compared against 30,000 generated samples. In 2023, FID score is the most common automatic evaluation metric for text-to-image models, with FID-30K on the MS-COCO Captions validation set being the benchmark for image fidelity.

3. Related Work

In this chapter, we provide an overview of the development of Reinforcement Learning from Human Feedback (RLHF) as a fine-tuning method for aligning generative models with human preferences. We first review the foundational work by OpenAI on aligning Large Language Models (LLMs) using RLHF, which led to notable developments such as ChatGPT (Schulman et al., 2022) and GPT-4 (OpenAI, 2023), and which serves as the main inspiration for our approach. We then cover prior and concurrent works in text-to-image model alignment and discuss their relation to our work.

3.1. Aligning Large Language Models with Human Preferences

In a series of papers, OpenAI introduced and demonstrated the effectiveness of RLHF for aligning transformer-based LLMs, leading to significant improvements in accessibility and safety. Here, we briefly review four milestone papers.

Deep Reinforcement Learning from Human Preferences. The introduction of RLHF can be traced back to the seminal work of Christiano et al. (2017) on deep reinforcement learning from human preferences as a method for learning complex goals in reinforcement learning tasks without direct access to a reward function. The authors demonstrated that this approach could effectively solve challenging reinforcement learning tasks, such as Atari games and simulated robot locomotion, using human feedback on less than one percent of the agent's interactions with the environment. This work significantly reduced the cost of human oversight for State-of-the-Art (SOTA) reinforcement learning systems and demonstrated that complex novel behaviors could be trained within an hour of human time.

Fine-Tuning Language Models from Human Preferences. Building upon Christiano et al.'s work, Ziegler et al. (2019) were the first to apply RLHF to natural language tasks, including text continuation and summarization. They leveraged advances in generative pretraining of LLMs and showed that, with a relatively small number of human-evaluated comparisons, their models could produce high-quality text that aligned with human preferences. This paper demonstrated the feasibility and potential of applying RLHF to natural language processing tasks and laid the groundwork for LLM fine-tuning using RLHF.

Learning to Summarize from Human Feedback. Subsequent work by Stiennon et al. (2020) further explored the application of RLHF for training language models on summarization tasks. The authors demonstrated that fine-tuning LLMs using RLHF significantly improved summarization performance, outperforming both human reference summaries and larger models fine-tuned with supervised learning alone, as assessed by human evaluators. This work provided additional evidence that RLHF can be successfully applied to language-related tasks.

Training Language Models to Follow Instructions with Human Feedback. Advancements to the RLHF methodology by Ouyang et al. (2022) showed that language models could be trained to follow instructions using human feedback. Their approach, dubbed InstructGPT, improved truthfulness and reduced toxic output generation, as assessed by human evaluators, while maintaining performance on public natural language benchmarks when compared to the base GPT-3 (Brown et al., 2020) model. This work demonstrated that RLHF could be applied to a wide range of tasks and added to the growing body of evidence supporting the efficacy of RLHF in aligning language models with human preferences; It represents the main inspiration for our work.

3.2. Aligning Text-to-Image Models with Human Preferences

While aligning LLMs using RLHF has been extensively explored, the field of aligning text-to-image models using human preferences is still emerging. In this section, we review some prior and concurrent works that focus on different aspects of aligning text-to-image models using human preferences.

Optimizing Prompts for Text-to-Image Generation. The work by Hao et al. (2022) is one of the earliest attempts to apply RLHF to improving text-to-image models. They proposed a method called prompt adaptation, which automatically adapts user input to model-preferred prompts. The method consists of two steps: supervised fine-tuning of a pre-trained language model on manually engineered prompts, followed by reinforcement learning of the language model to explore better prompts. Their approach outperformed manual prompt engineering both in terms of automatic metrics and human preference ratings. Although their method shares similarities with ours, such as using RLHF to improve text-to-image models, they optimize an LLM policy for automatic prompt engineering, used as a pre-processing step, without fine-tuning the text-to-image model itself.

3.3. Concurrent Works

Several concurrent works also aim to align text-to-image models using human feedback. Most focus on creating reward models which model human preferences, which is only part of the RLHF framework. Here, we discuss their approaches and how they relate to our work.

Aligning Text-to-Image Models using Human Feedback. Lee et al. (2023) proposed a fine-tuning method for aligning text-to-image models using human feedback in three stages: collecting human feedback, training a reward function, and fine-tuning the text-to-image model using reward-weighted likelihood maximization. Their method generated objects with specified colors, counts, and backgrounds more accurately than the pre-trained model.

This work is closely related to ours, as both approaches aim to align diffusion-based text-to-image models using human preferences by directly fine-tuning the diffusion model. However, there are some differences in methodology: Lee et al. do not use reinforcement learning and instead fine-tune the model using a reward-weighted maximum likelihood objective, which is more similar to standard diffusion training. The authors note that an RLHF-based approach may lead to better models in terms of alignment with human preferences, but left exploration of this to future work as it would require "extensive hyperparameter tuning and engineering".

Better Aligning Text-to-Image Models with Human Preference. Wu et al. (2023) proposed a method called Human Preference Score (HPS) for adapting text-to-image models to better align with human aesthetic preferences. They collected human preferences based on real user interactions with Stable Diffusion and trained a Contrastive Language-Image Pretraining (CLIP)-based reward model. They demonstrated that HPS outperformed existing scoring methods and had good generalization capability toward images generated from other models. By tuning Stable Diffusion with the guidance of HPS, their adapted model was able to generate images that were more preferred by human users.

While this work is closely related to ours, it does not employ reinforcement learning for fine-tuning. Instead, the authors opt for the much simpler option of fine-tuning the text-to-image model using Low-Rank Adaptation (LoRA) (Hu et al., 2021) on a reward-filtered dataset.

ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation. Xu et al. (2023) proposed ImageReward, a BLIP-based (Li et al., 2022) general-purpose text-to-image human preference reward model trained on a large dataset of expert comparisons. ImageReward outperformed existing scoring methods in human evaluation and was presented as a promising automatic metric for evaluating and improving text-to-image synthesis. Although Xu et al.'s work only proposes a reward model, the authors note that their reward model could be used with future RLHF frameworks for text-to-image models, such as our proposed method.

Pick-a-Pic: An Open Dataset of User Preferences for Text-to-Image Generation. Kirstain et al. (2023) introduced Pick-a-Pic, an open dataset containing over 500,000 images generated from 35,000 distinct prompts and combined with real user preferences collected through a custom web interface. Similar to Wu et al.'s work, Kirstain et al.'s approach is based on human preferences collected from real user interactions with Stable Diffusion. They used this dataset to train a CLIP-based reward model called PickScore, which demonstrated superhuman performance on predicting human preferences and correlated much better with human rankings than prior automatic evaluation metrics, such as HPS and ImageReward. While their work provides a valuable resource for the research community, they do not explore the application of their reward model in aligning text-to-image models with RLHF.

3.4. Conclusion

In summary, the field of aligning generative models using RLHF has seen significant progress in recent years. The success of RLHF in aligning large language models inspired our work on applying this approach to diffusion-based text-to-image models. Although our work is the first to propose using RLHF to directly fine-tune diffusion-based text-to-image models, some prior and several concurrent works have explored similar problems.

While the field is still in its infancy, we anticipate that there will be significant advancements in RLHF-based alignment for text-to-image models in the coming months. By understanding the current SOTA and incorporating lessons learned from related works, our research aims to contribute to pushing the boundaries of what can be achieved with text-to-image models aligned with human preferences.

4. Human Data Collection

This chapter explains the first two steps of the methodology presented in this Thesis (see section 1.3 for a high-level overview of the methodology). First, the methodology for task specification and labeling dataset curation is presented (step 1). This involves generating sample images for labeling in tasks of fixed sizes based on curated prompts that cater to the base diffusion model's knowledge. Then, the methodology for human preference collection is explained (step 2). The human labelers are given tasks of images generated in the first step and are asked to assemble a total preference order from the images within the tasks. To facilitate this process, a custom labeling software is employed.

4.1. Task Specification and Sample Data Generation

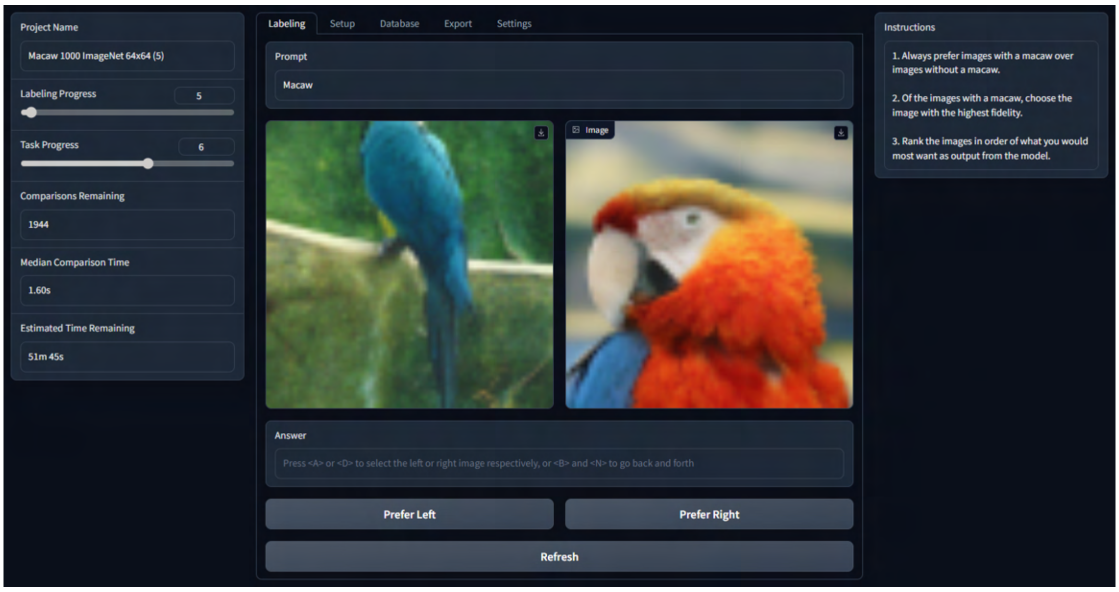
Before a human labeler can start collecting preferences, sample images for labeling are generated. The images are generated in tasks of size K, with all images in a task generated using the same conditioning prompt. The task prompts are sampled randomly from a list of prompts curated at the start of the experiment.

Since the proposed methodology is an alignment technique, the list of prompts should be curated based on the knowledge already ingrained in the base diffusion model. We therefore limit task prompt selection to only that which the model is already somewhat capable of doing when curating the prompt list. We theorize that to achieve an efficient reward model training, most tasks should contain a range of both "good" and "bad" generations, to make the labeled order within each task as accurate to its true distribution as possible. Task size specification then becomes a trade-off between a more accurate order (higher K) and faster labeling (lower K).

Exact values for various sampling details, dataset size and task size are specified on a per-experiment-basis in chapter 6.

4.2. Human Preference Collection

After a labeling dataset has been curated, the evaluators can start collecting human preferences. Adapting the preference collection methodology introduced by Stiennon et al. (2020), we present a task to the human labeler and ask them to create a total preference ordering of images within that task. To do this, the human labeler is presented with two



Project Name
Macaw 1000 ImageNet 64x64 (5)

Labeling Progress
5

Task Progress
6

Comparisons Remaining
1944

Median Comparison Time
1.60s

Estimated Time Remaining
51m 45s

Prompt
Macaw

Answer
Press <A> or <D> to select the left or right image respectively, or <B> and <N> to go back and forth

Instructions

1. Always prefer images with a macaw over images without a macaw.

2. Of the images with a macaw, choose the image with the highest fidelity.

3. Rank the images in order of what you would most want as output from the model.

Figure 4.1.: Screenshot of the custom labeling software. Captured during the preference collection of Experiment 4: *Macaw*.

images from the same task, as well as the task's prompt, and is assigned with selecting which of the two images they prefer given a set of static, predefined instructions. After a preference is selected, another pair of images from the same task is presented to the human labeler. Once all the images in a task have been compared against one another, the human labeler moves onto the next task.

Labeling Software. Figure 4.1 shows the "Labeling" pane of our custom preference collection software. In the middle of the screen is the main interface for making image-to-image comparisons. At the top of this interface is a box with the current task's prompt. Below the task prompt are the two images being compared. When the labeler decides which image they prefer given the instructions and the task prompt, a button under the corresponding image can be pressed. Alternatively, the labeler can press a key on the keyboard to quickly mark their preference and move onto the next comparison. The labeling software also supports going back and forth between comparisons to relabel images if the labeler changes their opinion or makes a mistake.

Whenever a preference comparison is made, the data is saved to a database for reliable storage. This ensures the human labeler can continue from where they left off and not be afraid of losing valuable time and progress in the event that something were to happen with the labeling software.

The labeler has access to two additional panels of information at all times. The panel on the right-hand side of the workspace displays the instructions for the preference collection.

The panel on the left-hand side of the workspace displays the labeler's progress. This panel shows which task they are currently labeling and an estimate of the remaining time to finish labeling.

Further images of the software as well as a more thorough description of its design and functionality is available in Appendix E.

Transitivity. Following the definition of a labeled task as a total ordering of images, we can assume that a task is transitive. By assuming transitivity, the amount of image-to-image comparisons the human labeler needs to make is greatly reduced. To support transitivity, the labeling software uses binary insertion sort (Auddy, 2023). Binary insertion sort combines binary search with insertion sort to efficiently locate the position in the order of sorted images to insert the next image. This reduces the comparisons needed to order a task, to the comparisons that make up the binary search. For this reason, binary insertion sort is especially efficient for labeling small tasks (we use task size $K = [5, 8]$; Auddy, 2023).

Transitivity assumes consistent human labeling. When the human labelers become inconsistent, cycle comparisons can appear, breaking the total ordering. This limitation still holds when labeling in the software's transitivity mode, but here such potential cycles are effectively broken by which comparison the labeler is presented with first. In experiments where we use transitive labeling, we therefore try to naturally guide the labeler to acyclic labeling by constructing instructions that promote transitivity, rather than relying on arbitrarily solving cycles in software.

5. Reinforcement Learning for Fine-tuning Diffusion Models

This chapter presents a novel approach for aligning diffusion-based text-to-image models with human preferences using Reinforcement Learning from Human Feedback (RLHF). The chapter begins by reframing the reverse diffusion process as a Markov Decision Process (MDP), which allows for the application of reinforcement learning techniques to optimize the diffusion model policy. The chapter then moves on to a detailed presentation of the proposed reward modeling approach, which is designed to capture complex human preferences and handle simultaneous text and image input. The reward model architecture and training methodology are discussed, with insights drawn from related studies in reinforcement learning for fine-tuning language models. Lastly, the chapter presents the policy optimization setup, focusing on an implementation of Proximal Policy Optimization (PPO) tailored to work with diffusion models

5.1. Reframing the Reverse Diffusion Process as a Markov Decision Process

Our approach involves reframing the reverse diffusion process, defined as a Markov chain (see section 2.5), as a MDP. To this end, we present how states, actions, and reward functions can be defined in the context of diffusion models in the following subsections.

State Space. The state space S is defined by the original conditional diffusion model as $s_t = (x_t, t, c)$, where x_t is the intermediate noisy image latent at timestep t in the diffusion process, t is the current diffusion timestep, and c represents the conditioning (i.e., text prompt). The initial state is denoted as $s_T = (x_T, T, c)$ with x_T being randomly sampled Gaussian noise. The terminal states are given by $s_0 = (x_0, 0, c)$ and correspond to completed image samples.

Action Space. The action space A is defined by a multivariate Gaussian distribution with mean μ_t and diagonal standard deviation σ_t, as produced by a single reverse step of the diffusion model[1], for the associated timestep. The action space is continuous and

[1]i.e., a pass through the U-Net, producing ϵ_t, which is transformed into μ_t and σ_t by the noise schedule.

represents pixel values[2], depending on the diffusion model employed. Actions are sampled from this distribution, $a_t \sim \mathcal{N}(\mu_t, \sigma_t)$, akin to a reparameterization trick. In addition to the randomized initial states, this sampling process makes the MDP environment stochastic.

Transition Function. A key observation about the action space is that sampling a_t is equivalent to Denoising Diffusion Probabilistic Model (DDPM) sampling of x_{t-1}, hence, an action directly defines its subsequent state. This allows for reparameterization of the reverse diffusion process, which originally defines a state transition function, into a policy: $\pi_\theta(a_t|x_t) \coloneqq p_\theta(x_{t-1}|x_t)$. This explicitly defines the state transition function as $P(s_{t-1}|s_t, a_t) \rightarrow (a_t, t-1, s_t[c])$. Furthermore, the MDP trajectories can be represented by the original reverse diffusion trajectories: $\tau = (s_T, s_{T-1}, \ldots, s_1, s_0)$. However, as per standard diffusion model sampling, only a select few timesteps from the trajectory are actually employed during the sampling process (see section 2.5).

Reward Model. The reward model, denoted $r_\phi(s_t) : S \rightarrow \mathbb{R}$, produces a scalar reward that is zero for all states except for terminal states s_0 (reward is given at episode end, similar to a bandit problem). The reward should be high for terminal states (completed image samples) that are likely to be preferred by humans and low for terminal states that are less likely to be preferred. The model parameters ϕ are learned from a dataset of human comparisons, where each comparison consists of a pair of terminal states and a label indicating which state is preferred by a human (see chapter 4). The training objective for ϕ is to minimize the disagreement between the reward model's predictions and the human labels. This approach is known as reward modeling and is often used in reinforcement learning to learn complex tasks that are difficult to define with a simple, hand-crafted reward function. Our proposed reward model for accomplishing this is detailed in the following section.

5.2. Reward Modeling

In this section, we present the architecture of our proposed reward model. The reward model serves as a proxy for human preferences, guiding the fine-tuning of the diffusion model policy in a reinforcement learning environment. The following sections describe the design criteria and inspiration behind our proposed reward model, as well as the architecture and training methodology.

Design Criteria and Inspiration. The reward model is designed with several key criteria in mind:

[2] or latent values if the diffusion model operates in a latent space.

1. The reward model should have sufficient expressive power to capture complex human preferences.
2. The reward model should be multimodal, enabling it to handle simultaneous text and image input, as text-image alignment is an essential part of human preferences.
3. The reward model should produce a scalar reward for each text-image pair provided.

To address these criteria, we derive inspiration from two related studies: Stiennon et al. (2020) and Dhariwal and Nichol (2021).

Stiennon et al. (2020) investigate the fine-tuning of Large Language Models (LLMs) using RLHF. In their work, they initialize the reward model from the same base model as the policy. This approach ensures that the reward model has comparable expressive power to the policy being evaluated and guarantees that both models operate in similar domains, thereby facilitating a consistent understanding of the inputs.

Dhariwal and Nichol (2021) propose classifier guidance for diffusion models (refer to subsubsection 2.5.7). They employ U-Net encoder blocks analogous to those used in constructing their diffusion models, in conjunction with a final attention pooling layer (Radford et al., 2021), to develop their classifier architecture. The classifier is trained from scratch to predict the conditions associated with intermediate noisy images during the diffusion process. Consequently, their architecture does not simultaneously accommodate both images and conditions as input.

Architecture. Building on the design criteria and drawing inspiration from Stiennon et al. (2020) and Dhariwal and Nichol (2021), we propose the following reward model architecture:

1. Start with a pre-trained U-Net of a diffusion-based text-to-image model, consisting of an encoder, middle bottleneck, and decoder.
2. Remove the decoder and feed the output of the bottleneck into a newly initialized cross-attention pooling layer instead. This layer attends to both text and image, producing a scalar reward output.
3. Freeze all parameters except for the new cross-attention parameters to preserve the encoding capabilities and alignment with the initial policy model.

The architecture meets our design criteria by incorporating both text and image input using a cross-attention mechanism, and the large U-Net encoder ensures that the reward model has great expressive power. Furthermore, by leveraging the pre-trained U-Net from the diffusion-based text-to-image model, our architecture guarantees that the reward model operates in a similar domain as the policy being evaluated.

Another benefit of this approach is that when the diffusion model employed is a latent diffusion model, which is the case for our experiments involving Stable Diffusion, our reward model can directly take latent input. As a result, there is no need to decode images produced by the policy before feeding them into the reward model, which provides a stronger coupling between the text-to-image model and reward model, additional computational efficiency, and avoids lossy decoding-encoding.

Training Objective. The training process for our reward model is based on the methodology presented by Ouyang et al. (2022), which is, in turn, inspired by Stiennon et al. (2020). We train the reward model on a dataset of comparisons between pairs of image samples generated using the same prompt by the base diffusion model (see chapter 4). A cross-entropy loss is utilized, with differences in rewards indicating the log odds of one response being preferred over the other by a human labeler.

Following Ouyang et al. (2022), to speed up comparison collection, labelers rank more than two images per prompt, generating a set of pairwise comparisons for each prompt (see chapter 4). However, treating each comparison as a separate data point can lead to overfitting (Ouyang et al., 2022). To address this issue, we train the model using all pairwise comparisons from each prompt as a single batch element. This approach is more computationally efficient, requires fewer forward passes of the reward model, and helps avoid overfitting.

Specifically, the loss function for the reward model is:

$$L(\phi) = \frac{1}{\binom{k}{2}} \mathbb{E} \left[\log \left(\sigma \left(r_\phi(x_{\text{winner}}) - r_\phi(x_{\text{loser}})\right)\right)\right] \tag{5.1}$$

where K is the number of image samples per prompt, σ is the sigmoid function, $r_\phi(x_{\text{winner}})$ and $r_\phi(x_{\text{loser}})$ represent the rewards for the winning and losing images, respectively.

Classifier-Free Reward Guidance. We retain the classifier-free guidance capabilities of the original diffusion model by randomly dropping the text condition during training. This approach allows us to control how much weight to assign to text conditioning after training, ensuring that the text plays a crucial role in determining the reward and maintaining the importance of text-image alignment.

Reward Standardization. Following Ouyang et al. (2022), we standardize[3] the rewards produced by the reward model before using them for reinforcement learning. Since the reward model loss is invariant to shifts in reward, we standardize the reward model to produce a mean reward of 0.0 and a standard deviation of 1.0, given the training dataset. This standardization makes it easier to compare reinforcement learning runs using different reward models and reduces the need to adjust hyperparameters.

[3]Also referred to as *normalization* in the literature

$$\tilde{r}_\phi(x) = \frac{r_\phi(x) - \mu_r}{\sigma_r} \tag{5.2}$$

where μ_r and σ_r are the mean and standard deviation of training dataset rewards respectively.

KL Regularization. Following Ziegler et al. (2019), we modify the reward model by adding a per-timestep KL penalty from the base diffusion model to mitigate mode collapse of the learned generative distribution and overoptimization towards the reward model. This regularization technique keeps the diffusion policy from diverging too much from the base diffusion model, and subsequently from the range where r_ϕ is valid (Ziegler et al., 2019). Additionally, it serves as an entropy bonus, and replaces the traditional entropy bonus in most available implementations (Ziegler et al., 2019). The modified reward model can be written as:

$$R(s_t) = \tilde{r}_\phi(s_t) - \beta\,\mathrm{KL}(\pi_\theta \parallel \pi_{\mathrm{Base}}) \tag{5.3}$$

where π_θ is the policy model, π_{Base} is a frozen copy of the policy before optimization, β is the coefficient of the KL penalty, and KL is an estimator of the KL divergence. While Ziegler et al. (2019) uses $\mathrm{KL}(p \parallel q) = \log(p/q)$, we use $\mathrm{KL}(p \parallel q) = ((p/q) - 1) - \log(p/q)$, as it is an unbiased estimator with lower variance (Schulman, 2020).

5.3. Policy Optimization

With the MDP formulation established, it is possible to apply policy gradient methods such as REINFORCE, Trust Region Policy Optimization (TRPO), and PPO to optimize the policy π_θ. Following Ziegler et al. (2019), we chose to use PPO in all our experiments.

Our implementation of PPO is loosely based on the continuous PPO implementation in the CleanRL library[4]. Our adaptation of PPO for RLHF is inspired by the available LLM approaches, notably, OpenAI's lm-human-preferences[5], CarperAI's trlX[6], and Lucidrain's PaLM-RLHF[7].

The following subsections describe our PPO setup, which includes an overview of the models used and the training loop (rollout + optimization). Experiment-specific parameters are presented with the relevant experiments in chapter 6.

[4]Available at: `https://github.com/vwxyzjn/cleanrl`
[5]Available at: `https://github.com/openai/lm-human-preferences`
[6]Available at: `https://github.com/CarperAI/trlx`
[7]Available at: `https://github.com/lucidrains/PaLM-rlhf-pytorch`

Models. The main two models needed for our method are the policy model π_θ and reward model r_ϕ. These are both initialized from diffusion models; we use a 64x64 ImageNet-1k model[8] by OpenAI in our class-conditional experiments and Stable Diffusion v1.5[9] by Runway in our text-to-image experiments. Additionally, we make copies of both to use as the base model and the value function respectively. The base model is used for computing KL penalty, while the value function is needed for value estimation in PPO (see section 2.9). Parameters are not shared between any models and are frozen for the reward and base models. Remark that, unlike Stiennon et al. (2020), the models are not initialized from supervised-finetuned weights.

Rollout. During rollout, sampling trajectories are generated from multiple parallel environments and stored in a rollout buffer for use during optimization. The image generation settings used are shared across all trajectories and are the same as the ones used during dataset curation for reward model training (see chapter 4). The only difference between trajectories is the text condition used, which is uniformly sampled from our prompt dataset, and the initial state Gaussian noise x_T. This ensures that the model operates close to the domain where the reward model r is valid, and simplifies implementation. Notably, employing the same timestep schedule for all trajectories ensures that they have the same length and terminate in unison, which allows for a more efficient storage layout in the buffer and better utilization of GPU resources.

Rollout is equivalent to standard diffusion model sampling (see section 2.5), the only difference being that the intermediate states are saved to the rollout buffer with associated log-likelihoods, values, and rewards.

Log-likelihoods are computed under the action distribution on a per-pixel basis, then summed over the action dimensions to produce the joint log-likelihood for the state. Values are computed by the value function. Rewards are computed in accordance with section 5.2. To compute the KL penalty, log-likelihoods have to be computed under the action distribution of the base policy as well. The completed image samples are not saved to the rollout buffer, but are kept for logging/evaluation purposes.

Optimization. During optimization, the PPO algorithm updates the policy model π_θ based on the states, log-likelihoods, values, and rewards collected in the rollout buffer. The optimization procedure is performed with several modifications from the original PPO algorithm proposed by OpenAI, following current best practices (Schulman et al., 2017; Huang et al., 2022). Some of these modifications include normalizing advantages and clipping values. The implementation is similar to the one used in lm-human-preferences (Ziegler et al., 2019). Advantages are estimated using Generalized Advantage Estimation (GAE) (Schulman et al., 2015).

[8]Model weights available at: `https://github.com/openai/guided-diffusion`
[9]Model weights available at: `https://huggingface.co/runwayml/stable-diffusion-v1-5`

The rollout buffer contents are utilized as a batch for optimization, where the batch size corresponds to the number of environments multiplied by the number of timesteps used for sampling. The batch is split into multiple minibatches, with specific values for these quantities varying across experiments. Notably, minibatches comprise random states from different environments, as opposed to keeping states from same trajectory grouped. This is a deviation from OpenAI's approach, where all tokens in a response are treated as a single batch element. We leave an investigation of how this affects performance to future work.

For optimization, we employ the AdamW optimizer (Loshchilov and Hutter, 2017) with separate learning rates for the policy and value function. Learning rates are constant but include a linear warmup for a specified number of optimization steps. Gradient clipping is applied to stabilize learning, while gradient accumulation and gradient checkpointing can be optionally employed to adjust minibatch size and save on VRAM usage. Most experiments utilize full-precision (FP32), but we also experiment with mixed-precision techniques such as FP16, TF32, and 8-bit optimizer (see chapter 6).

6. Experiments

This chapter presents the series of experiments conducted to iteratively develop and test the methodology laid out in chapter 4 and chapter 5. These experiments were designed to address the research questions posed in chapter 1 and contribute insights that help achieve the overarching goal of this thesis. In the following sections, an experimental plan outlining the high-level structure of the experiment series is provided, followed by each experiment's description, setup, results, and evaluations.

6.1. Experimental Plan

The experiment series is divided into three sections of increasing complexity and with an overarching goal per section. The end goal was to attempt to align a state-of-the-art text-to-image model (e.g., Stable Diffusion) using human preferences. To accomplish this, the experiments started with simple diffusion models trained on restricted datasets like MNIST and ImageNet-1k, low-resolution images like 32×32 and 64×64, and unconditional or class-conditional settings, gradually incorporating human feedback, before moving on to the more complex and higher-resolution text-to-image model. This allowed for scaling up the approach as it proved successful and ensured a solid understanding of the methodology as it evolved.

A high-level overview of the experiment series is presented in Table 6.1.

Section 1: Aligning Simple Diffusion Models Using Fixed Reward Functions. The first section of the experiment series focused on aligning simple diffusion models using fixed reward functions. The main goal was to verify that diffusion models could learn from reward signals. We defined reward functions such that it was trivial to observe the optimization process, making these experiments useful for testing and debugging during the initial implementation phase of our methodology.

Section 2: Aligning Simple Diffusion Models Using Human Feedback. The second section of the experiment series focused on aligning simple diffusion models using human feedback. The main goal was to incorporate human feedback into the reward model and to verify that the optimized policy effectively learned the human preferences captured by the reward model.

Num.	Experiment Name	Base Policy	Reward Model
Section 1: Aligning simple diffusion models using fixed reward functions			
1	*Target Image*	32×32 MNIST (subset)	Fixed function: modified RMSE to a target image
2	*Colorfulness*	64×64 ImageNet-1k	Fixed function: color vividness and diversity
Section 2: Aligning simple diffusion models using human feedback			
3	*Aesthetic Score*	64×64 ImageNet-1k	Existing model: LAION Aesthetics Predictor V2
4	*Macaw**	64×64 ImageNet-1k	Custom model: trained to capture human preferences for the macaw class of ImageNet-1k
5	*Macaw: Without RL*	64×64 ImageNet-1k	Custom model: trained to capture human preferences for the macaw class of ImageNet-1k
Section 3: Aligning diffusion-based text-to-image models using human feedback			
6	*Pink**	Stable Diffusion v1.5	Custom model: trained to capture human preferences for 4 basic prompts of animals wearing pink sunglasses
7	*PickScore*	Stable Diffusion v1.5	Existing model: PickScore v1

Table 6.1.: High-level overview of the experiment series. Experiments marked with * constitute the main tests of our methodology, following all four steps outlined in section 1.3.

Section 3: Aligning Diffusion-Based Text-to-Image Models Using Human Feedback. The third section of the experiment series aimed to align state-of-the-art text-to-image models using human preferences. This involved scaling the method to much larger and more complex diffusion models, which posed technical challenges, such as multi-GPU and mixed precision training.

6.2. Experimental Setups, Results, and Evaluations

In the following sections, descriptions, setups, results, and evaluations for each experiment in the series are presented, organized according to the high-level structure outlined in Table 6.1.

Table 6.2 provides an overview of hyperparameters for all experiments. It is provided for completeness, as a reference to the reader, and to anyone wishing to recreate our experiments.

6.3. Training Hardware

The experiments were run using two different hardware configurations. For the smaller experiments, a machine equipped with a single NVIDIA GeForce RTX 4090 24 GB GPU was utilized. For the text-to-image experiments, which required more VRAM, the IDUN High-Performance Computing (HPC) cluster (Själander et al., 2019) was utilized. Specifically, a Dell DSS 8440 node with 10 × NVIDIA A100 80 GB GPUs was employed. Refer to Table 6.2 for an overview of the hardware configuration used per experiment.

Section 1: Aligning Simple Diffusion Models Using Fixed Reward Functions

6.4. Experiment 1: Target Image

The first experiment, dubbed *Target Image*, is a simple experiment involving a very basic diffusion model and reward function. The experiment was designed such that the optimization process yielded an easily observable binary outcome: it worked if all images converged on the same target image, and it did not work if they did not. Hence, we are trying to intentionally overfit the model using reward signals.

This experiment does not follow the methodology presented in section 1.3. The reward model was fixed to a simple function, thus only *Step 4* of the methodology was carried out and tested. The primary motivation was to demonstrate that diffusion models could be trained using the methodology for mapping diffusion models to a reinforcement learning environment (see section 5.1). A secondary motivation was to validate the implementation of Proximal Policy Optimization (PPO) and to become more familiar with the hyperparameter values of the training scheme.

6.4.1. Experimental Setup

A custom 32x32 diffusion model was trained using the improved-diffusion[1] code by OpenAI. The model was trained on a subset of the MNIST dataset consisting of only the images of the digit “4” (see subsection 2.11.1 for more information on MNIST). Since the MNIST dataset is 28×28, it was first upsampled to 32×32 using bilinear interpolation. This choice of increasing the resolution to 32×32 was made as a necessity given the constraints of the improved-diffusion code, which only supports resolutions of 32, 64, and 256 (without modifications). For simplicity, training on all digits was not conducted, as

[1]Available at: `https://github.com/openai/improved-diffusion`.

	Target Image	**Colorful-ness**	**Aesthetic Score**	**Macaw**	**Macaw: Without RL**	**Pink**	**PickScore**
Diffusion							
Base Policy	32 × 32 MNIST (subset)	64 × 64 ImageNet-1k	64 × 64 ImageNet-1k	64 × 64 ImageNet-1k	64 × 64 ImageNet-1k	Stable Diffusion v1.5	Stable Diffusion v1.5
Resolution	32 × 32	64 × 64	64 × 64	64 × 64	64 × 64	512 × 512	512 × 512
Cond. Type	Uncond.	Class	Class	Class	Class	Text	Text
CFG Scale	-	-	-	-	-	7.5	7.5
Latent Diff.	7	7	7	7	7	3	3
Timesteps	25	32	50	25	25	16	50
Reinforcement Learning							
RM Type	Fixed function	Fixed function	Existing model	Custom model	Custom model	Custom model	Existing model
Environments	32	32	32	32	32	80 (8 × 10)	80 (8 × 10)
GAE γ	1	1	1	1	-	1	1
GAE λ	0.95	0.95	0.95	0.95	-	0.95	0.95
PPO Clip Coeff.	0.2	0.2	0.2	0.2	-	0.2	0.2
PPO VF Coeff.	0.5	0.5	0.5	0.5	-	0.1	0.1
KL Penalty Coeff.	0	0.02	0.03	0.02	-	0	0
General Optimization							
Epochs	3000	400	500	250	750	25500	20850
Minibatch Size	25	32	25	25	100	16	10
Learning Rate Policy	1e-7	1e-6	5e-7	1e-6	1e-6	1e-7	5e-7
Learning Rate VF	1e-5	1e-5	1e-5	1e-5	-	1e-6	1e-6
Warmup Steps Policy	4096	8192	8192	8192	100	4096	4096
Warmup Steps VF	1024	1024	1024	1024	-	1024	1024
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Adam β_1	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Adam ϵ	1e-5	1e-8	1e-8	1e-8	1e-8	1e-8	1e-8
Adam Weight Decay	0.01	0	0	0	0.01	0.01	0.01
Grad. Accum. Steps	1	1	1	1	8	2	4
Grad. Clip Norm	0.5	0.5	0.5	0.5	-	1	1
Hardware							
Precision	FP32	FP32	FP32	FP32	FP32	FP16	FP16
Allow TF32	7	7	7	7	7	3	3
GPUs	4090	4090	4090	4090	4090	10×A100	10×A100

CFG: Classifier-free guidance, RM: Reward model, VF: Value function

Table 6.2.: Model parameters and optimization settings for all experiments.

it restricted the image distribution while still maintaining some diversity to avoid making the task too trivial. Consequently, the model was also kept unconditional.

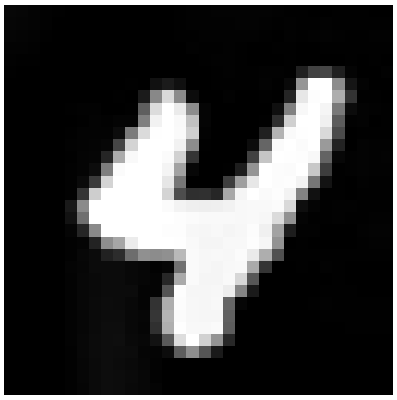


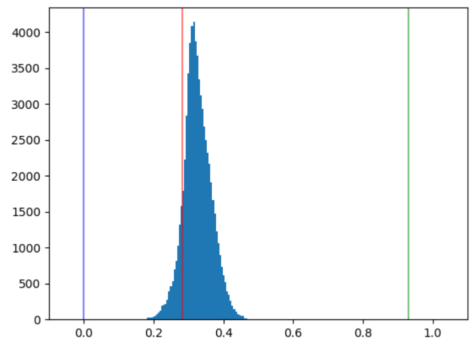
Figure 6.1.: Target image for the *Target Image* experiment.

Algorithm 1 Algorithm for computing target image reward.

```
1: function TargetImageReward($image$, $target$)
2:     Normalize $image$ and $target$ to have values between 0 and 1
3:     $loss \leftarrow MSE(image, target)$
4:     $loss \leftarrow loss + loss \cdot target/mean(target)$ ▷ Adjust the loss to avoid local minima
5:     $loss \leftarrow mean(loss, dim = (1, 2, 3))$           ▷ Reduce loss over image dimensions
6:     $loss \leftarrow \sqrt{loss}$                                          ▷ MSE to RMSE
7:     $reward \leftarrow -loss \cdot 10$                     ▷ Loss is negated for gradient decent
8:     return $reward$
9: end function
```

The reward function was based on a fixed target image of the digit "4" (see Figure 6.1). A digit with significant font weight and a slanted appearance was chosen to make it distinct. Algorithm 1 outlines the computation of the target image reward, which is based on an adjusted Root-Mean-Square Error (RMSE) loss. The loss was adjusted in such a way as to reduce the likelihood of the model getting stuck in a local minima consisting of purely black images. Figure 6.2 shows the effect of the adjustment on the base model error distribution. The loss was negated to transform it into a reward, where a lower RMSE corresponds to a higher reward. Lastly, the loss was scaled by an arbitrary constant (10) to increase the value range of the reward. The arbitrary scaling was incorporated with the rationale that a larger reward scale could help accentuate the differences in reward between different potential actions, thereby promoting more discerning action selection and faster convergence during training. The reward was not standardized according to the methodology (see *Reward Standardization* in section 5.2).

In this experiment, the KL penalty (see *KL Regularization* in section 5.2) was not included in the reward calculation as the aim was to collapse the distribution into a single image


(a) Unadjusted

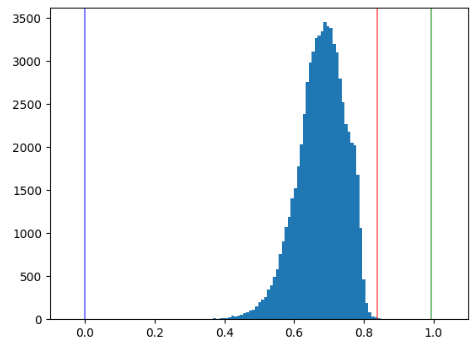

(b) Adjusted

Figure 6.2.: Error distributions for the adjusted and unadjusted reward functions on the digit-“4” subset of MNIST. The vertical blue line (left) represents the target objective (0 error). The vertical red line (middle) represents the error for a full-black image (local minima). The vertical green line (right) represents the error for a full-white image. For the adjusted reward function, the mean of the distribution (starting point for policy optimization) yields a lower error than a full-black image (local minima).

(i.e., a Dirac delta distribution). The KL penalty serves to keep the diffusion policy from diverging too much from the base diffusion model; however, staying close to the original distribution was not a concern in this case.

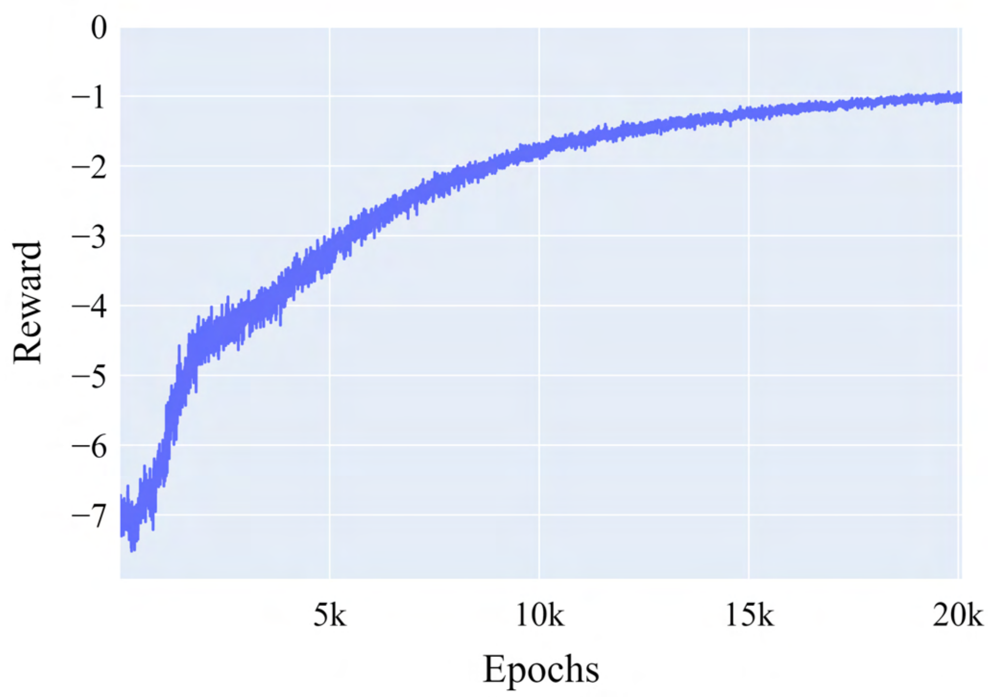
For the value function, a custom convolutional neural network with a max-pooling layer and a fully connected output layer was utilized. The value function was initialized from a classifier trained on all MNIST digits. The detailed model architecture can be found in Appendix B.

During rollout, only 25 out of 1000 timesteps were used for sampling, to speed up the optimization process. Reducing the number of timesteps represents a trade-off between model fidelity and optimization efficiency. In testing, 25 timesteps were sufficient for generating readable and diverse digits. It was conjectured that the optimization would generalize to the other timesteps as well since all denoising steps utilized the same U-Net.

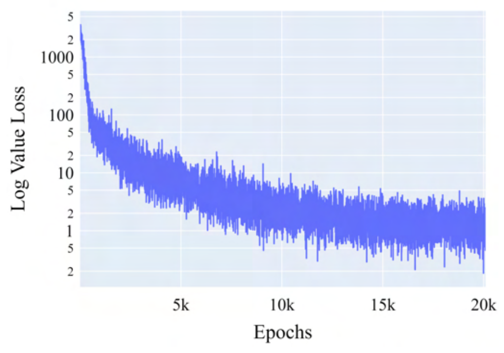
Both the policy and value function’s learning rates were set at a very conservative level: 1×10^{-7} and 1×10^{-5} respectively. This is much lower than the learning rate used to train the base model initially, which was 1×10^{-4}. Preliminary testing indicated that the diffusion model could deteriorate rapidly if the learning rate was set too high, resulting in the generation of noise which did not resemble the digit “4”. While this experiment was not concerned with retaining the original data distribution, it was not clear if the optimization process would be capable of aligning the policy via reward signals if the generated samples deviated too far from the target image.

Refer to Table 6.2 for a comprehensive overview of the experiment's hyperparameters.

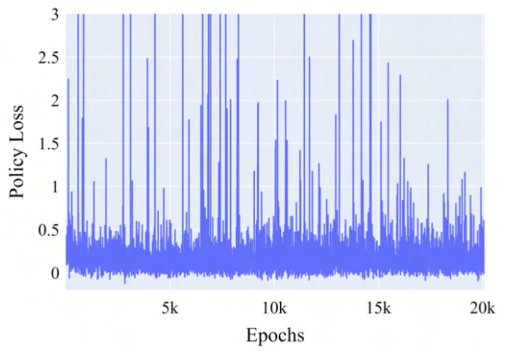
6.4.2. Results



(a) Reward



(b) Log value loss



(c) Policy loss

Figure 6.3.: Training graphs showing mean rollout reward and policy and value losses per epoch for the *Target Image* experiment.

The training graphs, presented in Figure 6.3, show that the reward monotonically increased during training, while the value loss decreased and policy loss remained spiky without any clear decreasing trend. While the reward continued to increase until the end of training, there was little visual change to the samples beyond epoch 3000. Hence, a checkpoint for epoch 3000 serves as the "optimized" model.

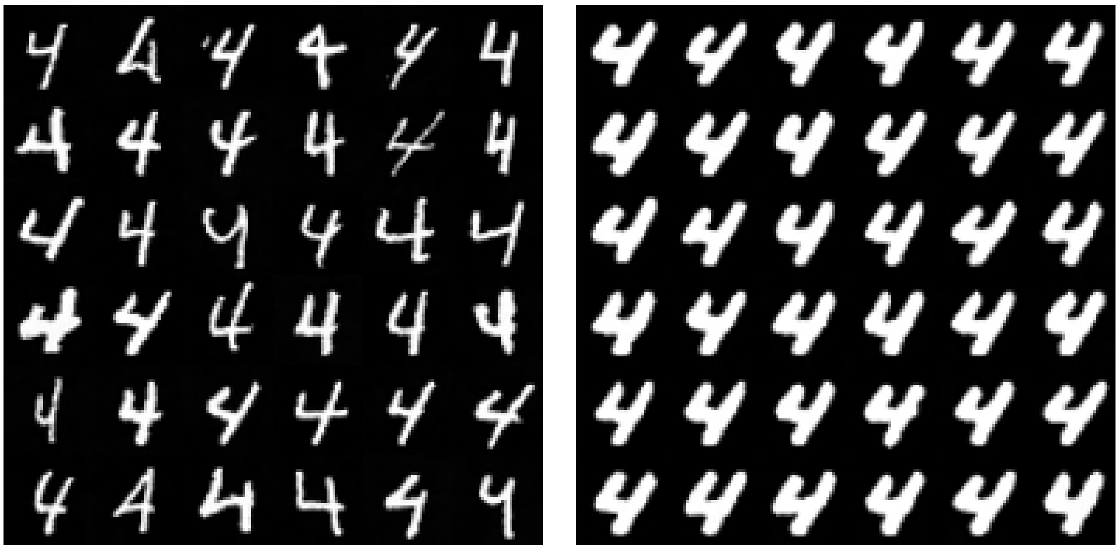

(a) Base model (b) Optimized model

Figure 6.4.: Samples from the base model and the optimized model for the *Target Image* experiment. The samples are sampled using DDPM with 250 timesteps. The two grids are sampled using the same conditions, starting noise, and seed.

Samples from the base model and the optimized model are shown in Figure 6.4. The optimized model generates digits resembling the target image.

The optimization progression, illustrated in Figure 6.5, demonstrates that as training proceeded, the generated images became progressively closer to the target image. This indicates that the reward function effectively guided the optimization process. Furthermore, samples that initially resembled the target image smoothly transitioned to the target image (e.g., row 1). Other samples transitioned through an intermediate state where the digit appeared as a white "blob" before converging on the target image (e.g., row 2).

The reward distributions for samples from the base model and optimized model, presented in Figure 6.6, show a mean reward increase of around 5 standard deviations and significantly lower reward variance.

6.4.3. Evaluation

The samples from the optimized model demonstrate that the experiment was successful in aligning the policy with the target image. The optimized samples are nearly identical to the target image, indicating significant overfitting, which was the desired outcome in this specific experiment.

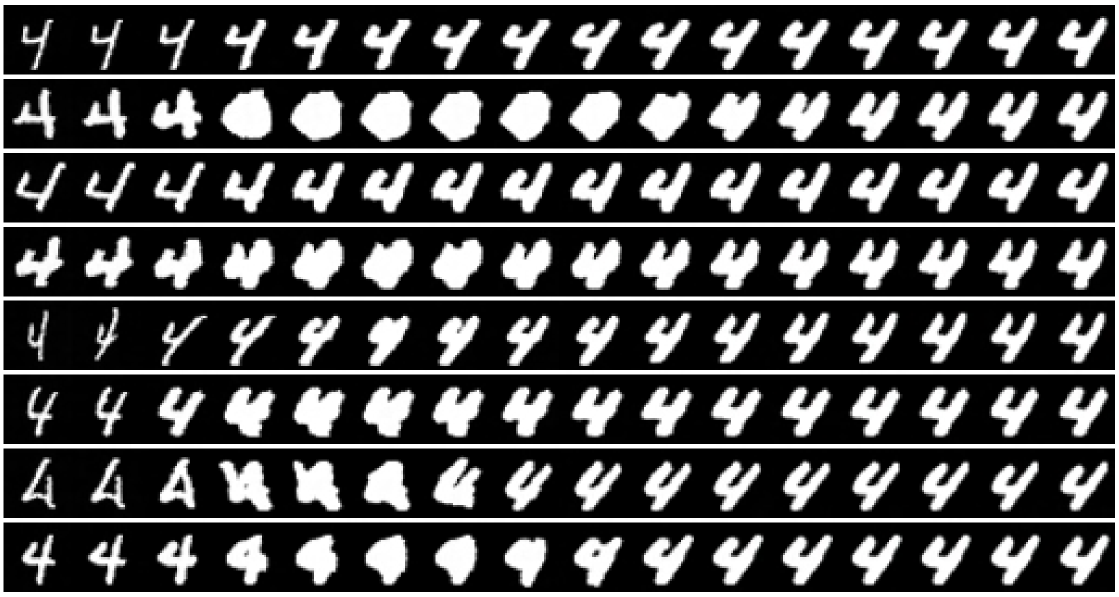


Figure 6.5.: Optimization progression for the *Target Image* experiment. Starting from the base model (left) and ending at the optimized model (right).

Tracking the optimization progression over time provided further insights. The observation that some samples transition through a degraded intermediate stage indicates that the starting distribution might not be critically important for this optimization task to work. For instance, starting with an untrained diffusion model or a model trained on all MNIST digits would likely result in slower convergence but ultimately achieve similar results.

Additionally, examining the reward distributions of the base and optimized models revealed noteworthy differences consistent with what one would expect when converging on a Dirac delta distribution.

However, despite its success, the optimization process required a substantial number of epochs to complete. While this was not a significant issue with the small model used in this experiment, the problem of longer sampling times would become more prominent when scaling up to larger and more complex models. Thus, finding ways to increase the convergence rate is a crucial goal for subsequent experiments. Raising the learning rate emerges as the primary target to accelerate this process, although such adjustments need to be made carefully to avoid rapid model degradation, as noted based on our preliminary testing.

Overall, the experiment demonstrated that our methodology for adapting diffusion models to a reinforcement learning environment was effective in a very restricted setting. The results validated our implementation of PPO and provided insights into the training scheme's hyperparameter values, notably that it was viable to optimize using few rollout timesteps.

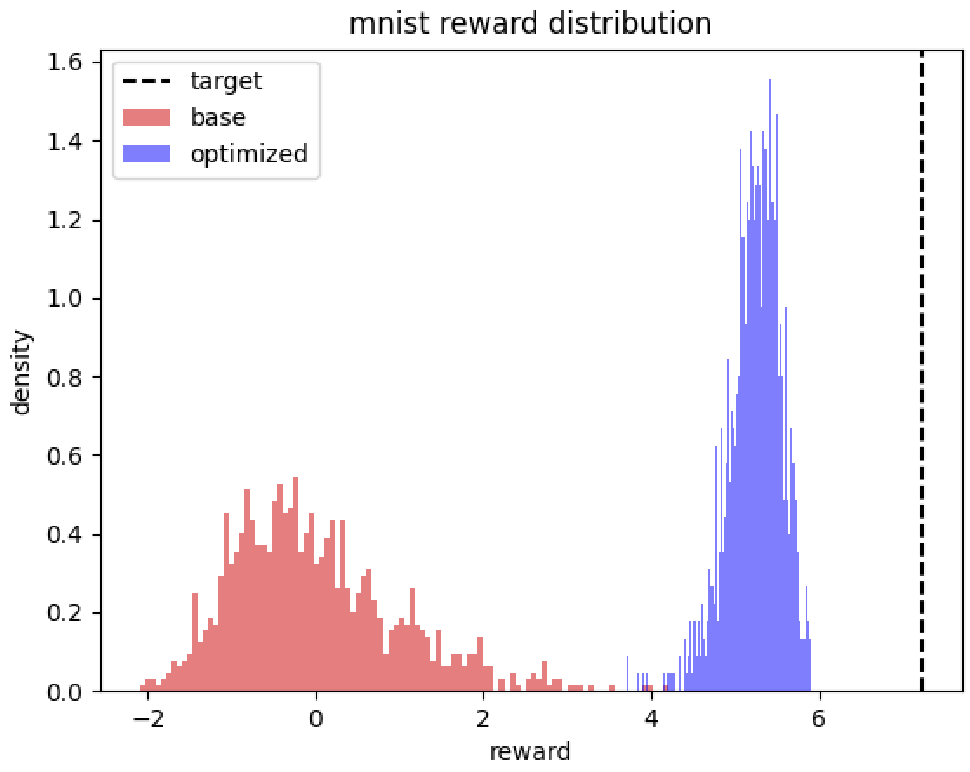
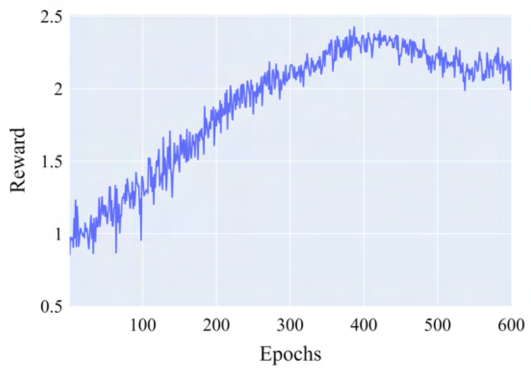
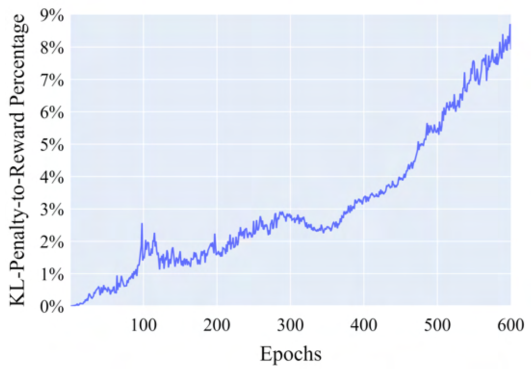


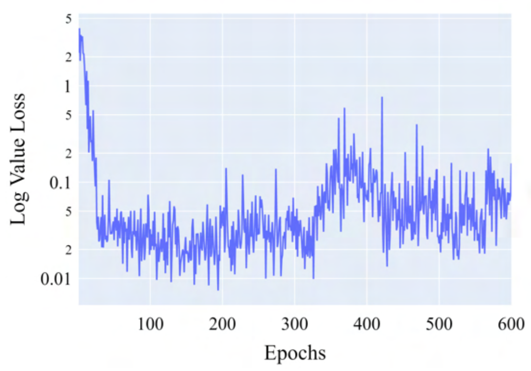
Figure 6.6.: Reward distributions for samples from the base model and the optimized model for the *Target Image* experiment. Each distribution is based on 1024 samples evaluated by the reward function. The distributions were standardized based on the mean and std of the base distribution. The samples are sampled using Denoising Diffusion Probabilistic Model (DDPM) with 25 timesteps and without classifier guidance. The samples for the two distributions are sampled using the same conditions, starting noise, and seed. The target line represents the theoretical maximum standardized reward (0 error).



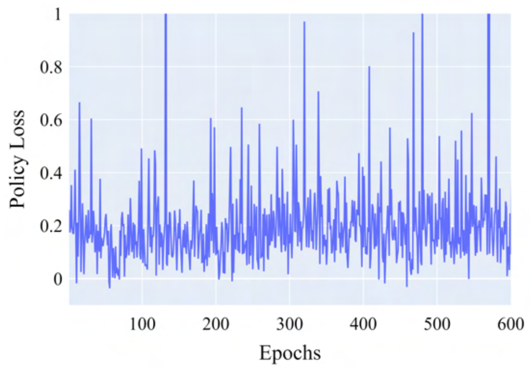
(a) Reward



(b) KL-penalty-to-reward percentage



(c) Log value loss



(d) Policy loss

Figure 6.7.: Training graphs from policy optimization for the *Colorfulness* experiment.

6.5. Experiment 2: Colorfulness

The second experiment, dubbed *Colorfulness*, moderately increased in complexity over the first experiment by utilizing a larger diffusion model with a distribution that was more representative of the text-to-image models that were being worked towards and a reward model that was more representative of visual quality akin to a human preference.

This experiment also deviated from the methodology presented earlier as the reward model remained fixed to a function. Only *Step 4* of the methodology was carried out and tested. The primary motivation behind this experiment was to prove that the optimization process could align a diffusion model with the capacity to generate diverse images, using a reward model that was more nuanced. A secondary motivation was to prevent the distribution collapse observed in Experiment 1: *Target Image* and to increase optimization speed.

Unlike in Experiment 1: *Target Image*, where the objective was to collapse the distribution around a specific target image, the intention in this experiment was to maintain a certain diversity in the output while aligning the samples towards a particular attribute that would still be easily observable. To this end, a reward function promoting colorfulness was chosen.

6.5.1. Experimental Setup

The base policy used in this experiment was a class-conditional 64×64 ImageNet-1k model[2] trained and released by OpenAI. This model represented a significant increase in complexity compared to the diffusion model used in Experiment 1: *Target Image*, as the resolution is twice as high in the width and height dimensions and the ImageNet-1k dataset it is trained on is much more diverse than MNIST (see subsection 2.11.2 for more information on ImageNet-1k).

To optimize the policy toward increased colorfulness, a fixed reward function was used. The reward function was based on a reward introduced by Pinto et al. (2023), which was designed to promote color vividness and diversity. Since the original paper only provided a brief textual description of the reward function, a custom implementation had to be written; the implementation is provided in Appendix A.

For the value function, the 64×64 ImageNet-1k classifier[3] trained and released by OpenAI, was repurposed. This classifier was originally meant to be used with the base policy for classifier guidance (see subsubsection 2.5.7). To adapt it for use as a value function, a fully connected layer was added at the end of the classifier to produce a single output scalar. Since the reward function was class-agnostic, the value function did not take the class condition as additional input. However, it was provided with the diffusion timestep associated with the state to be evaluated, providing more context about how far along the reverse diffusion process the state was.

During rollout, an ImageNet-1k class label was sampled at random per environment to condition the trajectory on. These class labels serve as the equivalent of a "prompt", as referred to in chapter 5.

To prevent distribution collapse and ensure model fidelity, a KL penalty was introduced (see *KL Regularization* in section 5.2). The KL penalty coefficient was set to 0.02, following the work of Ouyang et al. (2022). Unlike a reward model that uses the same base model as the policy, such as the one proposed in section 5.2, the reward function in this experiment did not have any knowledge about the underlying distribution of the policy. Consequently, there was a higher risk of policy divergence, as the reward model might guide the policy toward producing unrealistic colors or image features. Therefore, the KL penalty was necessary to keep the policy close to the original distribution.

[2] Available at: `https://openaipublic.blob.core.windows.net/diffusion/jul-2021/64x64_diffusion.pt`

[3] Available at: `https://openaipublic.blob.core.windows.net/diffusion/jul-2021/64x64_classifier.pt`

The number of rollout timesteps was increased to 32, from 25 in Experiment 1: *Target Image*, to account for the increased complexity of the diffusion model. Additional timesteps were deemed necessary to generate samples with sufficient quality. However, the number of timesteps was still kept relatively low, which speeds up optimization.

The learning rate was increased by one order of magnitude to accelerate optimization. The learning rate was set to 1×10^{-6}, up from 1×10^{-7} in Experiment 1: *Target Image*. As noted previously, the prior learning rate was set conservatively and necessitated a significant number of optimization epochs. However, after rectifying some bugs from the initial versions of our code, training became stable at this increased learning rate. Furthermore, the larger number of rollout timesteps resulted in a larger optimization batch, which could justify the increased learning rate.

The policy warmup period was increased compared to Experiment 1: *Target Image*, from 4096 to 8192 optimization steps. This adjustment was made to accommodate the more intricate reward function. By providing a longer warmup period, the value function had more time to learn the complexities of the reward scheme before guiding the policy optimization.

For a full overview of hyperparameters, refer to Table 6.2.

6.5.2. Results

The training graphs in Figure 6.7 show that the reward steadily increased during policy optimization but started decreasing after around epoch 400. Therefore, a checkpoint saved at epoch 400 was selected as the final "optimized" model. The mean reward after optimization was more than twice as high as before.

The value loss decreased for the first portion of training, around 200 epochs, then slowly increased before experiencing a larger increase after epoch 250.

The policy loss remained relatively constant and spiky throughout training, similar to Experiment 1: *Target Image*.

The KL penalty increased at an increasing rate throughout training. It grew faster than the colorfulness reward and constituted around 3.3 % of the total reward at epoch 400.

Samples from the base policy and the optimized model are shown in Figure 6.8. The optimized model generated images that were more colorful compared to the base model. A qualitative analysis of the images showed that the images also appear to have increased contrast and sharpness, more high-frequency information, such as lines and distinct shapes, and less blur, specifically in the backgrounds. The optimization progression, illustrated in Figure 6.9, demonstrates that the images became progressively more colorful over time, and that the image content is altered during the process.

The reward distributions in Figure 6.10 show that the mean reward of the optimized model increased by around 2 standard deviations, with a lower variance compared to the base model.

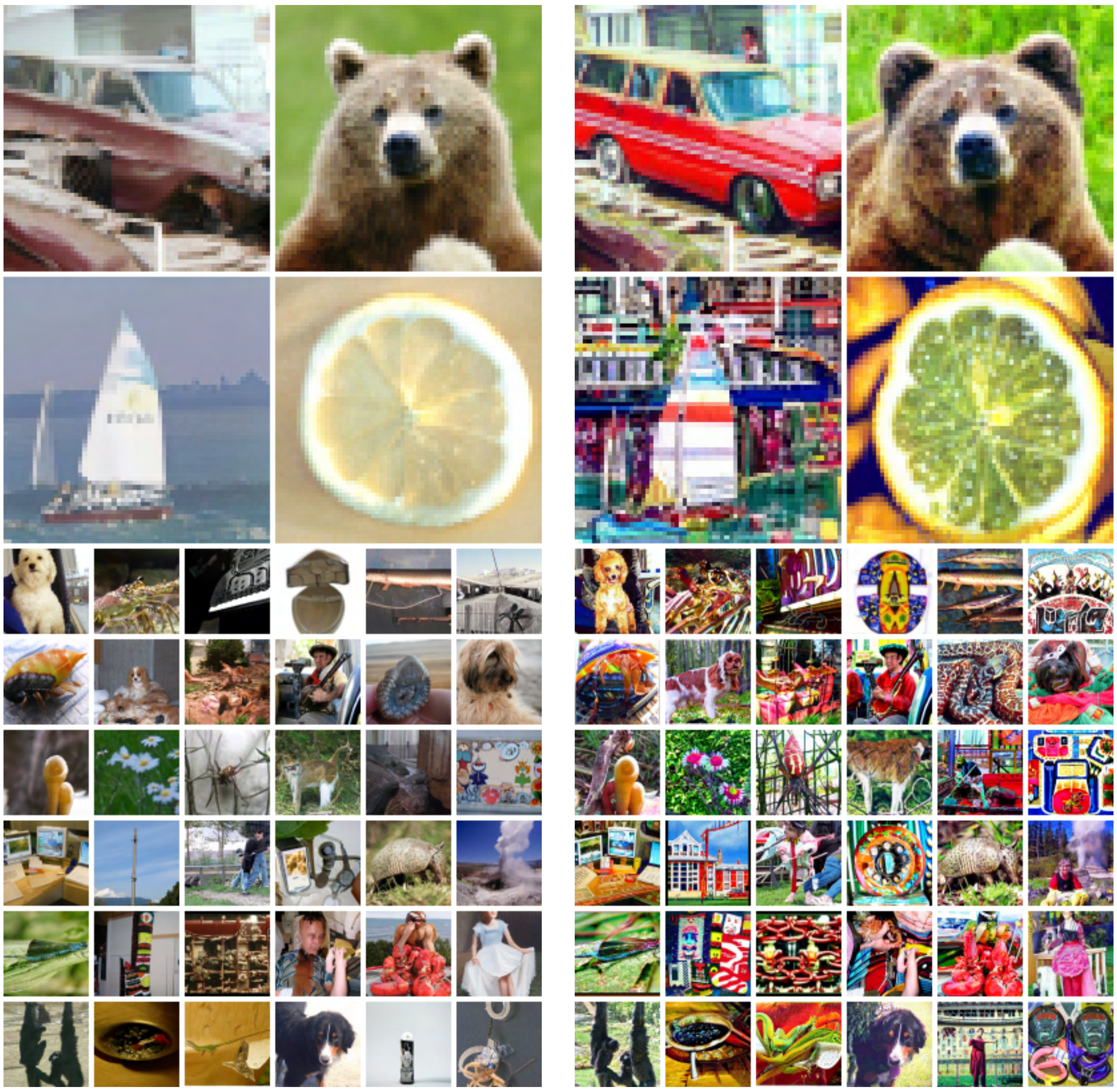
6.5.3. Evaluation

The samples and reward distributions clearly demonstrate that the policy was successfully aligned with the colorfulness reward function. The samples show an increase in color vividness and diversity, which aligns with the objective of the experiment. However, the optimization process resulted in samples with more unrealistic and oversaturated colors compared to those generated by the base model. This is unsurprising, considering that the reward function specifically encourages this, but it does indicate that the optimization process could have been cut prior to epoch 400. This overexaggeration of the effects of the reward model, though it led to less naturalistic images, can be considered a positive aspect of the experiment as it highlighted the effects of optimization more distinctly, facilitating a more straightforward evaluation of the optimization process.

The change in image content observed during optimization could be a byproduct of model degradation, but can potentially be explained by the samples shifting toward more colorful regions of the original data distribution. It can also be attributed to the inherent nature of diffusion models, which are sensitive to small variations in model weights or starting noise. The recursive reverse diffusion process amplifies the effects of small changes, resulting in more pronounced changes in the final image samples. Optimization for fewer epochs or an increase in the KL penalty coefficient could potentially mitigate this effect.

The increase in value loss after epoch 250 is likely related to the fact that the KL penalty played an increasingly significant role in the total reward provided to the policy throughout training. The value function spent the first portion of training learning the behavior of the reward function, when the KL penalty was low, and struggled to adapt when the KL penalty started to increase. As a result, it was overfitting to the initial reward and was unable to effectively model the transition to a higher penalty regime. Consequently, this created a divergence between the model's expected rewards and the actual rewards obtained, leading to a marked increase in value loss after epoch 250. Future experiments should favor a setup that leads to a more gradual increase in the KL penalty, or using methods such as adaptive KL control (Ziegler et al., 2019), to allow the model to better adapt to changing reward structures over time.

In conclusion, this experiment successfully aligned the policy with the colorfulness reward model while maintaining a certain level of diversity in the generated images. The results provided insights into the optimization process, especially the challenge of balancing reward vs penalty, and demonstrated the effectiveness of the methodology. However, more work is needed to address the observed reduction in image fidelity.



(a) Base model (b) Optimized model

Figure 6.8.: Samples from the base model and the optimized model for the *Colorfulness* experiment. The images are sampled using DDPM with 250 timesteps and without classifier guidance. The two grids are sampled using the same random conditions, starting noise, and seed.

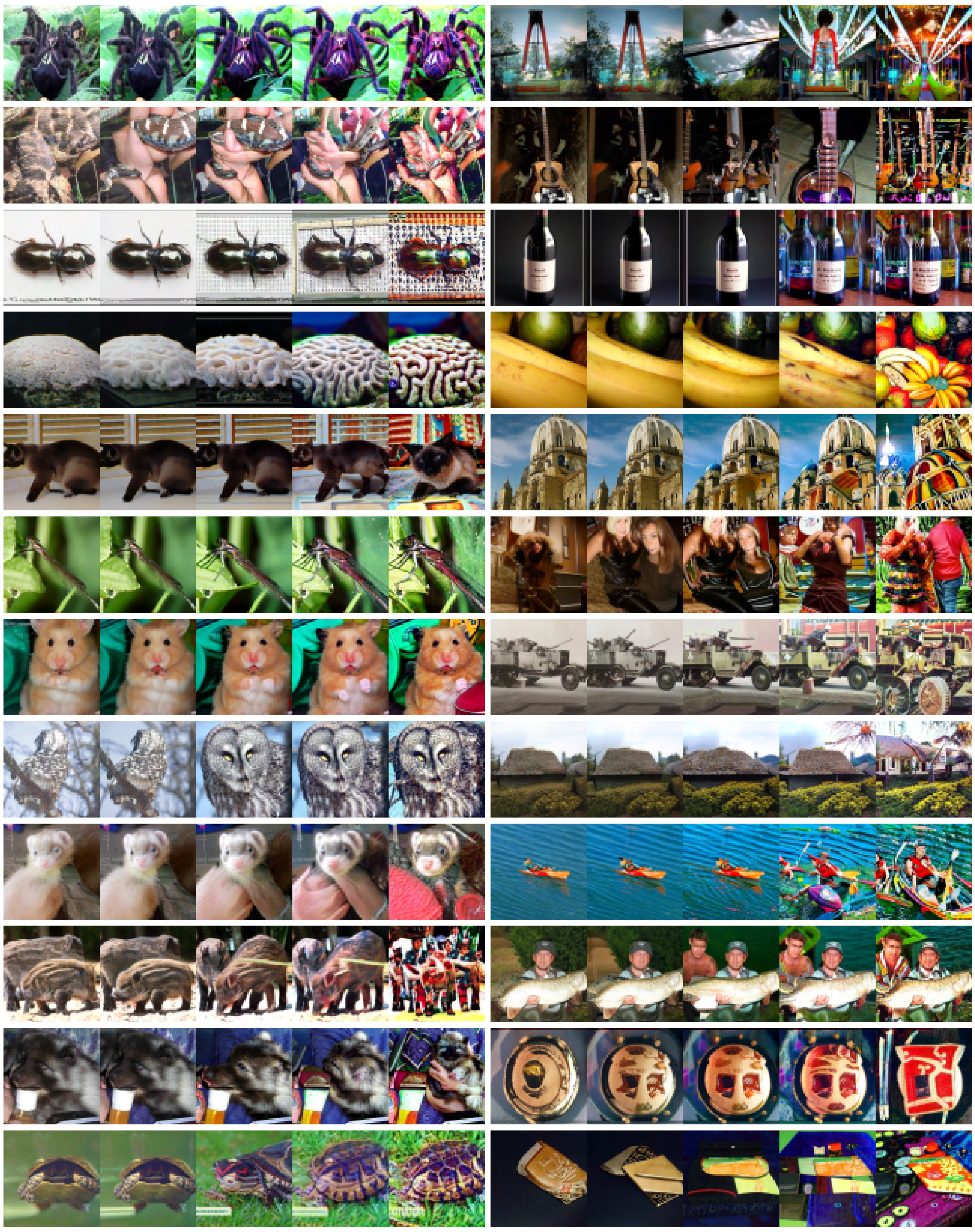


Figure 6.9.: Optimization progression for the *Colorfulness* experiment. Each column starts with the base model (leftmost) and ends at the optimized model (rightmost). The images are sampled using DDPM with 250 timesteps and without classifier guidance. The images in a progression-row are sampled using the same random conditions, starting noise, and seed.

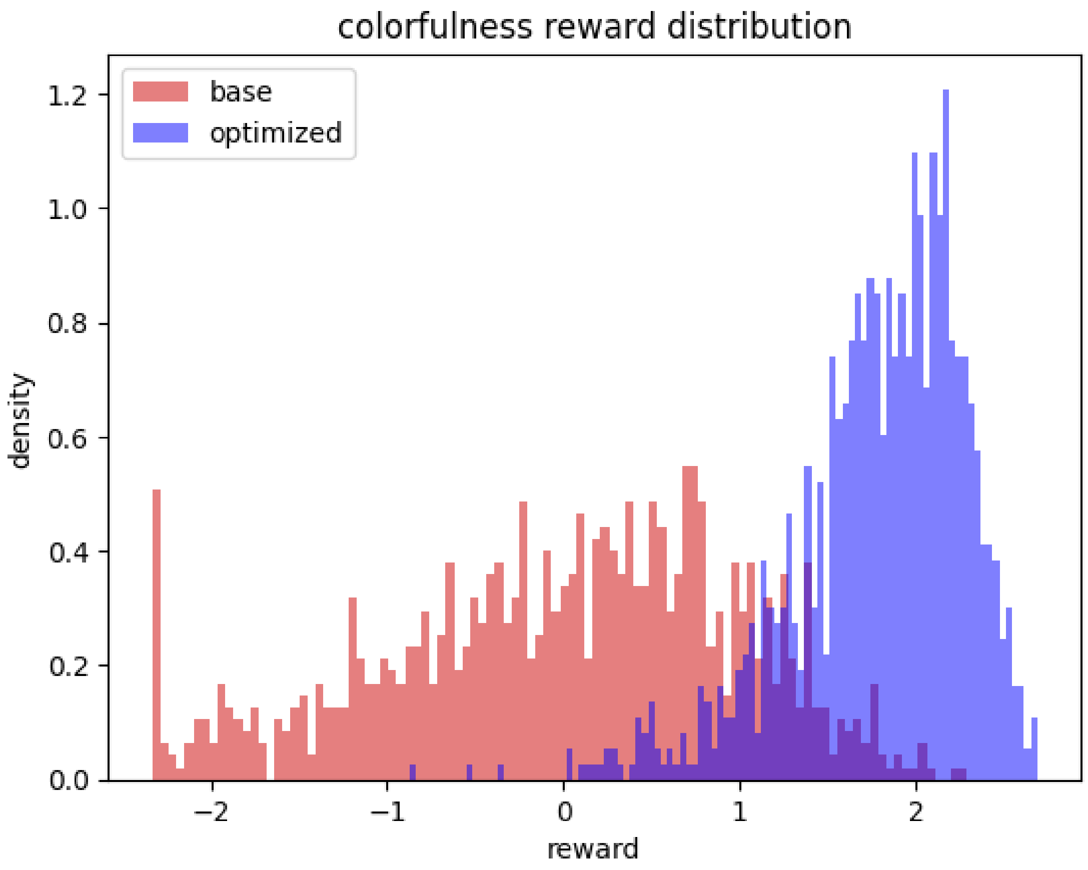
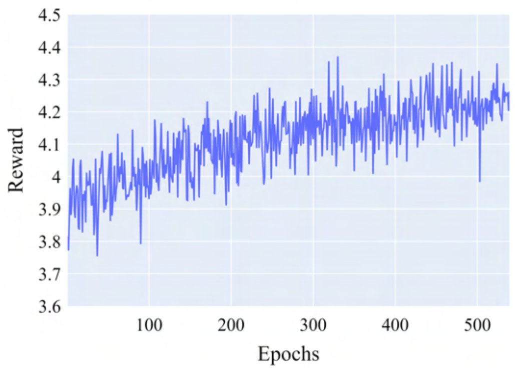
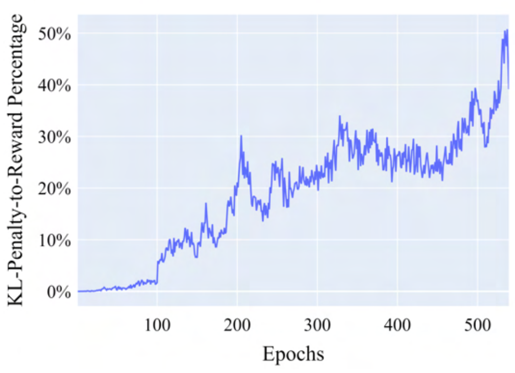


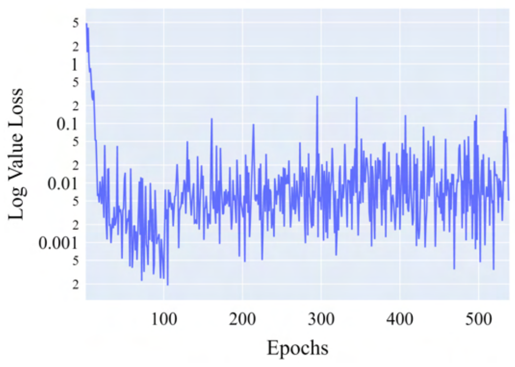
Figure 6.10.: Reward distribution for samples from the base model and the optimized model for the *Colorfulness* experiment. Each distribution is based on 1024 samples evaluated by the reward function. The samples are sampled using DDPM with 25 timesteps and without classifier guidance. The samples for the two distributions are sampled using the same conditions, starting noise, and seed.



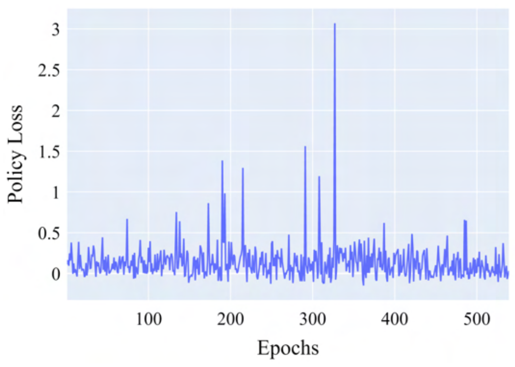
(a) Reward



(b) KL-penalty-to-reward percentage



(c) Log value loss



(d) Policy loss

Figure 6.11.: Training graphs from policy optimization for the *Aesthetic Score* experiment.

Section 2: Aligning Simple Diffusion Models Using Human Feedback

6.6. Experiment 3: Aesthetic Score

The third experiment, dubbed *Aesthetic Score*, aimed to align a simple diffusion model with a reward model trained using human feedback. This experiment deviated from the methodology presented earlier as it involved utilizing an existing reward model instead of training our own. The primary objective was to validate that our methodology can effectively align diffusion models with reward models based on human preferences.

6.6.1. Experimental Setup

For this experiment, we utilized the same base policy model and value function as in Experiment 2: *Colorfulness*. However, instead of using a fixed reward function like in the two previous experiments, we incorporated the LAION Aesthetics Predictor V2[4]. The LAION Aesthetics Predictor V2 is a reward model that has been trained using human feedback and provides aesthetic scores based on human ranking from 1 to 10 on a per-image basis.

It should be noted that the Aesthetics Predictor was not trained on such low-resolution images, like those generated by our chosen base policy model (i.e., 64×64). Therefore, there might be a discrepancy between the images generated by the policy during rollout and the image distribution the reward model is familiar with evaluating. To compensate for this potential mismatch, we increase the KL penalty coefficient slightly, to 0.03. By increasing the KL penalty, we encourage the policy optimization process to remain closer to the original distribution and preserve its characteristics.

We also modify some hyperparameters to improve image quality to further compensate for the low-resolution images. We increase the number of diffusion timesteps to 50, allowing for higher-quality samples despite the lower resolution. Additionally, we slightly decrease the learning rate of the policy to 5×10^{-7} (with a learning rate of 2×10^{-7} for the first 100 epochs) in order to achieve better training stability and maintain fidelity closer to the original data distribution for longer.

Full details of the experimental setup, including all relevant hyperparameters, can be found in Table 6.2.

6.6.2. Results

Training graphs are shown in Figure 6.11. During training, the reward gradually increased over time. However, there was no significant increase in reward observed throughout the entire optimization process.

The KL-penalty-to-reward percentage experienced a major jump at epoch 100 and continued to climb until it reached approximately 50 % during training, before dropping below 40 % for the fully optimized policy.

The log value loss rapidly decreased to around 0.001 within the first 100 epochs of training. Afterward, the loss showed an upward trend, stabilizing at around 0.01 for the remainder of the training process.

The policy loss exhibited no clear pattern or trend, with occasional spikes occurring intermittently throughout training.

Samples from both the base model and the optimized model are shown in Figure 6.12. The optimized model's samples generally exhibit a white background, cropping out everything except for the subject. Furthermore, compared to the base model, these images tend to have increased brightness and lighter colors.

[4]Available at: `https://github.com/christophschuhmann/improved-aesthetic-predictor`.

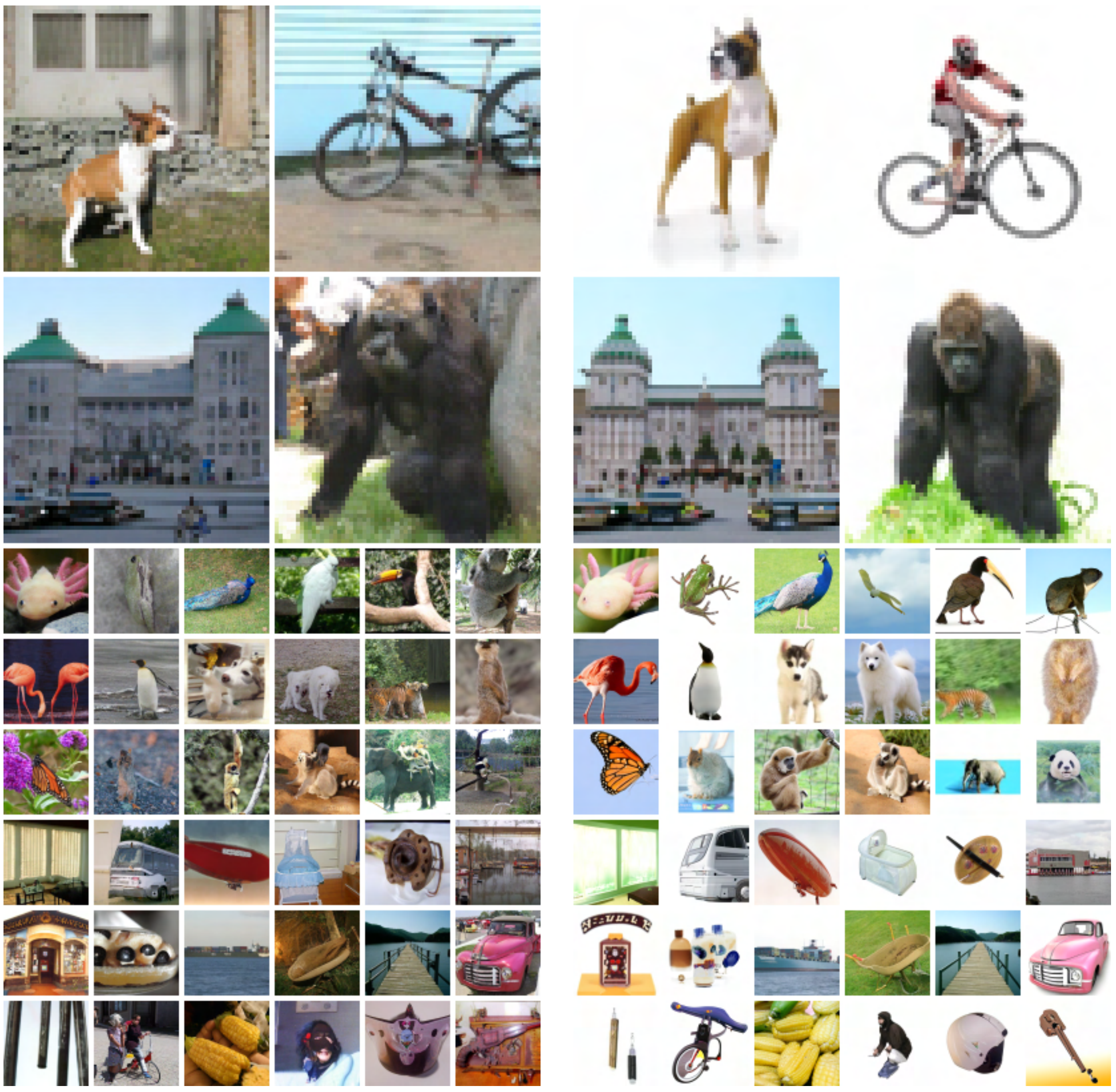
Analyzing the reward distributions depicted in Figure 6.14, we observe that although there is a slight shift toward higher rewards in the optimized model compared to the base model, the overall improvement is minimal.

6.6.3. Evaluation

The experimental results suggest that the alignment between the diffusion model and the Aesthetics Predictor reward model was moderately successful in increasing the reward. The noticeable increase in the KL-penalty-to-reward percentage indicates that the policy deviated further away from the original distribution as training progressed, resulting in large penalization due to larger divergence. This deviation may have hindered better alignment with the Aesthetics Predictor reward model.

The stark shift toward a white background and lighter colors in the generated images is likely related to the preferences captured by the reward model. Since we did not train this model ourselves, it is difficult to evaluate if the resulting images align with what human evaluators would deem preferrable, without conducting a human evaluation trial. Even so, we conjecture that the reason for favoring such types of images could be related to the fact that the Aesthetics Predictor is in part trained on a dataset of 15,000 logo image-text pairs with aesthetic ratings from 1 to 10, which typically exhibit characteristics such as small square images, with white backgrounds, and a distinct subject in focus.

Altogether, this experiment highlights some challenges related to aligning diffusion models with complex reward models based on human preferences. The discrepancy between the input images' resolution and the trained reward model's expectations may pose an additional constraint. To achieve better alignment in future experiments, it may be necessary to employ lower learning rates or slightly reduce the KL penalty coefficient to prevent over-preservation of the original distribution.



(a) Base model (b) Optimized model

Figure 6.12.: Samples from the base model and the optimized model for the *Aesthetic Score* experiment. The samples are sampled using DDPM with 250 timesteps and without classifier guidance. The two grids are sampled using the same conditions, starting noise, and seed.

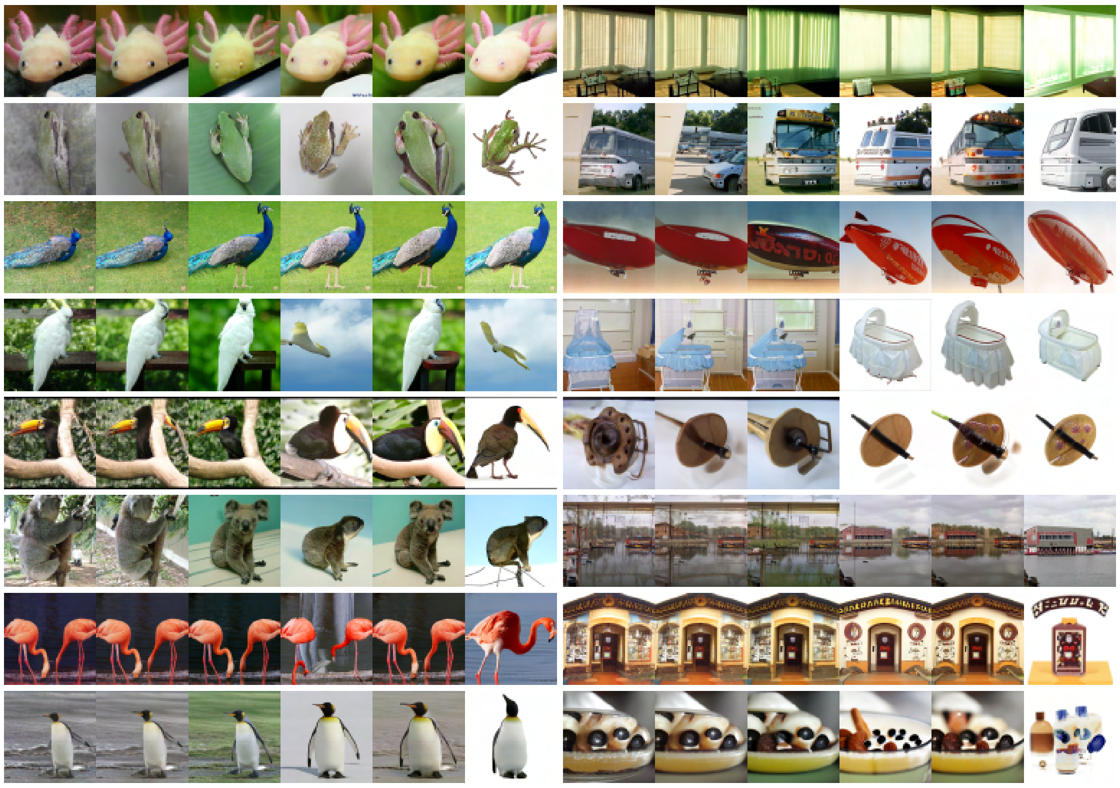


Figure 6.13.: Optimization progression for the *Aesthetic Score* experiment. Each column starts with the base model (leftmost) and ends at the optimized model (rightmost). The images are sampled using DDPM with 250 timesteps and without classifier guidance. The images in a progression-row are sampled using the same random conditions, starting noise, and seed.

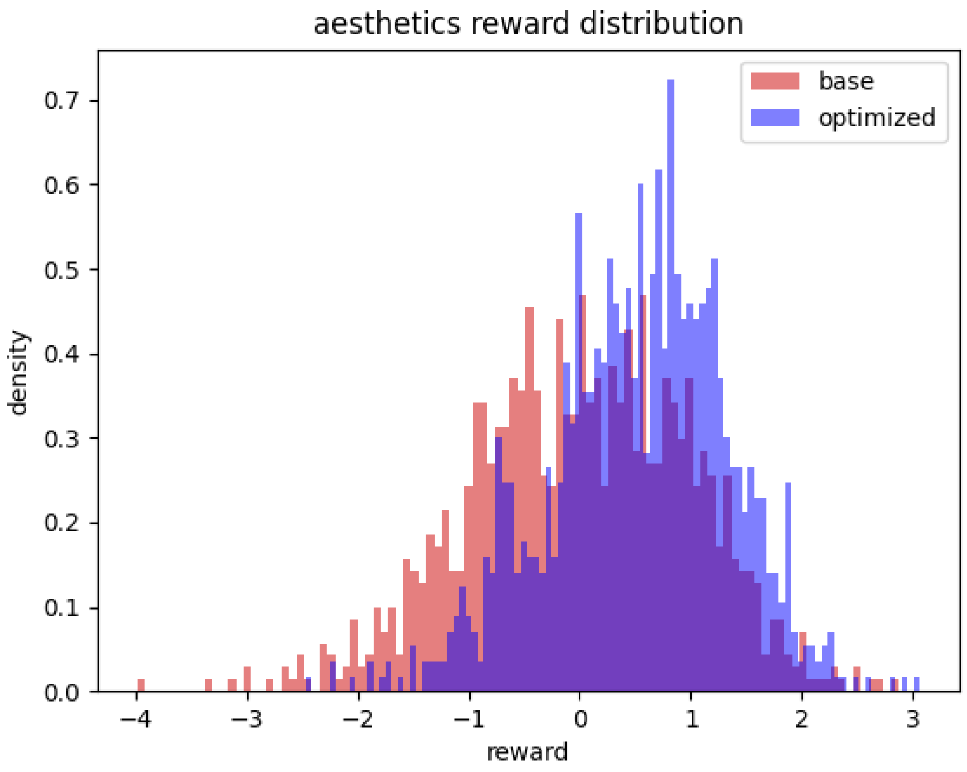


Figure 6.14.: Reward distribution for samples from the base model and the optimized model for the *Aesthetic Score* experiment. Each distribution is based on 1024 samples evaluated by the *aesthetics* reward function. The images are sampled using DDPM with 25 timesteps and without classifier guidance. The samples for the two distributions are sampled using the same conditions, starting noise, and seed.

6.7. Experiment 4: Macaw

The fourth experiment, dubbed *Macaw*, serves as the main implementation of the methodology presented in chapter 4 and chapter 5. The experiment uses the same base policy as Experiment 2: *Colorfulness* and Experiment 3: *Aesthetic Score*, but increases the complexity from previous experiments by training a custom reward model on human preferences. The primary motivation behind this experiment was then to prove that the methodology is capable of capturing human preferences in a reward model and aligning a class-conditional diffusion model to those preferences.

6.7.1. Experimental Setup

The experiment utilizes the same base policy and value function as in Experiment 2: *Colorfulness* and Experiment 3: *Aesthetic Score*. To simplify preference collection and reward model training, the experiment was limited to a single class: macaw (88). The remaining experimental setup is presented according to the four steps introduced in the high-level methodology in section 1.3.

Refer to Table 6.2 for a comprehensive overview of the experiment's hyperparameters.

Step 1: Curating a Labeling Dataset. The labeling dataset curated for this experiment consisted of 1000 images of macaws, split into 200 tasks for a task size of $K = 5$. The low task size kept the labeling efficiency high without affecting in-task diversity. The images were sampled at 25 diffusion timesteps using the DDPM sampler, without classifier guidance.

Step 2: Collecting Human Feedback. For this experiment, a single human labeler (one of the authors) was used. We theorized that it would be easier to model the preferences of a single labeler, compared to the potentially diverging preferences of multiple labelers, and that we as a result could get by with collecting less data to achieve a similar, or potentially even better, result.

The 200 tasks were labeled using the labeling software and methodology presented in chapter 4. Every image in a task was compared against every other image in the task. As a result, a total of 2000 comparisons were made. The instructions for the preference collection can be seen in Table 6.3.

Step 3: Training a Reward Model. The reward model was based on OpenAI's pre-trained ImageNet classifier, an accompanying model originally used for classifier guidance with the base diffusion model. To produce scalar rewards, a fully-connected layer was concatenated to the classifier.

Num.	Instruction
1	Always prefer images with a macaw over images without a macaw.
2	Of the images with a macaw, prefer the image with the highest perceived fidelity.
3	Rank the images in order of what you would most prefer as output from the model.

Table 6.3.: Instructions for the *Macaw* preference collection.

Preliminary testing of the reward model implementation involved training the model on 80 % of the data from the collected human preferences, and then evaluating it using the remaining 20 % of the data. After validating the reward model implementation, the final reward model used for policy optimization was trained on all of the collected human preferences. Both reward models were trained with a batch size of 8 tasks (constituting 40 images), using a learning rate of 1×10^{-4} for 1000 epochs. The reward model outputs were not standardized.

Step 4: Optimizing Policy with Reinforcement Learning. Similar to prior experiments, the policy optimization of the diffusion model used 32 environments. Following the methodology, the images generated during rollout used the same settings as when curating the labeling dataset, for a total of 800 reinforcement learning states (32 images $\times$ 25 diffusion timesteps) per epoch. The states were randomly split into 32 minibatches (of size 25) for policy optimization.

The policy model was trained using a learning rate of 1×10^{-6}, half a magnitude higher than Experiment 3: *Aesthetic Score* and similar to Experiment 2: *Colorfulness*. The linear warmup schedule of the policy was 8192 minibatch steps, while the value function used a learning rate of 1×10^{-5} and a linear warmup schedule of 1024 minibatch steps, all identical to that of the two previous experiments.

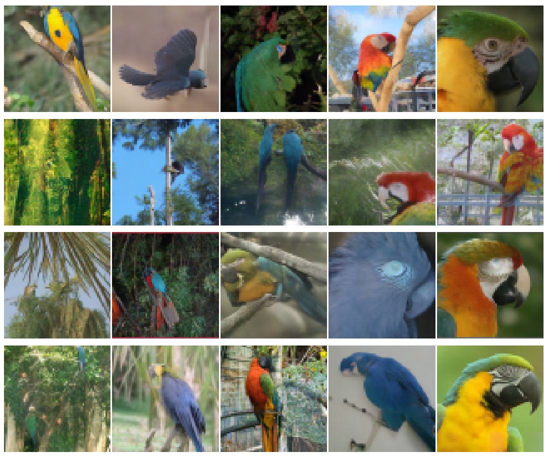
Following an aggressive distribution preservation in Experiment 3: *Aesthetic Score*, the KL penalty coefficient was reduced to 0.02, to match Wu et al. (2021).

6.7.2. Results

The results of the experiment are divided into the sections “Human Preference Collection”, “Reward Model Training” and “Policy Optimization”, for clarity.

Human Preference Collection

Four example tasks preference ordered by the human labeler are shown in Figure 6.15. The preference ratings match the preference feedback provided by the human labeler after the experiment in Table 6.4.


(a) Human labeler

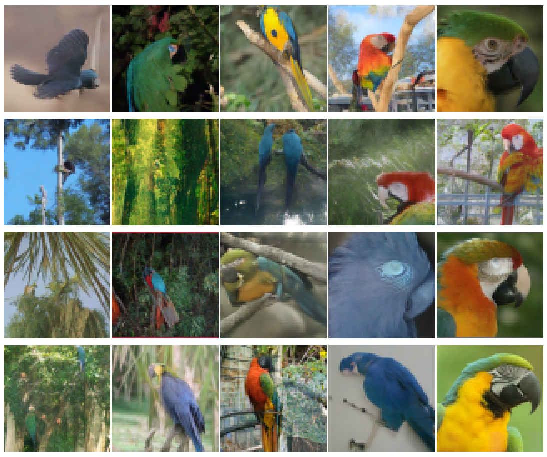

(b) Reward model

Figure 6.15.: Four example tasks (one per row) of five images each, preference-rated by (a) the human labeler, and (b) the reward model for the *Macaw* experiment. The images within a task are ordered from least preferred (leftmost) to most preferred (rightmost).

Num.	**Preference Feedback**
1	Preferred images with photogenic macaws.
2	Preferred images with colorful macaws.
3	Preferred images with macaws close to the “camera”.
4	Preferred images with a single macaw.
5	Preferred images with bokeh.
6	Did not prefer images without macaws.

Table 6.4.: Feedback from the human labeler when asked about their *Macaw* labeling preferences after the experiment.

Reward Model Training

Reward model training statistics for both the preliminary model and the final model can be seen in Figure 6.16. The training loss of both models reach near-zero values after a few epochs. The validation loss of the preliminary model reaches its lowest point by epoch 10, after which it steadily climbs. Both reward models reach a training accuracy of 100 % throughout training. The validation accuracy of the preliminary reward model stabilizes at around 80 % to 83 % after a few epochs.

Four example tasks ordered by the trained reward model are shown in Figure 6.15. The reward model orders the most preferred images similar to the human labeler, with slight deviations for the less preferred images in some of the tasks.

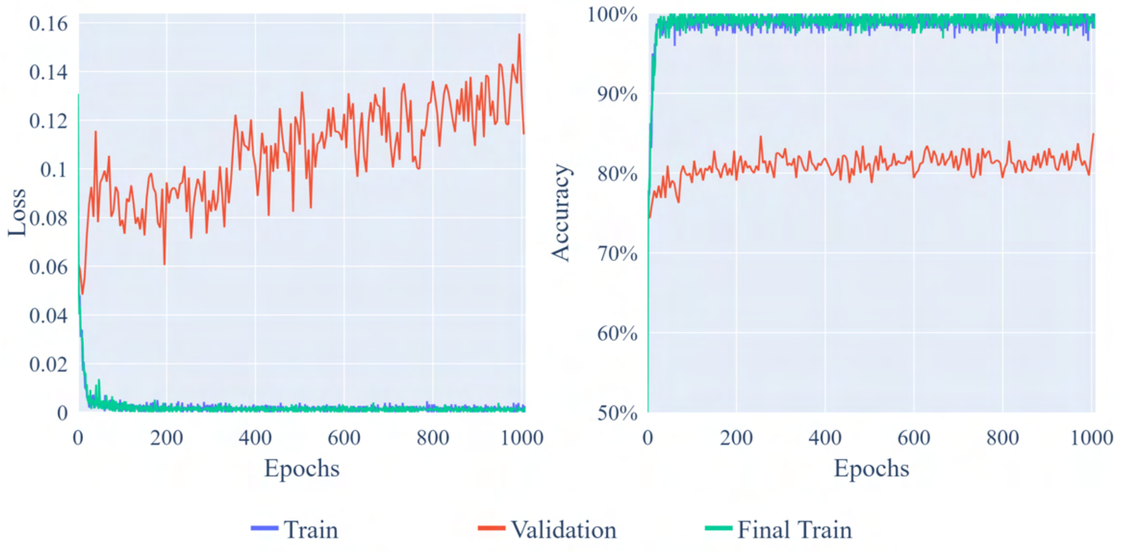


Figure 6.16.: Loss and accuracy graphs for the preliminary and final reward models for the *Macaw* experiment. Training statistics for both reward models are shown, but only the preliminary reward model has validation statistics. The y-axis in the accuracy graph starts at 50 %.

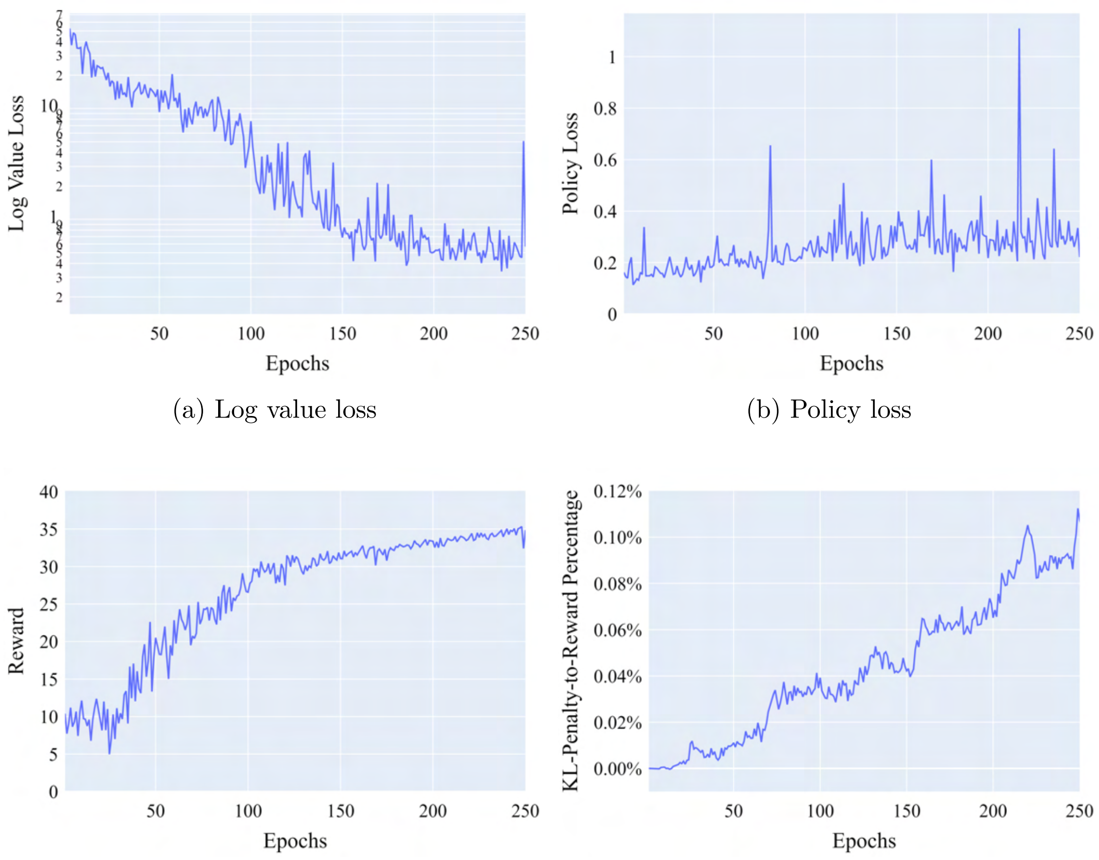
.

Policy Optimization

Training statistics from policy optimization of the diffusion model can be seen in Figure 6.17. The log value loss steadily decreases throughout policy optimization, with a spike at epoch 250. The policy loss is spikey with a slight upwards trend. The reward monotonically increases from epoch 25, and reaches a mean reward that is 3.5 times as high compared to that of the base policy at the end of training. The KL-penalty-to-reward percentage stays beneath 0.12 % throughout the duration of training. The final "optimized" model uses a checkpoint saved at epoch 250.

Samples from the base policy and the optimized model can be seen in Figure 6.18. Compared with the base policy, the optimized model prefers centered, close-up images of the macaw. A qualitative analysis of the images show that the images also appear to have increased contrast and a simplistic background or a soft bokeh. The optimization progression, illustrated in Figure 6.19, demonstrates that the composition converges to headshots of a single macaw around epoch 100. By the end of policy optimization, the preference-aligned model prefers to sample images of the blue-and-yellow macaw (Ara ararauna).

Figure 6.20 shows how samples from classes other than the macaw were affected by the macaw policy optimization. Many of the samples gained bokeh and increased contrast after policy optimization. The animals in the samples also often moved closer to the



(a) Log value loss

(b) Policy loss

(c) Reward

(d) KL penalty reward-contribution

Figure 6.17.: Training statistics from policy optimization for the *Macaw* experiment.

“camera” throughout training, and became easier to discern.

Figure 6.21 shows the reward distributions for the final reward model on samples from both the base policy and the optimized model on the class macaw and the classes other than macaw. The mean reward has increased by two standard deviations for the class macaw in the optimized model, along with a significantly reduced variance. For the classes other than macaw, the mean reward has increased by over half a standard deviation in the optimized model, with a similar variance to the base policy.

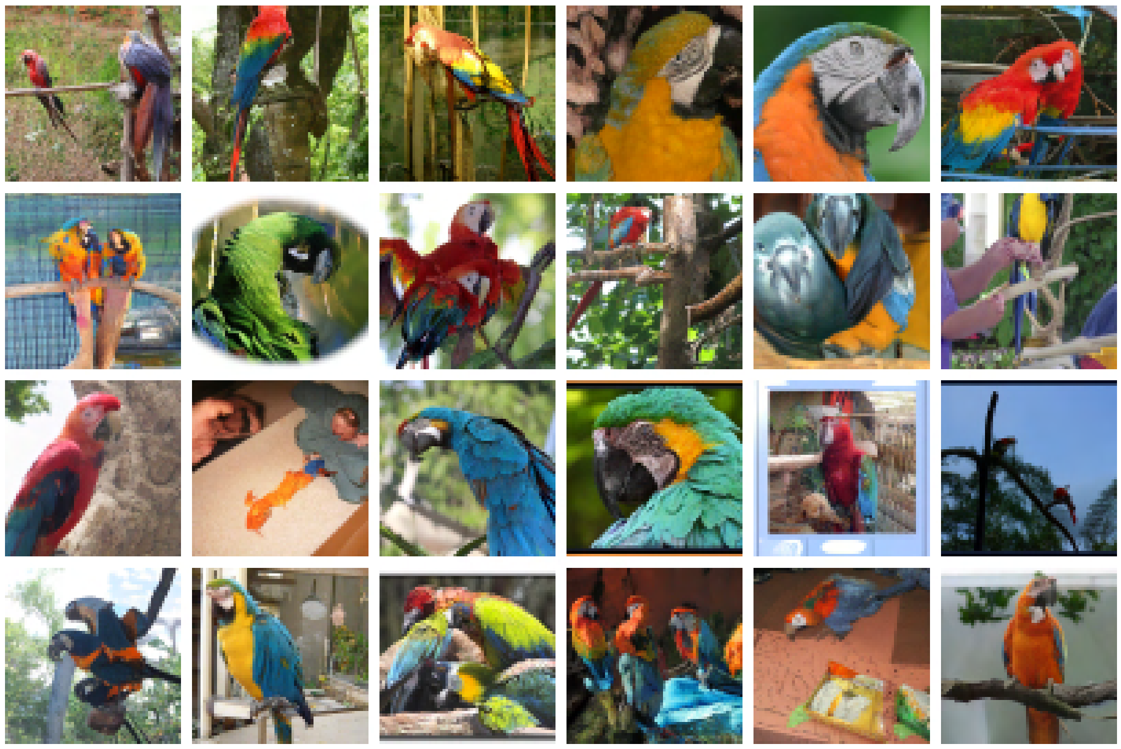
6.7.3. Evaluation

A validation accuracy of 85 % indicates that the preliminary reward model learned to differentiate between less preferred and more preferred image generations. This is corroborated by the comparison in Figure 6.15, where the tasks are ordered almost identically between the human labeler and the final reward model (bar a few differences in the lower-rated generations). However, as seen in the training statistics in Figure 6.16, the validation loss decreased notably during the initial epochs but increased throughout the remaining training duration. Despite the increasing validation loss, we did not consider this a major concern as it had little observable impact on the validation accuracy, and we considered the reward model good enough to validate the methodology.

Similar to previous experiments, the policy-optimized diffusion model successfully learned the objective from the reward model. This is evident in the reward histogram in Figure 6.21, which shows a clear shift toward higher rewards after policy optimization. Furthermore, the images generated by the policy-optimized model mostly match the preferences reported by the human labeler in Table 6.4.

An interesting observation arose when evaluating the macaw policy optimization on classes other than macaw. The results showed no apparent negative impact on other classes' image quality and, instead, demonstrated a positive transfer of learned preferences (Figure 6.20). We observed that these other classes exhibited more vibrant colors, bokeh backgrounds, and in some cases more detailed close-ups of the animals, suggesting that the alignment procedure is able to learn just the preferences of the human labeler, without negatively affecting the rest of the diffusion model, and that general preferences can be learned across domains.

Unlike in Experiment 3: *Aesthetic Score*, the KL penalty did not seem to significantly contribute to the reported reward. This implies that the KL penalty coefficient may be several orders of magnitude too low in this experiment, which could hamper the final model's diversity. Following the over-preservation of the original distribution in Experiment 3: *Aesthetic Score*, which stemmed from a too-high KL penalty coefficient, and the under-preservation in this experiment, it is clear that a more thorough study into effective values for the KL penalty coefficient is needed. We defer this to future work.


(a) Base model

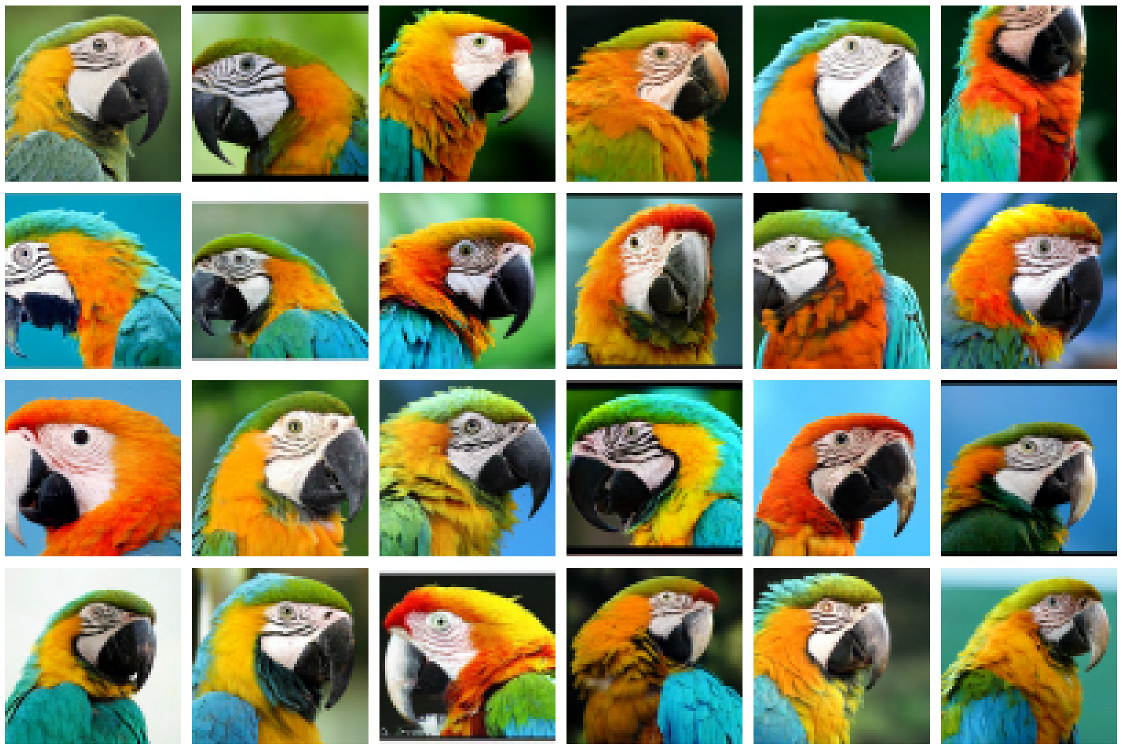

(b) Optimized model

Figure 6.18.: Samples of the macaw class from the base model and the optimized model for the *Macaw* experiment. The images are sampled using DDPM with 250 timesteps and without classifier guidance. The two grids are sampled using the same random conditions, starting noise, and seed.

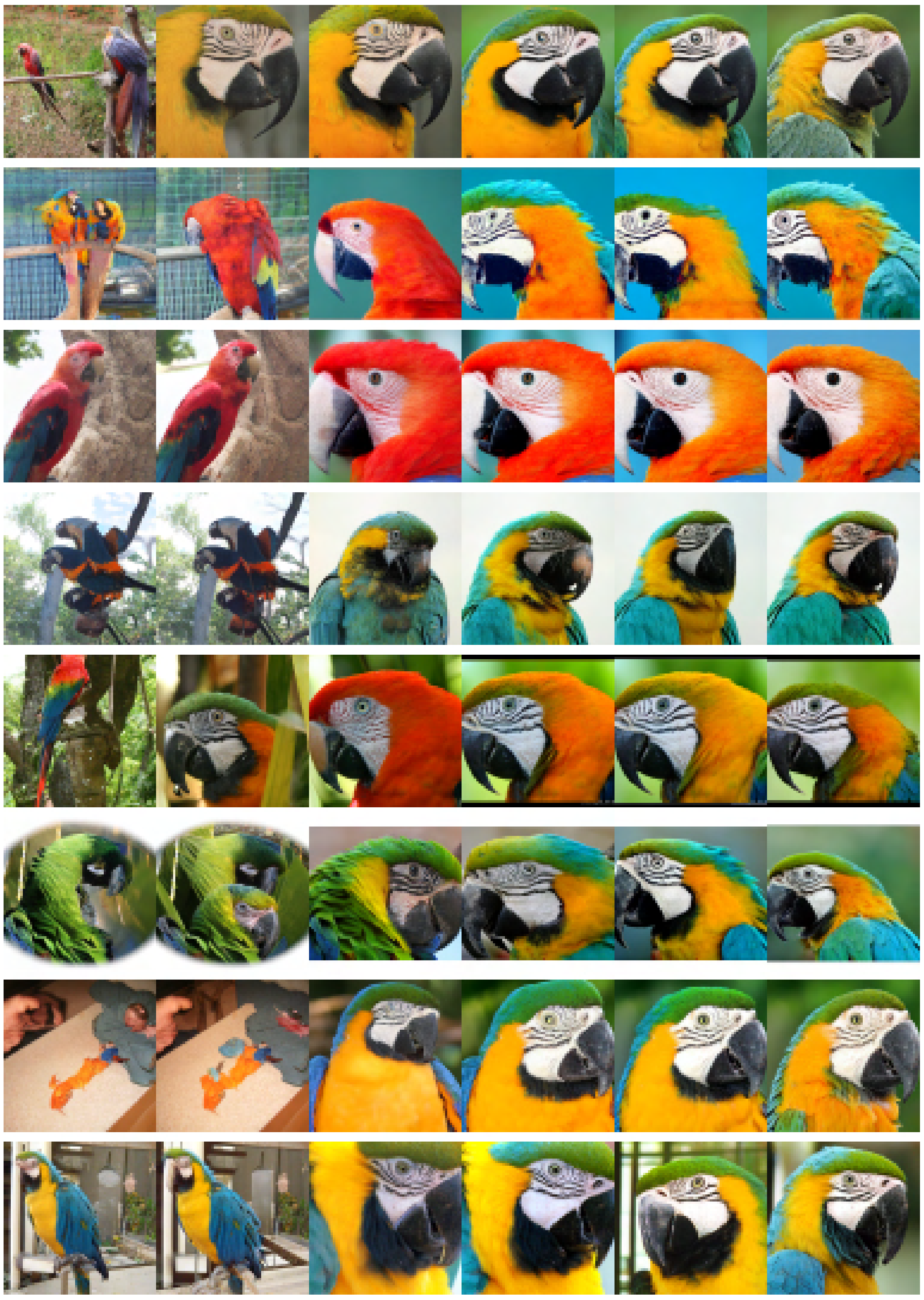


Figure 6.19.: Optimization progression of the macaw class for the *Macaw* experiment. Starting from the base model (left) and ending at the optimized model (right). The images are sampled using DDPM with 250 timesteps and without classifier guidance. Every image in a progression-row are sampled using the same random conditions, starting noise, and seed.

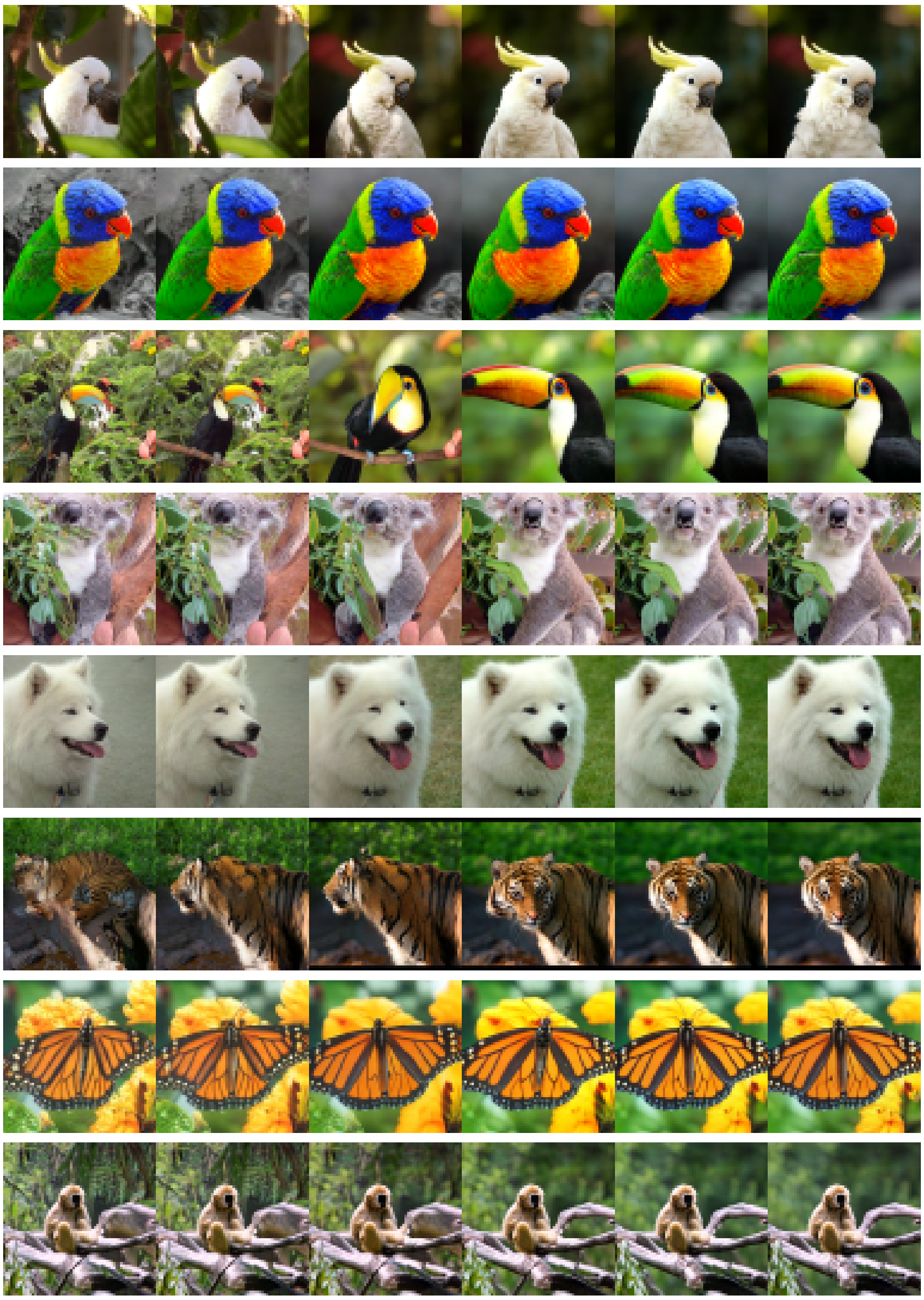
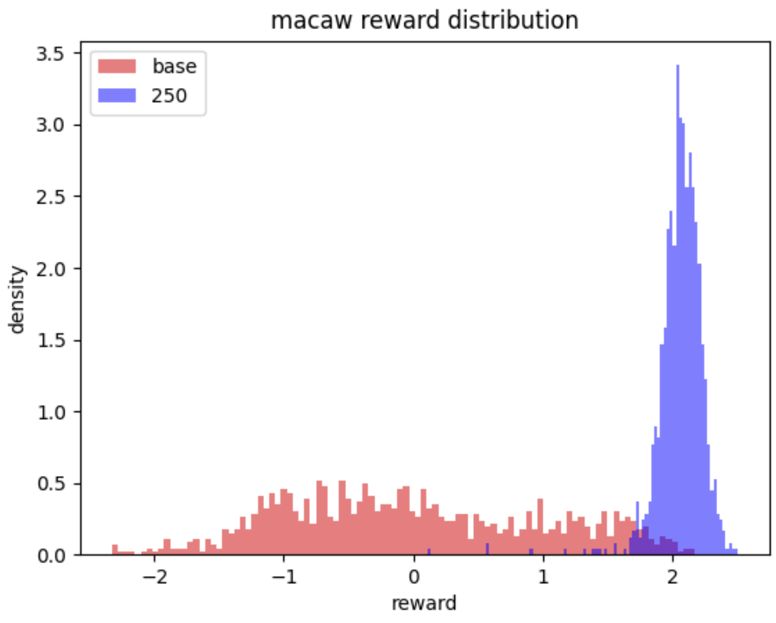
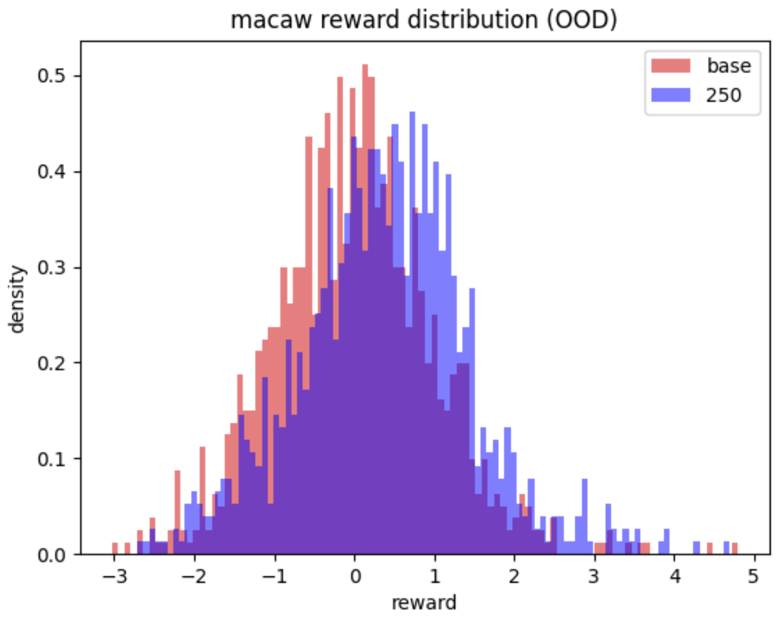


Figure 6.20.: Optimization progression for select classes other than macaw for the *Macaw* experiment. Starting from the base model (left) and ending at the optimized model (right). The images are sampled using DDPM with 250 timesteps and without classifier guidance. Every image in a progression-row are sampled using the same random conditions, starting noise, and seed.



(a) Class macaw



(b) Classes other than macaw

Figure 6.21.: Reward distributions for (a) class macaw, and (b) classes other than macaw, of both the base policy and the optimized model for the *Macaw* experiment. 1024 images were sampled per model. The histograms have been standardized around the reward distribution of the base diffusion model, giving it a mean reward of 0.0 and a standard deviation of 1.0.

6.8. Experiment 5: Macaw (Without RL)

In this supplementary experiment, we explore whether reinforcement learning is even necessary to align diffusion models with rewards. As such, the rewards are directly backpropagated through the sampling process instead of using reinforcement learning like in the policy optimization methodology presented in chapter 5. The experiment reuses the reward model from Experiment 4: *Macaw*.

6.8.1. Experimental Setup

The setup for this experiment was very similar to that of Experiment 4: *Macaw* to make the experiments as comparable as possible. As such, instead of collecting new preferences and training a new reward model, *Macaw (Without RL)* simply reused the reward model trained in Experiment 4: *Macaw*, making steps 1-3 of the methodology identical between the experiments. The base policy was also reused.

The experimental setup diverged from that of Experiment 4: *Macaw* in how the policy was trained. Since the entire pipeline of the proposed methodology is differentiable, from rollout to reward, *Macaw (Without RL)* could simply backpropagate the gradients from the loss all the way through the reward model and the policy. The loss was then defined as the negative of the reward, so that when the training objective attempted to minimize the loss, the reward was indirectly maximized. Since only the reward is needed to calculate the gradients, *Macaw (Without RL)* made away with the value function and base policy entirely, only needing one pass through the policy and reward model, compared to the two passes of the original methodology (to calculate log probabilities). As a result, *Macaw (Without RL)* is also unable to preserve the original distribution through the use of KL regularization. To not affect the reward model throughout training, its parameters were frozen.

Similar to Experiment 4: *Macaw*, 32 environments was used, generating 800 reinforcement learning states (32 images $\times$ 25 diffusion timesteps) per epoch. To save VRAM, the images were split into minibatches of 4 images each. The gradients for all 8 minibatches were accumulated into one big gradient update at the end of the epoch, for an effective batch size of 800.

Since there was no value function to learn, the linear warmup schedule of the policy was reduced to 100 optimization steps from 8192 optimization steps in Experiment 4: *Macaw* (comparatively 100 epochs vs 256 epochs, respectively).

For a full overview of hyperparameters, refer to Table 6.2.

6.8.2. Results

Logging of training statistics was not implemented for this methodology, and as such, no loss or reward graphs are provided.

In Figure 6.18, samples from the base policy and the optimized model can be seen. The optimized model tends to favor centered, close-up images of the macaw as opposed to the base policy. Upon qualitatively examining the images, we observe that the images have increased contrast and vividness, a simplistic background, or a soft bokeh. As depicted in Figure 6.19, the optimization process shows convergence toward headshots of macaws by the 300th epoch. Ultimately, the preference-aligned model leans towards generating images of the scarlet macaw (Ara macao).

Figure 6.24 shows how samples from classes other than macaw were affected by the macaw reward optimization. Many of the samples gained bokeh and few of the animals are easier to discern. There is no noticeable difference in composition from the base policy to the reward-optimized model.

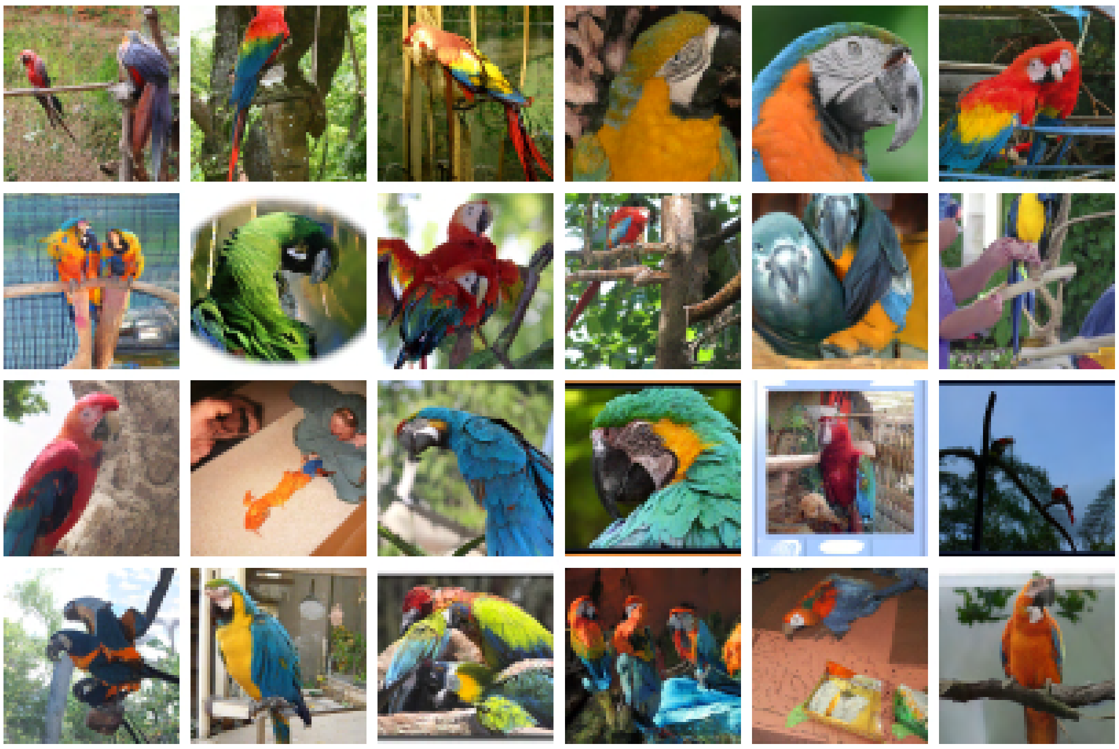
Reward distributions for images generated of macaw and classes other than macaw can be seen in Figure 6.25. The mean reward has increased by almost two standard deviations for the class macaw in the optimized model, along with a reduced variance. For the classes other than the macaw, the mean reward has increased by nearly a standard deviation in the optimized model, with a slightly increased variance to the base policy.

Images of classes other than macaw generated at 25 diffusion timesteps from the base policy, the optimized model from *Macaw* and the reward-optimized model from *Macaw (Without RL)* can be seen in Figure 6.26. We observe that the images generated by the model optimized without reinforcement learning have a red hue when generated with fewer timesteps.

6.8.3. Evaluation

Similar to in Experiment 4: *Macaw*, the images of macaws generated by the reward-optimized model show high image fidelity and follow the preferences reported by the human labeler (see Table 6.4, page 72). The most notable difference between the two optimized models when generating macaws, is how the model optimized with reinforcement learning generates images of the blue-and-yellow macaw, while the model optimized without reinforcement learning generates images of the scarlet macaw. However, this is likely because the two models randomly sampled different highly-rewarded macaw species in the early stages of the optimization process and that the momentum of the reward optimization led them to prefer different species. That there is no notable difference between preferring the two species is backed by the reward distribution histograms, which show that the two models are rewarded similarly after optimization, despite generating different species.

Interestingly, the model trained without reinforcement learning suffers from degradation when generating images of classes other than macaw at low timesteps, as the images gain a red hue. The red hue is likely an artifact from reward exploitation and overfitting on the scarlet macaws, and cannot be seen for the model optimized in Experiment 4: *Macaw*. The theory of reward exploitation is corroborated by the reward distribution charts, which show that the mean reward of the red-hued images have increased by almost one sigma, despite having noticeably decreased image fidelity. We speculate that a less aggressive learning rate, the introduction of KL regularization (which was disabled with the removal of the base policy) or a more robust reward model might help to mitigate this degradation, but a more thorough investigation of these potential solutions remains an area for future work.


(a) Base model

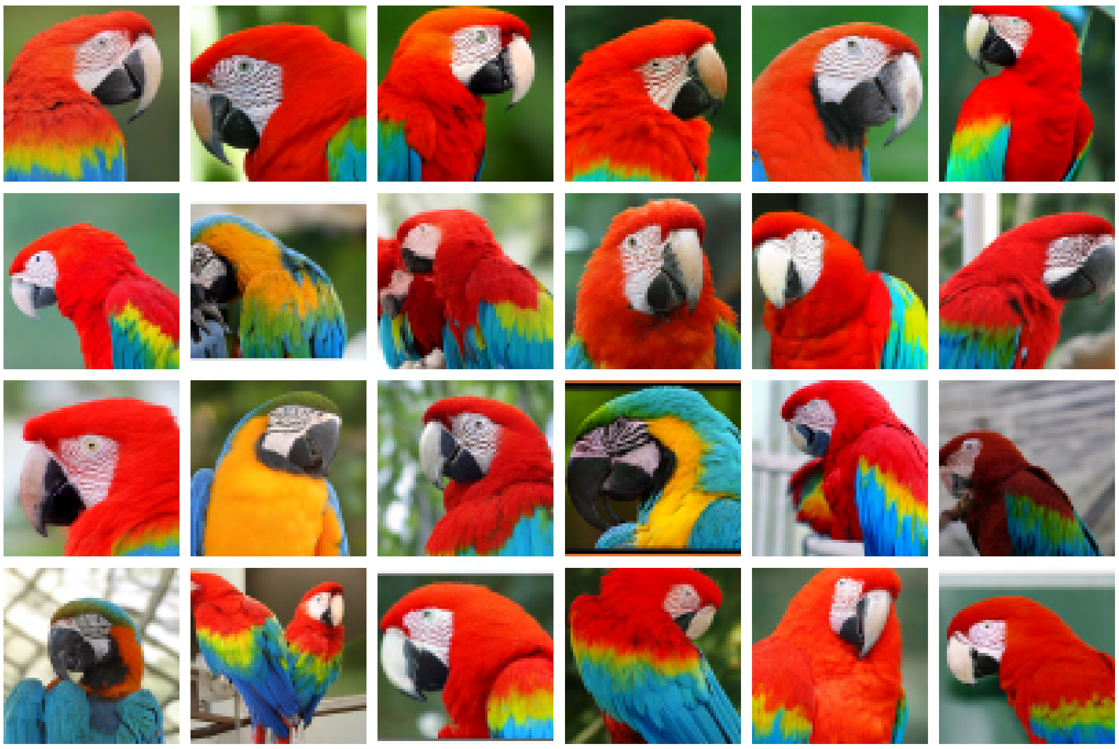

(b) Optimized model

Figure 6.22.: Samples of the macaw class from the base model and the optimized model for the *Macaw (Without RL)* experiment. The samples are sampled using DDPM with 250 timesteps and without classifier guidance. The two grids are sampled using the same conditions, starting noise, and seed.

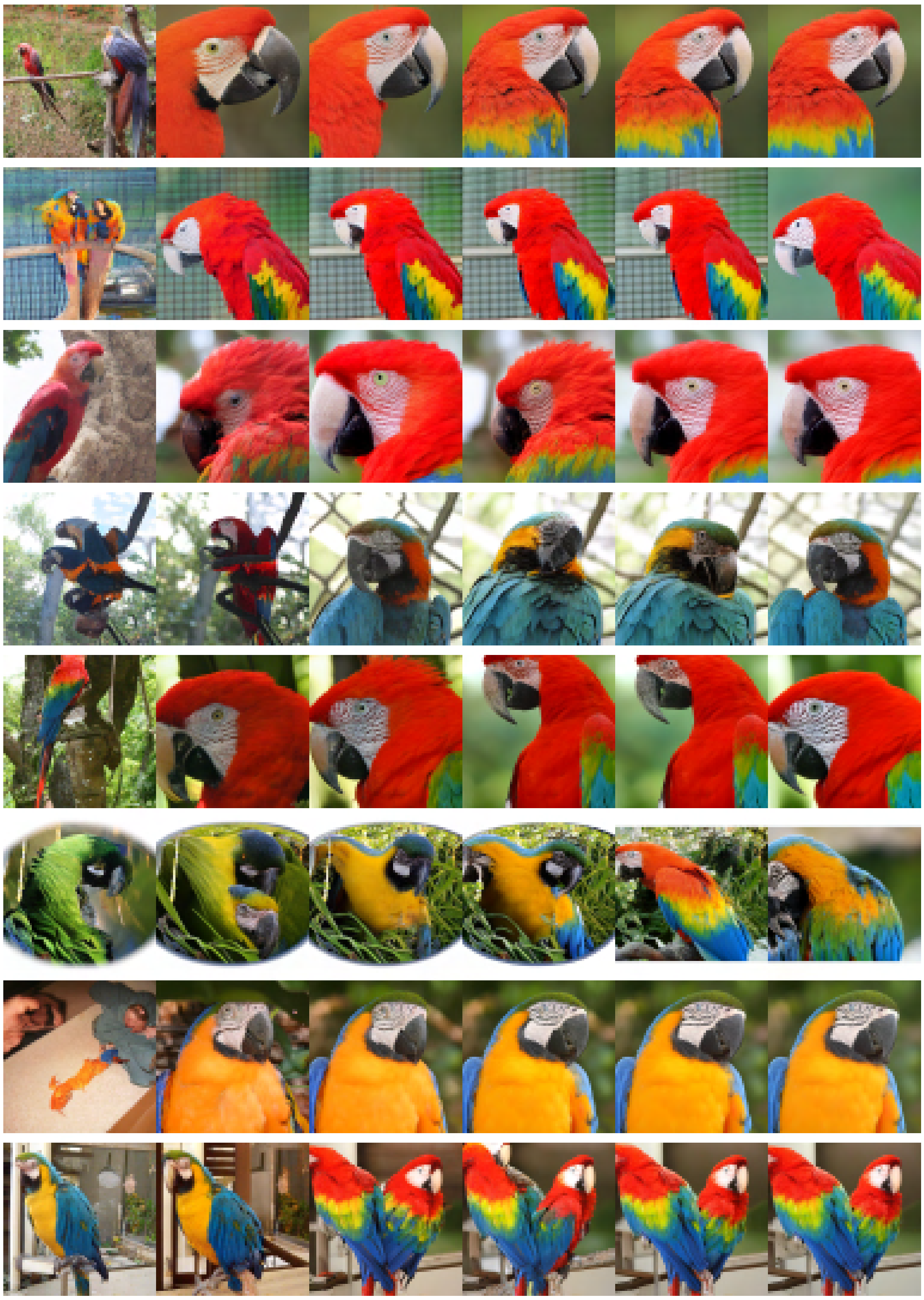


Figure 6.23.: Optimization progression of the macaw class for the *Macaw (Without RL)* experiment. Starting from the base model (left) and ending at the optimized model (right). The images are sampled using DDPM with 250 timesteps and without classifier guidance. Every image in a progression-row are sampled using the same random conditions, starting noise, and seed.

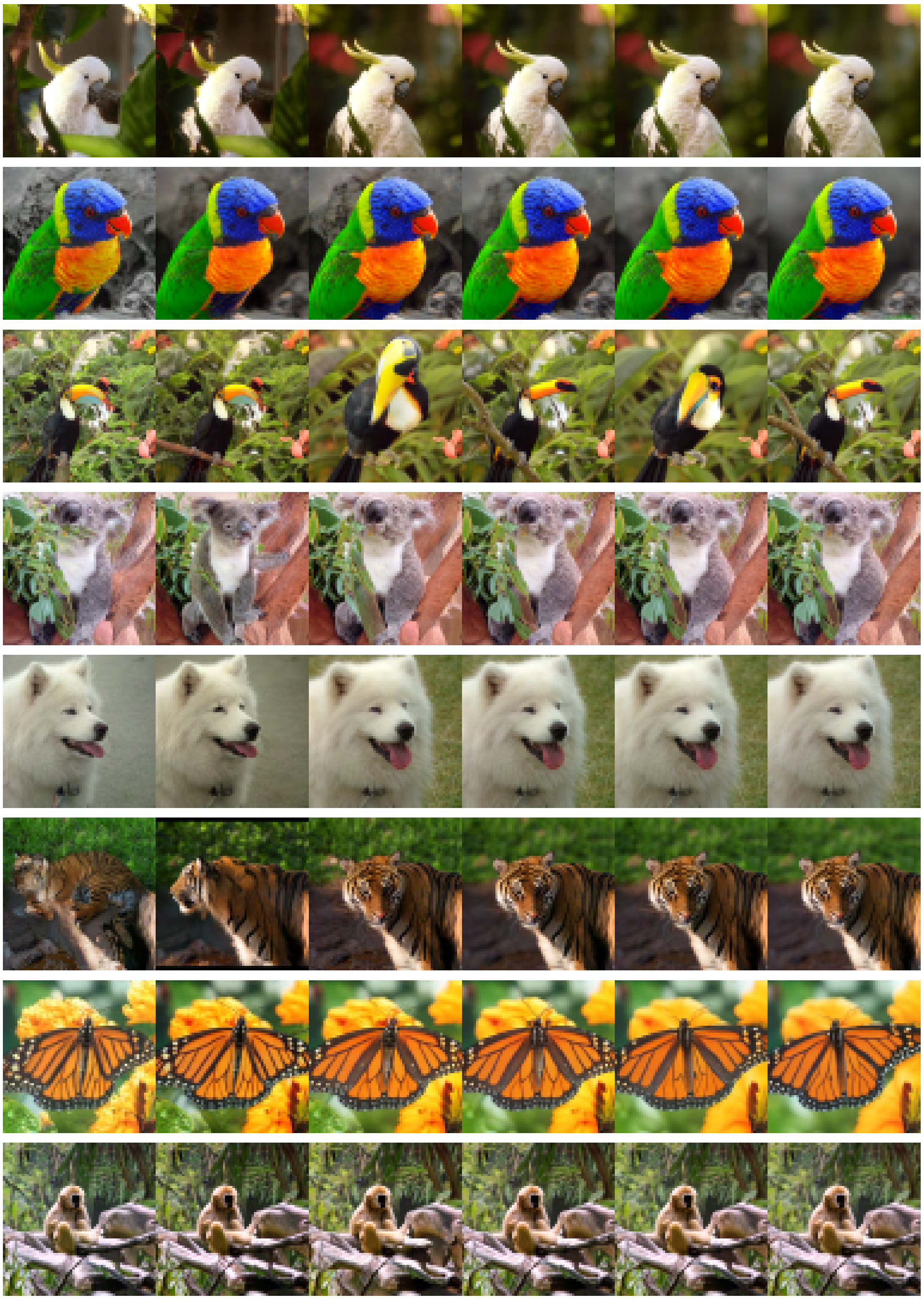
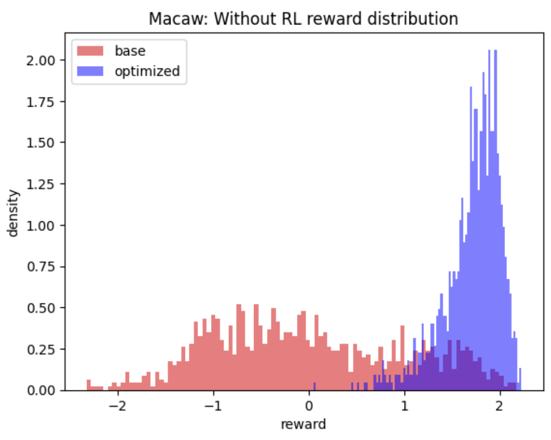
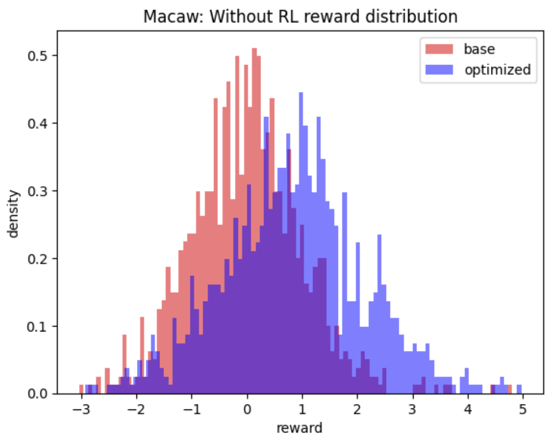


Figure 6.24.: Optimization progression of select classes other than macaw for the *Macaw (Without RL)* experiment. Starting from the base model (left) and ending at the optimized model (right). The images are sampled using DDPM with 250 timesteps and without classifier guidance. Every image in a progression-row are sampled using the same random conditions, starting noise, and seed.

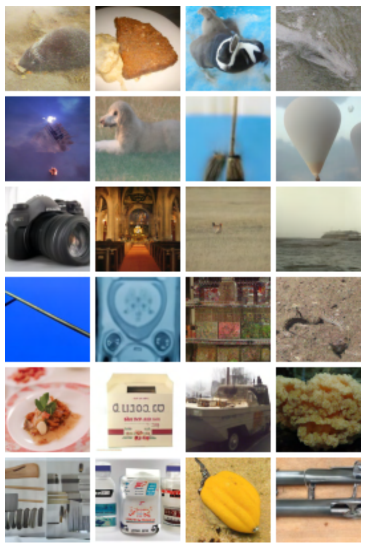


(a) Class macaw

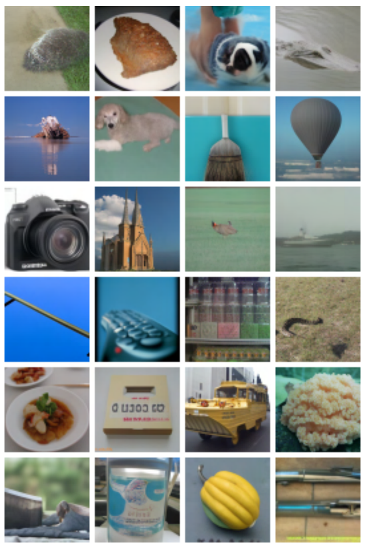


(b) Classes other than macaw

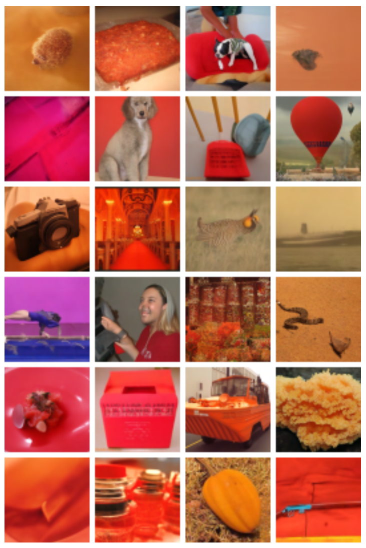
Figure 6.25.: Reward distributions for (a) class macaw, and (b) classes other than macaw, of both the base policy and the optimized model for the *Macaw (Without RL)* experiment. 1024 images were sampled per model. The histograms have been standardized around the reward distribution of the base diffusion model, giving it a mean reward of 0.0 and a standard deviation of 1.0.



(a) Base model



(b) *Macaw*



(c) *Macaw: Without RL*

Figure 6.26.: Samples from the base model and the optimized models for the *Macaw* and *Macaw: Without RL* experiments. The images are sampled using DDPM with 25 timesteps and without classifier guidance. The grids are sampled using the same conditions, starting noise, and seed.

Aligning Diffusion-Based Text-to-Image Models Using Human Feedback

6.9. Experiment 6: Pink

The sixth experiment, dubbed *Pink*, serves as the main implementation of the methodology with a State-of-the-Art (SOTA) text-to-image model. The primary motivation behind this experiment was to show that the methodology presented in chapter 4 and chapter 5 can align text-to-image model Stable Diffusion to human preferences. The experiment uses a very restricted prompt domain of various animals wearing pink sunglasses, with the goal of aligning the model to generating images in the prompt domain with high fidelity and image-text alignment, something the model was not already consistently capable of doing, and to see if this could generalize to unseen animals and colors of sunglasses.

6.9.1. Experimental Setup

The base policy for the experiment was Stable Diffusion v1.5 (Rombach et al., 2022), a text-to-image latent diffusion model. The remaining experimental setup is presented according to the four steps introduced in the high-level methodology in section 1.3.

Refer to Table 6.2 for a comprehensive overview of the experiment's hyperparameters.

Step 1: Curating a Labeling Dataset. To simplify preference collection and reward model training, the policy optimization was limited to a few specialized prompts centered around "animals wearing pink sunglasses". The prompt domain was chosen because it was a task the base policy was already somewhat capable of generating compelling and aligned images of, but also a task that it had not yet mastered. We theorized that this restricted prompt domain would effectively show the capabilities of the methodology without requiring the collection of vast amounts of human preferences. The four prompts used in the experiment can be seen in Table 6.5. The prefix "A photograph of" was added to avoid generating images of cartoons, children's drawings or graphics designs.

Num.	Prompt
1	A photograph of a **corgi** wearing pink sunglasses.
2	A photograph of a **deer** wearing pink sunglasses.
3	A photograph of a **lion** wearing pink sunglasses.
4	A photograph of a **mouse** wearing pink sunglasses.

Table 6.5.: Prompts for the *Pink* labeling dataset tasks. The animals have been highlighted in bold.

Because the base policy was better at this objective for the animals lion and corgi, than for deer and mouse, the task size was increased to $K = 8$ (from $K = 5$ in Experiment 4: *Macaw*) to ensure that the model was capable of producing tasks that included some good and bad samples for all prompts most of the time. As a result, the labeling dataset consisted of 120 tasks, for a total of 960 images. To speed up rollout and policy optimization, the images were sampled at 16 diffusion timesteps. All images had a resolution of 512×512, and had the classifier-free guidance scale set to 7.5.

Step 2: Collecting Human Feedback. Like in Experiment 4: *Macaw*, a single human labeler (one of the authors) was used for preference collection.

The 120 tasks were labeled using the labeling software and methodology presented in chapter 4. To speed up the preference collection, transitivity was assumed when ordering the images. The instructions for the preference collection, which can be seen in Table 6.6, were designed to split the generated images into four categories ("animal with sunglasses", "animal", "sunglasses" and "other") to make objective and consistent labeling easier.

Num.	Instruction
1	Always prefer images with **{animal}** wearing pink sunglasses over images with **{animal}** not wearing pink sunglasses.
2	Always prefer images with just **{animal}** over images without **{animal}**.
3	Always prefer images with just sunglasses over images without **{animal}** or sunglasses.
4	Within these categories, always prefer images with pink sunglasses over images with sunglasses in other colors.
5	Within these categories, rank the images in order of what you would most prefer as output from the model.

Table 6.6.: Instructions for the *Pink* preference collection. Here, **{animal}** refers to the animal in the task prompt.

Step 3: Training a Reward Model. The reward model for the experiment is described in the methodology presented in chapter 5.

To evaluate the reward model's performance, a preliminary reward model was first trained on 80 % of the labeled tasks and validated on the remaining 20 %. The final reward model was then trained on all of the tasks. Both models were trained using a learning rate of 1×10^{-4}, and a batch size of 4 tasks (constituting 32 images) for 40 epochs. The reward model was standardized to a mean reward of 0.0 and a standard deviation of 1.0.

Step 4: Optimizing Policy with Reinforcement Learning. With limited knowledge of good KL penalty coefficients or how various degrees of KL penalty would affect the policy from previous experiments, the KL penalty was disabled entirely for *Pink*.

Without a good value for the KL penalty coefficient, it was decided it was better to leave it off so the optimization of the policy was not unnecessarily discouraged by a too strong penalty. Besides, by disabling KL penalty, the frozen base policy was no longer needed, clearing VRAM for a larger batch size and faster policy optimization.

Because of the memory requirements of keeping the policy, reward model and value function in memory at the same time, it was no longer possible to run policy optimization with a decent number of environments with just 24 GBs of VRAM. The policy model and value function were therefore trained on a Dell DSS 8440 node of 10 $\times$ NVIDIA A100 80GB GPUs. To optimize the policy and value function across multiple GPUs, PyTorch's Distributed Data Parallel (DDP) was used. With DDP, every GPU held its own identical copy of the models' parameters, and performed its own rollouts and gradient calculations. Before every optimization step, the gradients were then synchronized so that all of the models were updated with the same gradients across the GPUs, ensuring the models were still identical copies by the start of the next epoch. Unlike previous experiments, *Pink* was also trained using mixed precision and TF32 to speed up optimization.

Because of the complexity of the policy, the number of environments was also increased to 80 (8 per GPU $\times$ 10 GPUs). The images generated during rollout used the same settings as when curating the labeling dataset, for a total of 1280 reinforcement learning states (80 images $\times$ 16 diffusion timesteps) per epoch. The states were then split into 80 minibatches (of size 16) for policy optimization. The states were not shared across GPUs.

The learning rates for the policy and value function were decreased by one order of magnitude from Experiment 4: *Macaw*, to 1×10^{-7} and 1×10^{-6}, respectively. This was done to prohibit aggressive changes to the policy. To speed up the start of the policy optimization, the linear warmup schedule of the policy was decreased from 8192 to 4096, as experience from previous experiments saw the value function "good enough" by this point in training.

6.9.2. Results

The results of the experiment are divided into the sections "Human Preference Collection", "Reward Model Training" and "Policy Optimization", for clarity.

Human Preference Collection

Four example tasks preference-ordered by the human labeler can be seen in Figure 6.28. The preference ratings match the preference feedback provided by the human labeler after the experiment shown in Table 6.7.

Num.	Preference Feedback
1	Preferred images with **{animal}** correctly wearing pink sunglasses.
2	Preferred images with **{animal}** wearing stylish sunglasses.
3	Preferred images with a single **{animal}**.
4	Preferred images with portrait shots of **{animal}**.
5	Preferred images with **{animal}** looking at the “camera”.
6	Preferred images with bokeh.
7	Did not prefer images without **{animal}** or sunglasses.

Table 6.7.: Feedback from the human labeler when asked about their *Pink* labeling preferences after the experiment. Where **{animal}** refers to the animal in a given task’s prompt.

Reward Model Training

The reward model training statistics can be seen in Figure 6.27. Both the preliminary and final reward models approached a near zero loss throughout training, and an accuracy close to 100 %. The validation loss of the preliminary reward model decreased for the first four epochs and increased throughout the rest of training. The validation accuracy stabilizes around 80 % after the first ten epochs of training.

Four example tasks ordered by the trained reward model can be seen in (b) in Figure 6.28. The reward model is consistently able to order the images into the categories defined by the instructions. Within these categories, the correspondence between the reward model’s and the human labeler’s rankings is mixed. The accuracy of the reward model compared with the human labeler is 81.7 % (albeit with a smaller task size of $K = 6$).

Policy Optimization

The policy optimization training statistics can be seen in Figure 6.29. Throughout training, the log value loss decreased, while the policy loss and the reward increased. At the end of training, the reward increased by four sigma. A checkpoint saved at epoch 2028 was selected as the final “optimized” model. At this point, the mean reward was 1.5.

Sample images from the base model and the optimized model can be seen in Figure 6.30. The images generated by the base model show a varying degree of success, with the images of the animals lion and corgi achieving the best image-text alignment. For the images of the animals lion and mouse, the glasses are often simple, cartoonish or disfigured. There is little detail in the background of most of the images from the base model, and only some of the images have bokeh. All images from the optimized model show the correct animal with a pair of pink sunglasses. The sunglasses are more realistic and stylish. In

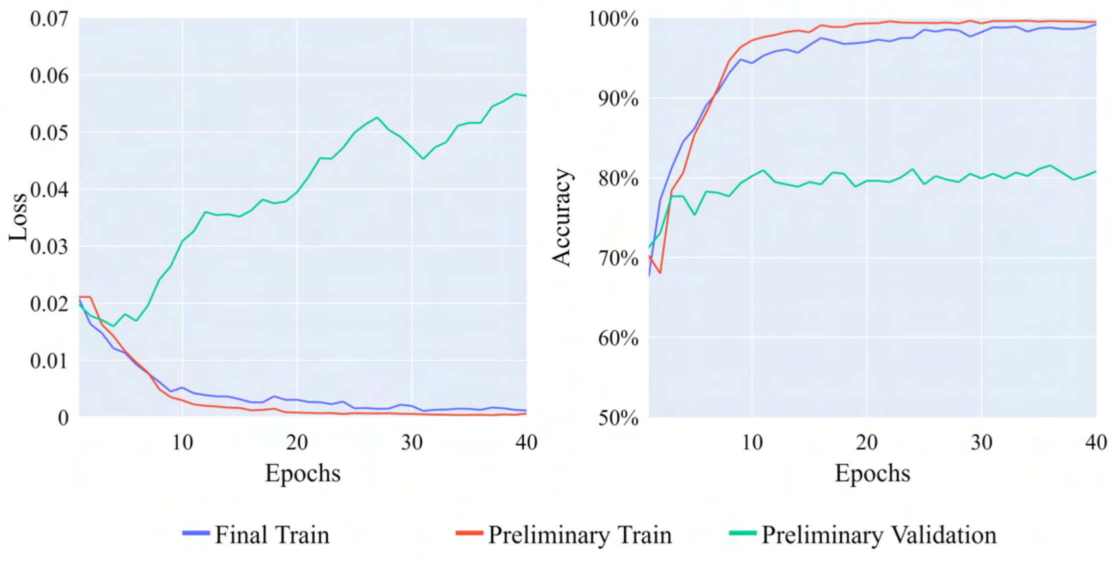


Figure 6.27.: Training and validation statistics for the *Pink* reward model. The y-axis in the accuracy graph starts at 50 %.

the images of the animals mouse, corgi and deer, the animals have moved closer to the “camera”. All of the images from the optimized model have a degree of bokeh.

Figure 6.31 shows samples from the base model and the optimized model on partially related and unrelated prompts. The optimized model is consistently able to generate images with other animals wearing pink sunglasses, as well as sunglasses of other colors, something the base model struggles with. When generating images of animals, regardless of them wearing sunglasses or not, the optimized model prefers the animals closer to the “camera” than the base model. The images generated by the optimized model with completely unrelated prompts are mostly unchanged from the base model.

Progression images generated at various stages throughout optimization for both original, related and unrelated prompts are shown in Figure 6.32. As the policy is overoptimized, it experiences a local mode collapse, whereby the images generated with prompts close to the original prompts (e.g., animals wearings various-colored sunglasses) drastically lose variety, and the animals become indistinguishable from one another. The animals not wearing sunglasses generated by the over-optimized model are also affected by the local collapse by relation, whereby the images generated of them too lose variety, but are still visually distinguishable from other animals. The unrelated prompts change minimally from base to overoptimized model.

Reward distributions for the base and optimized models using the original, relevant and unrelevant prompts are shown in Figure 6.33. For the original prompts, the base model can be divided into three reward distributions: (1) a very small distribution from rewards $(-2, -3)$, (2) a small distribution at reward -1, and (3) the main distribution from rewards $(0, 1)$. The optimized model has two distributions for the original prompts, one with a mean reward just below 2 and the other with a mean reward just below 3. The reward distributions for different prompts than the original prompts are not divided into smaller distributions to the same degree. As the prompts deviate further from the original prompts, the reward distributions for the base and optimized models became more and more similar.

6.9.3. Evaluation

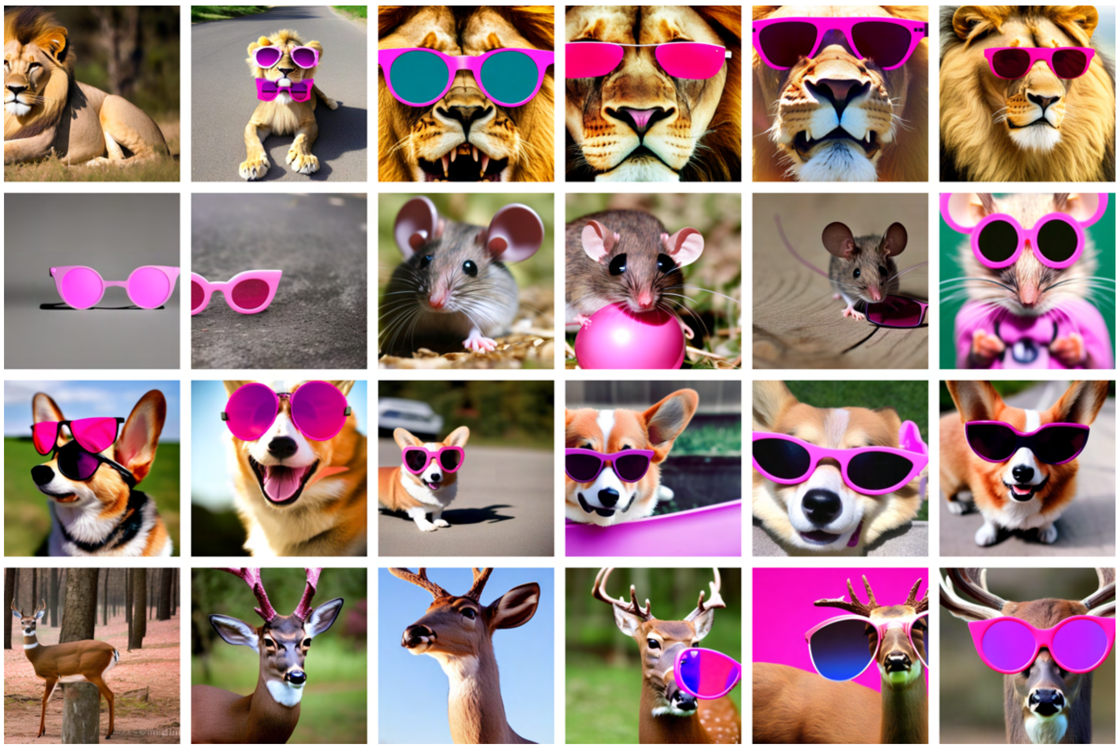
The preliminary reward model trained in the *Pink* experiment achieved a validation accuracy of 80.8 %. This matches the example tasks ordered by both the human labeler and the final reward model in Figure 6.28, which had an effective accuracy of 81.7 % (albeit with a smaller task size of $K = 6$, compared to $K = 8$ for the validation tasks).

However, while the reward model seems capable of ordering sample images according to human preferences, the degree to which it actually checks image-text alignment is uncertain. The only changing variable in the list of task prompts for this experiment is the animal. In our experience, when the base policy generated images for reward model labeling, it never once swapped the target animal for another. This means that the reward model could potentially always rely on the animal in the sampled image being correct, removing all variables from the prompt, and then learn to completely ignore the given prompt in its assessment of the image. This uncertainty could have been cleared with a more complex task specification, which we defer to future work.

Unlike Experiment 3: *Aesthetic Score*, the log value loss continued to decrease throughout the duration of the policy optimization. Likewise, the reward kept increasing until training stopped. This is likely a result of disabling KL penalty, which the value function would have had to optimize for (which increases the value loss) and which would have prohibited further reward optimization once the distributions started to deviate too much. That the reward kept increasing despite the visual fidelity and image-text alignment degradation experienced towards the end of training is a clear sign of reward exploitation, where the policy starts to exploit weaknesses in the reward model to keep gaining reward. Also noticeable as the policy is over-optimized is how the animals in the images sampled with the original prompts converge into one animal. This is a sign that the policy is experiencing a local mode collapse when sampling for these, or very similar, prompts. That the reward kept increasing through the reward model exploitation and policy mode collapse is a sign that the reward metric cannot be trusted to accurately evaluate model performance.

The three reward distributions visible for the base model using the original prompts are likely a result of the explicit instructions given to the human labeler before preference collection. Following the instructions, the highest-rewarded distribution would then likely be made up of images with both an animal and a pair of sunglasses. The second-highest-reward distribution would be made up of the images with just animals, leaving the last distribution for the images with just sunglasses or neither animal nor sunglasses.

Altogether, the experiment successfully captured the preferences of the human labeler in a reward model, and aligned Stable Diffusion v1.5 to those preferences. We also showed that the optimized policy was capable of generalizing to animals and colored sunglasses not seen in the alignment process. We also saw that the further away from the original prompts, the less the policy is changed. Ideally, KL regularization should be used to align the policy to improve sample variety and prevent mode collapse. However, due to a lack of knowledge on how KL penalty affects this specific policy and reward model, and what would constitute a good KL penalty coefficient to adequately preserve the original distribution, this was left for future work.


(a) Human labeler

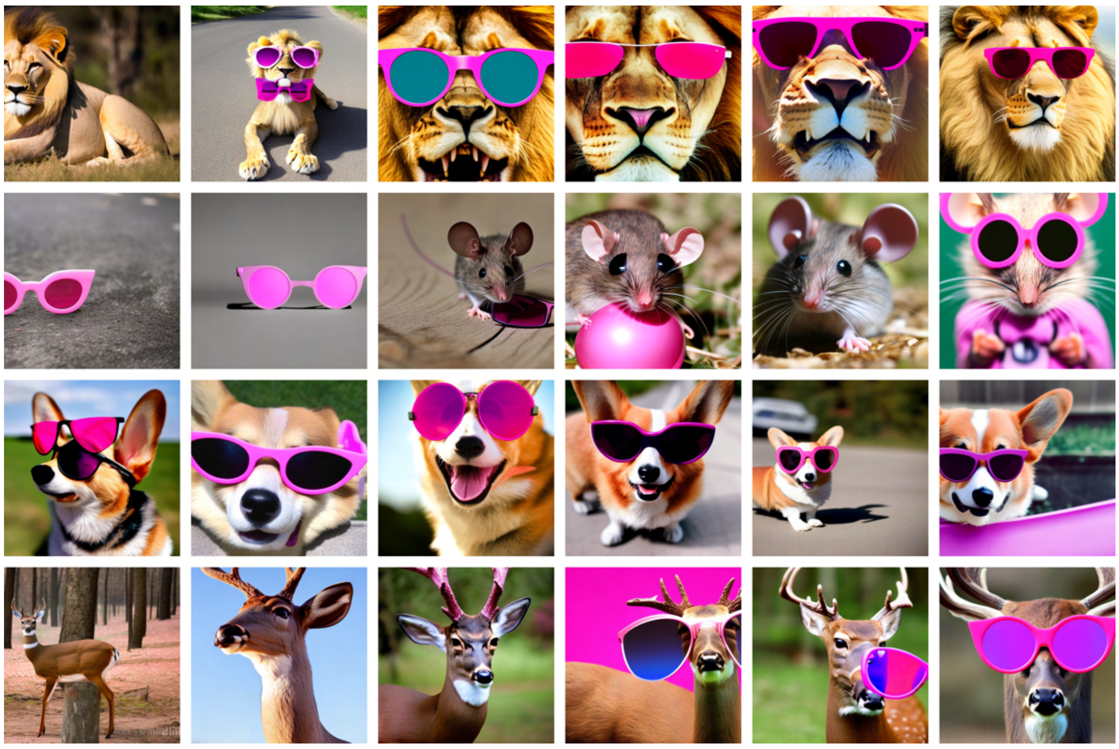
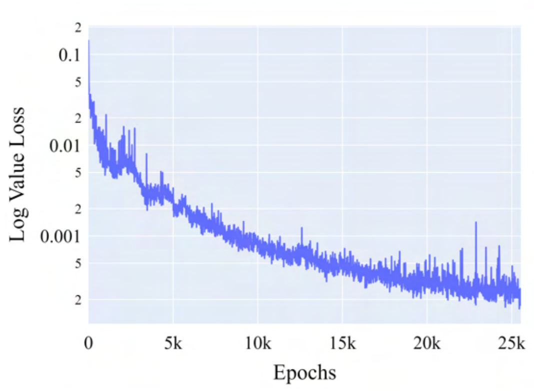
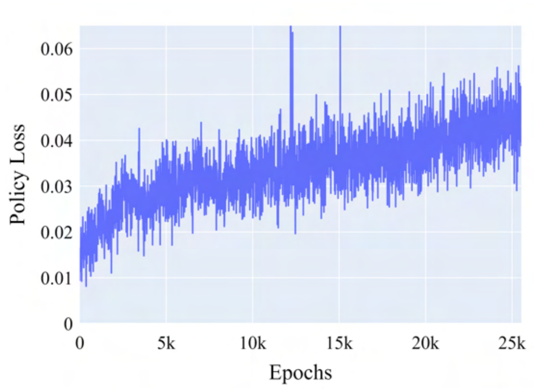

(b) Reward model

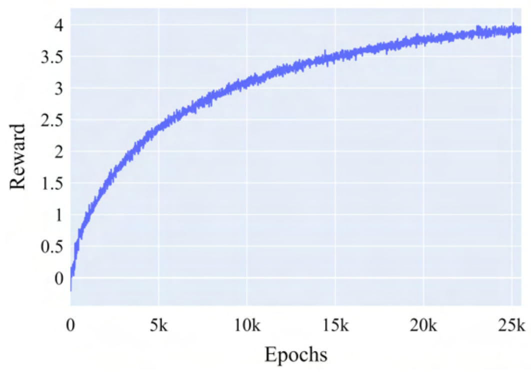
Figure 6.28.: Four example tasks (one per row) of six images each, preference-rated by (a) the human labeler, and (b) the reward model for the *Pink* experiment. The images in a task are ordered from least preferred (leftmost) to most preferred (rightmost).



(a) Log value loss

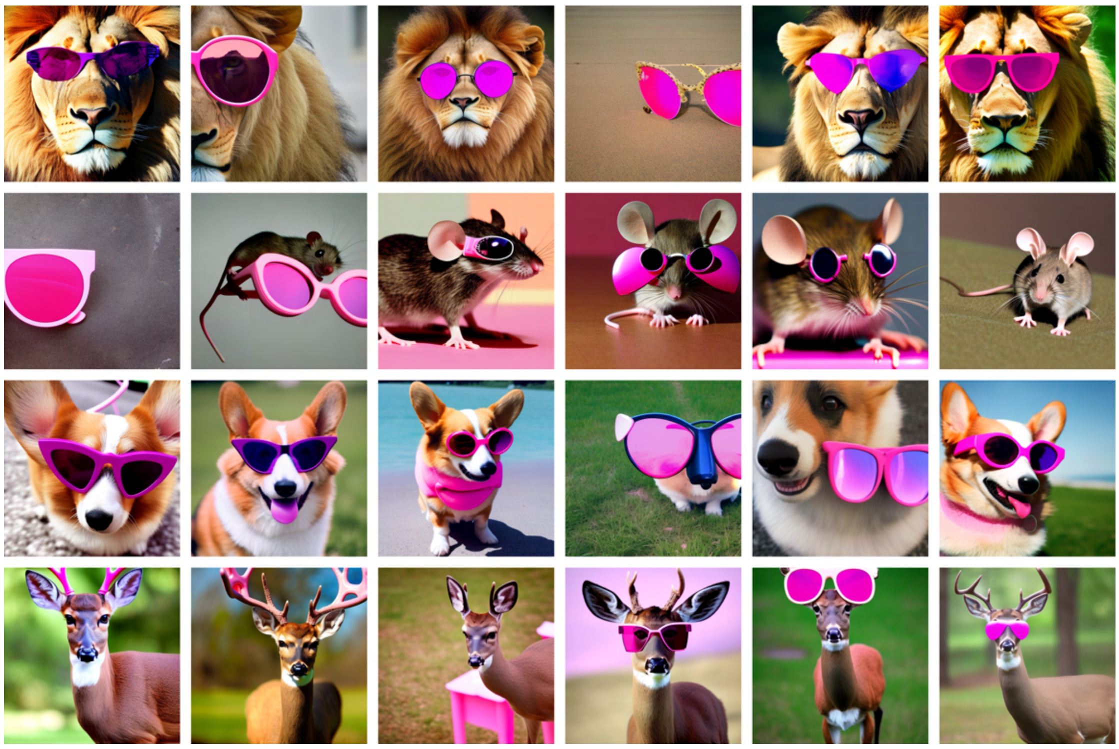


(b) Policy loss



(c) Reward.

Figure 6.29.: Training graphs from the policy optimization of *Pink*.


(a) Base model

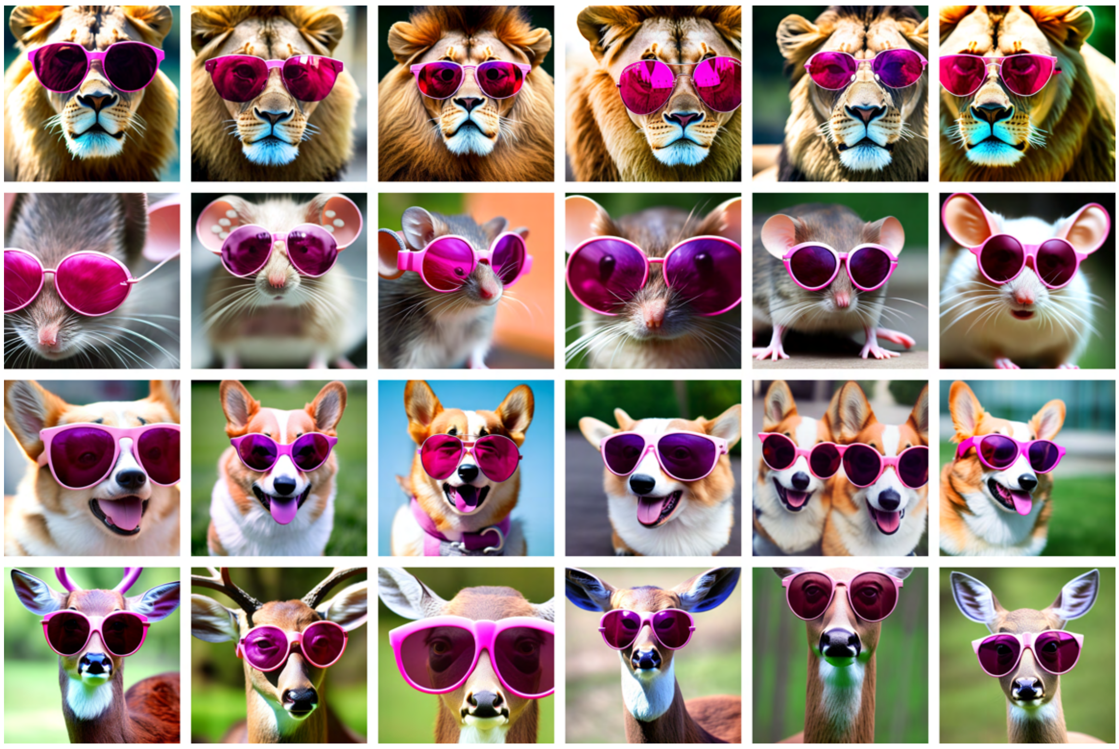
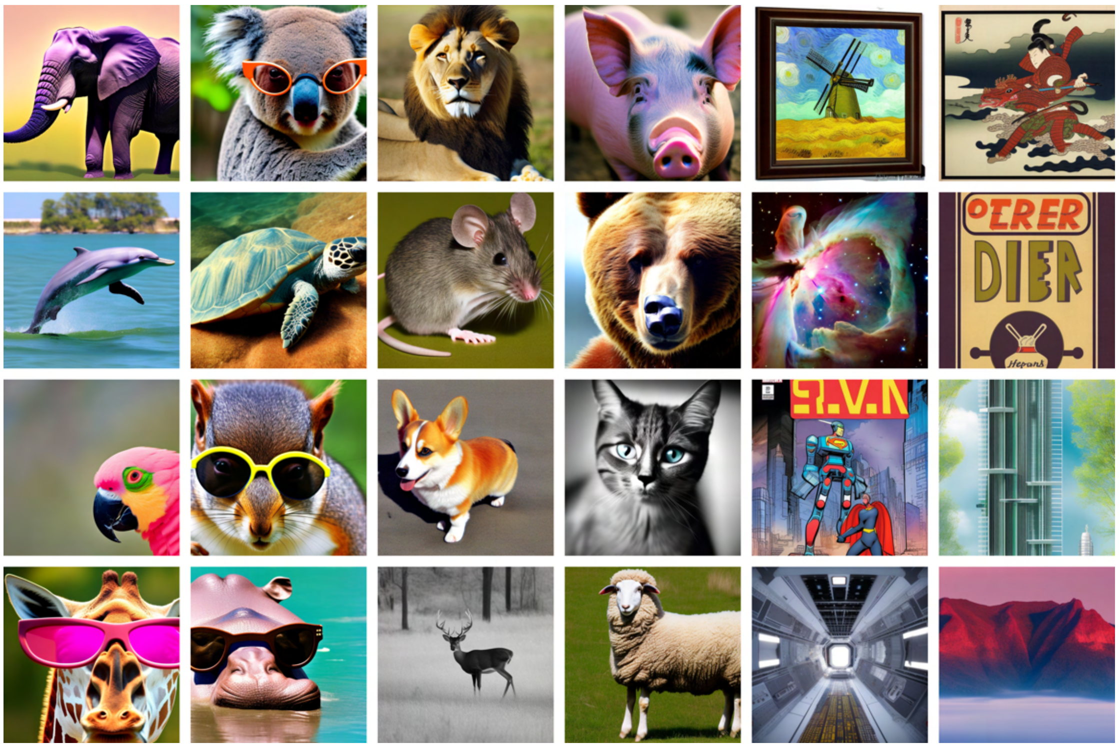
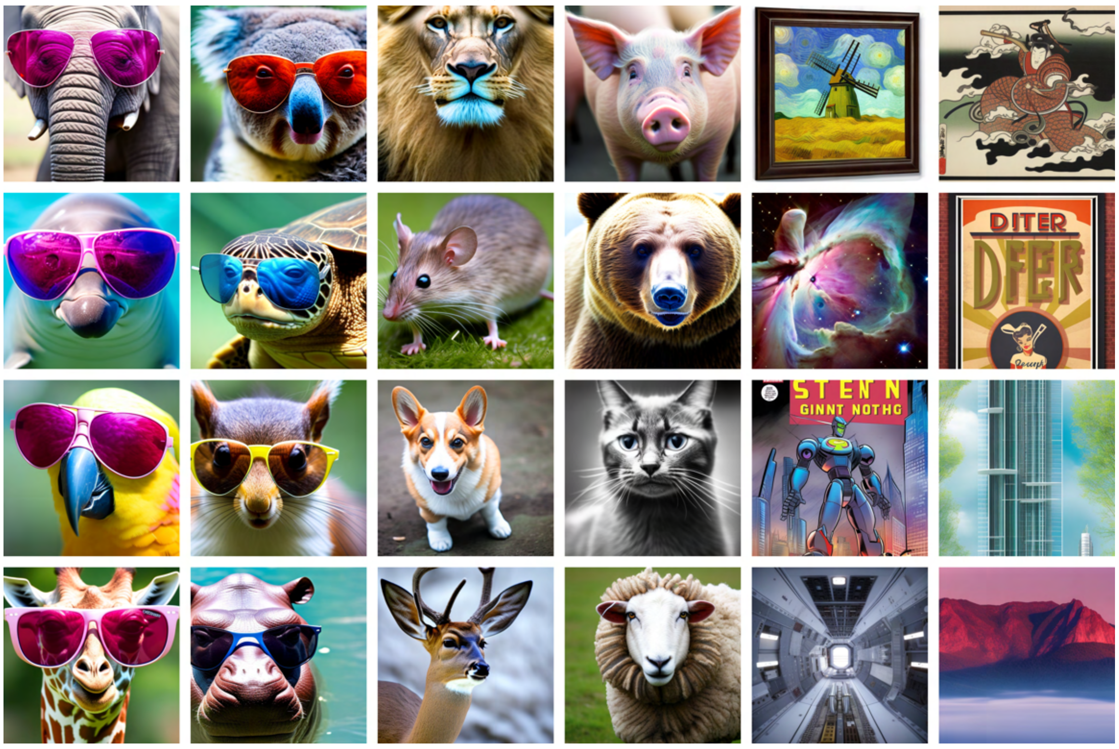

(b) Optimized model

Figure 6.30.: Samples from the base model and the *Pink* optimized model. The images are sampled using DDPM with 25 timesteps and with classifier-free guidance scale 7.5. The two grids are sampled using the same conditions, starting noise, and seed.

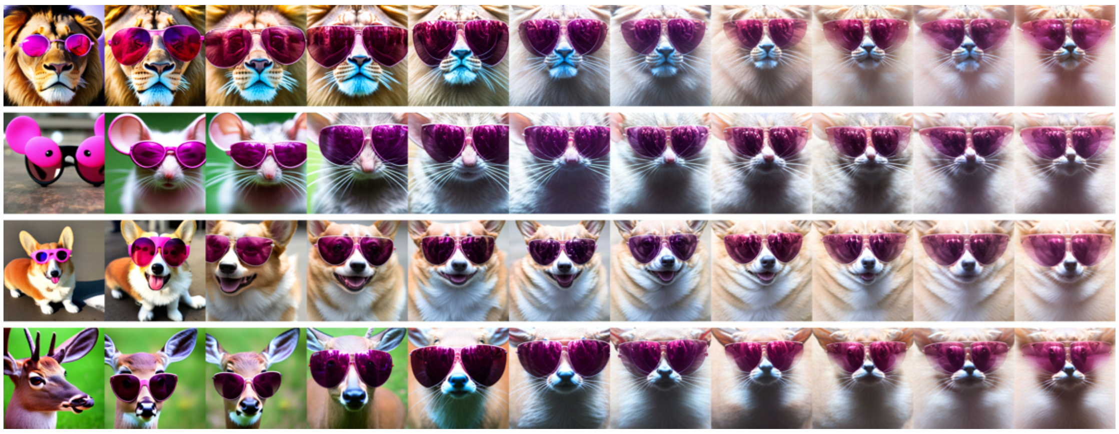


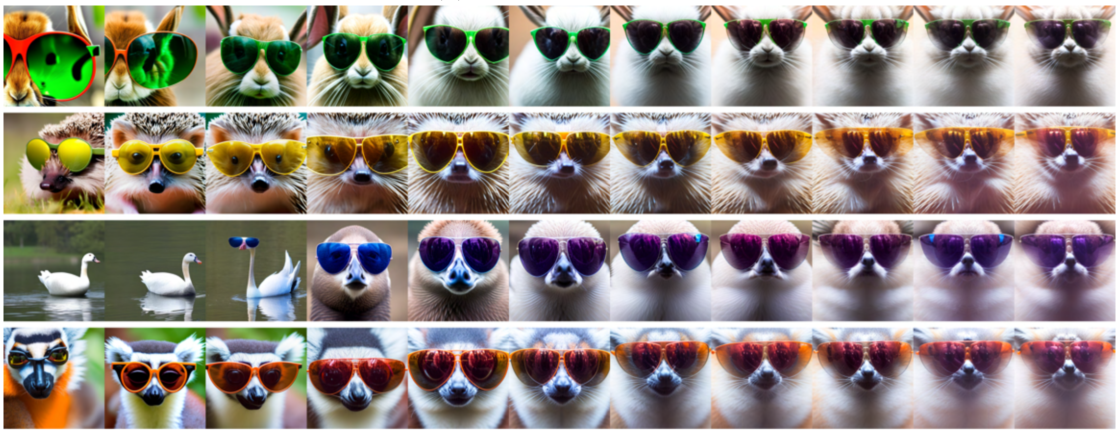
(a) Base model



(b) Optimized model

Figure 6.31.: Samples from the base model and the *Pink* optimized model using a variety of partially related and unrelated prompts. The first two columns show animals wearing pink and colored sunglasses. The next two columns show animals *without* sunglasses. The final two columns show completely unrelated prompts. The images are sampled using DDPM with 25 timesteps and with classifier-free guidance scale 7.5. The two grids are sampled using the same conditions, starting noise, and seed.


(a) Original prompts.


(b) Modified prompts with different animals and colored sunglasses.

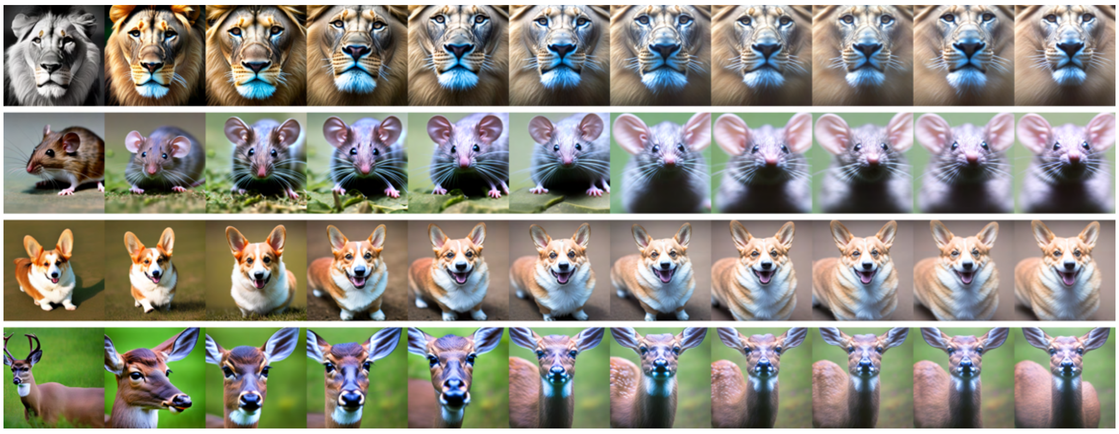
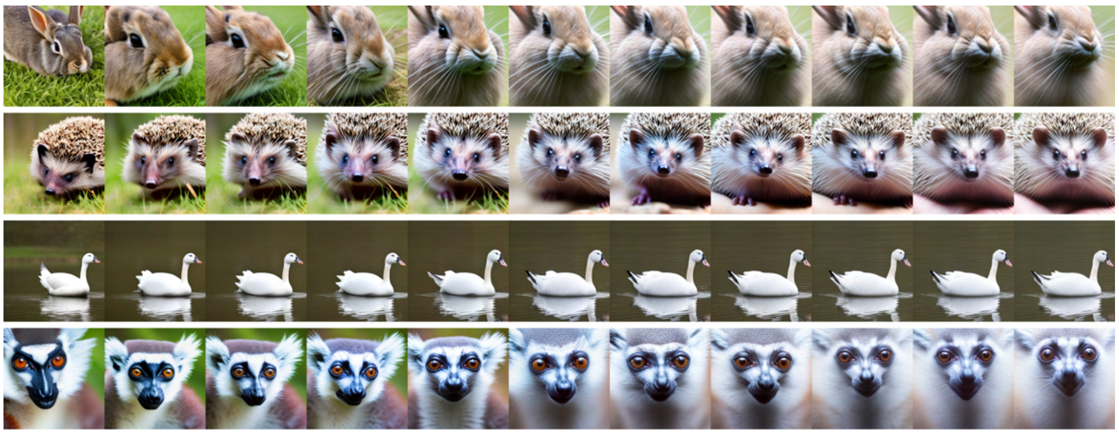

(c) Modified prompts with original animals without sunglasses.

Figure 6.32.: Over-optimization progression of the *Pink* policy for different prompts showing different levels of overfitting. From the base model (left) until epoch 20,000 (right), at 2000 epoch increments.


(d) Modified prompts with different animals without sunglasses.

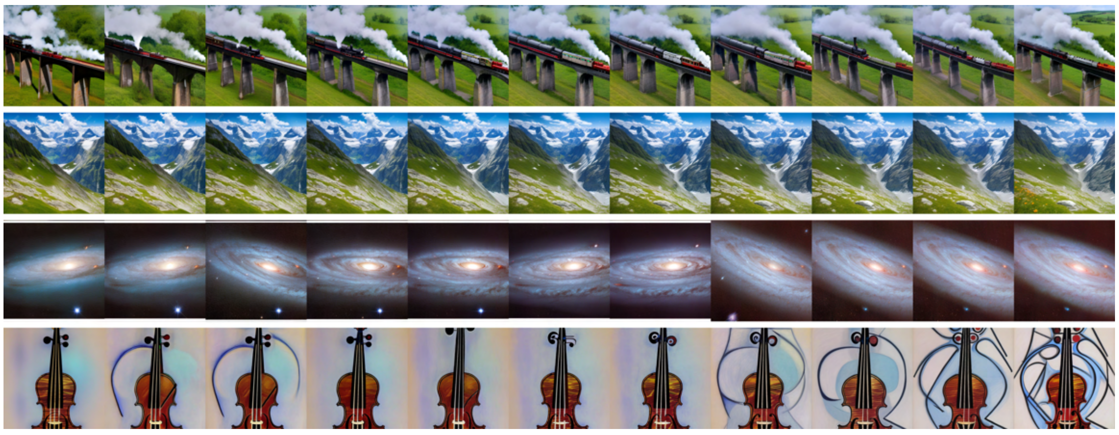
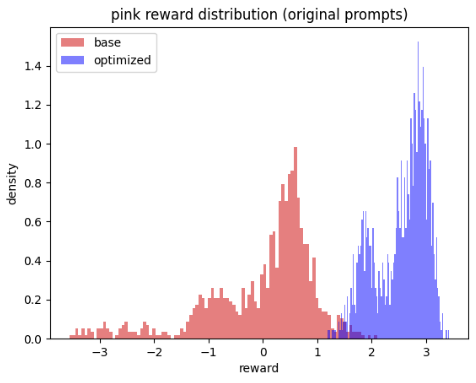
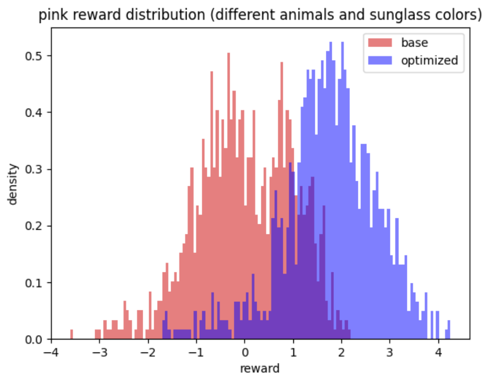

(e) Unrelated prompts.

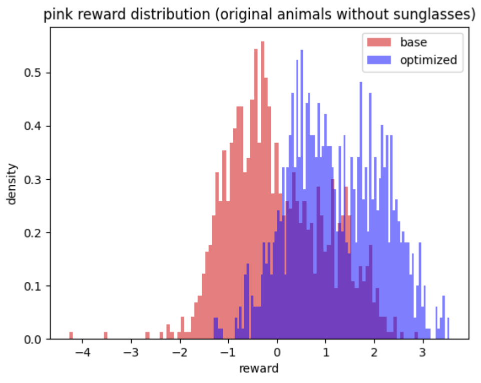
Figure 6.32.: Over-optimization progression of the *Pink* policy for different prompts showing different levels of overfitting. From the base model (left) until epoch 20,000 (right), at 2000 epoch increments.



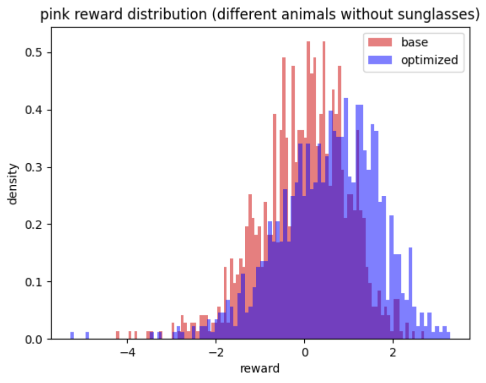
(a) Original prompts.



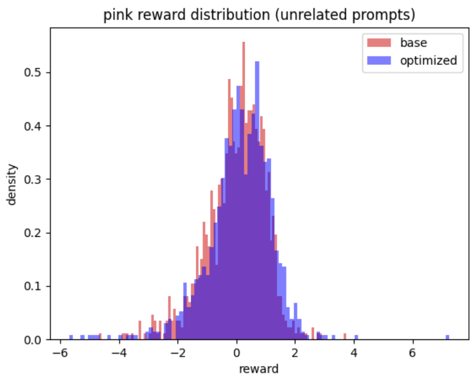
(b) Modified prompts with different animals and colored sunglasses.



(c) Modified prompts with original animals without sunglasses.



(d) Modified prompts with different animals without sunglasses.



(e) Unrelated prompts.

Figure 6.33.: Reward distributions for both the base and the *Pink*-optimized model using different prompts. The various animals and colored sunglasses used for generating the modified prompts are available in Appendix C. The histograms have been standardized around the reward distribution of the base diffusion model, giving it a mean reward of 0.0 and a standard deviation of 1.0.

6.10. Experiment 7: PickScore

The final experiment, dubbed *PickScore*, serves as a preliminary test for the application of the methodology to large-scale human-preference alignment of a state-of-the-art text-to-image model using a state-of-the-art reward model. The primary motivation was to show that the methodology was capable of scaling beyond the restricted domain of Experiment 6: *Pink*, thereby establishing it as a universally applicable method.

The experiment did not involve any new human preference collection nor reward model training but instead relied on an existing dataset and reward model from the concurrent work of Kirstain et al. (2023) (see paragraph 3.3 for more information). Thus, only *Step 4* of the methodology was tested.

6.10.1. Experimental Setup

The experimental setup closely resembled that of Experiment 6: *Pink*, but used the 37,523 unique prompts from the Pick-a-Pic v1[5] training dataset and the associated PickScore v1[6] reward model.

The only other difference to Experiment 6: *Pink* was the number of timesteps used, which was increased from 16 to 50 to produce better samples and to account for the increased complexity of the reward model, and the policy learning rate, which was increased from 1×10^{-7} to 5×10^{-7} for faster convergence. Slight modifications to minibatch size and gradient accumulation steps were also made to accommodate the increase in timesteps.

The rationale for a high number of timesteps was that if the human preferences captured by the reward model are sufficiently complex, they may present challenges for the policy's learning process, potentially requiring more training epochs to achieve a noticeable deviation from the original reward distribution. This could subsequently cause additional degradation of the base image distribution. By increasing the number of timesteps, we can ensure a more robust initial image distribution, which could help maintain the stability of the distribution for an extended training period.

As in Experiment 6: *Pink*, KL penalties were disabled for practical reasons, as they incur additional memory and time requirements and have been shown to be difficult to balance against the base rewards given by the reward model. The lack of such a regularization mechanism was unfortunate given the scale and nature of this experiment, but it was not a strict necessity to evaluate the potential of the methodology for large-scale alignment.

Training was stopped after 20,850 epochs.

[5]Available at: `https://huggingface.co/datasets/yuvalkirstain/pickapic_v1`.
[6]Available at: `https://huggingface.co/yuvalkirstain/PickScore_v1`.

6.10.2. Results

The training graph, presented in Figure 6.34, shows that the reward started increasing very early in training and at a rapid pace. It eventually flattened off before starting to decrease toward the end of training. The reward peaked at epoch 10,000 at around 1.2σ, which is a significant deviation from the base model.

Manual visual inspection of samples from every stage of the optimization process revealed that checkpoints around epochs 4,000 to 6,000 struck a balance between visible alignment and improved image samples without overdoing the effect and introducing excessive artifacting. However, the ideal checkpoint varies depending on the sampling settings and prompts used.

Diverse samples from the base policy and optimized model at epochs 4,000 and 20,000 are shown in Figure 6.35, Figure 6.36, and Figure 6.39, respectively. Samples showing a more restrictive prompt domain consisting of animals only from the base policy and optimized model at epochs 5,000 and 20,000 are presented in Figure 6.37, Figure 6.38, and Figure 6.40, respectively. The selection of text prompts used to generate the diverse and animal samples is provided in Appendix D.

Comparison and qualitative analysis of the base and optimized samples revealed that samples from optimized models exhibited a clear change in style and composition over base samples. The optimized samples appeared more realistic and natural, especially for animal samples. They displayed more consistent color grading, with colors being more muted and images gaining a bluish tint towards the end of optimization. Some samples changed more than others (e.g., the cherry blossom tree).

Samples from epoch 20,000 exhibited severe degradation, generating unnatural images that appeared “dreamlike” with unrealistic intricate details, often defying anatomy in the case of animals. The optimization process also resulted in a loss of diversity, with images converging on a centered subject and very similar composition regardless of prompt. Late-stage samples also demonstrated increased high-frequency information and striation-like patterns.

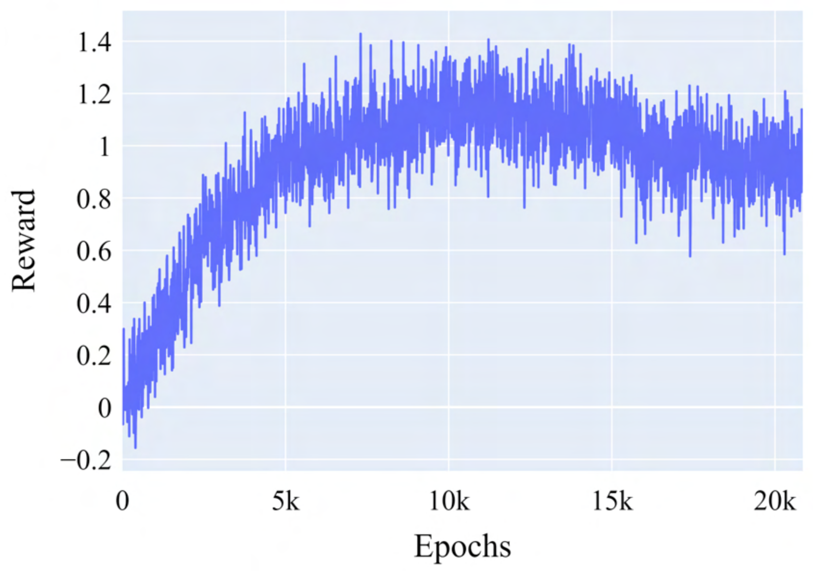


Figure 6.34.: Graph showing reward progression during *PickScore* experiment.

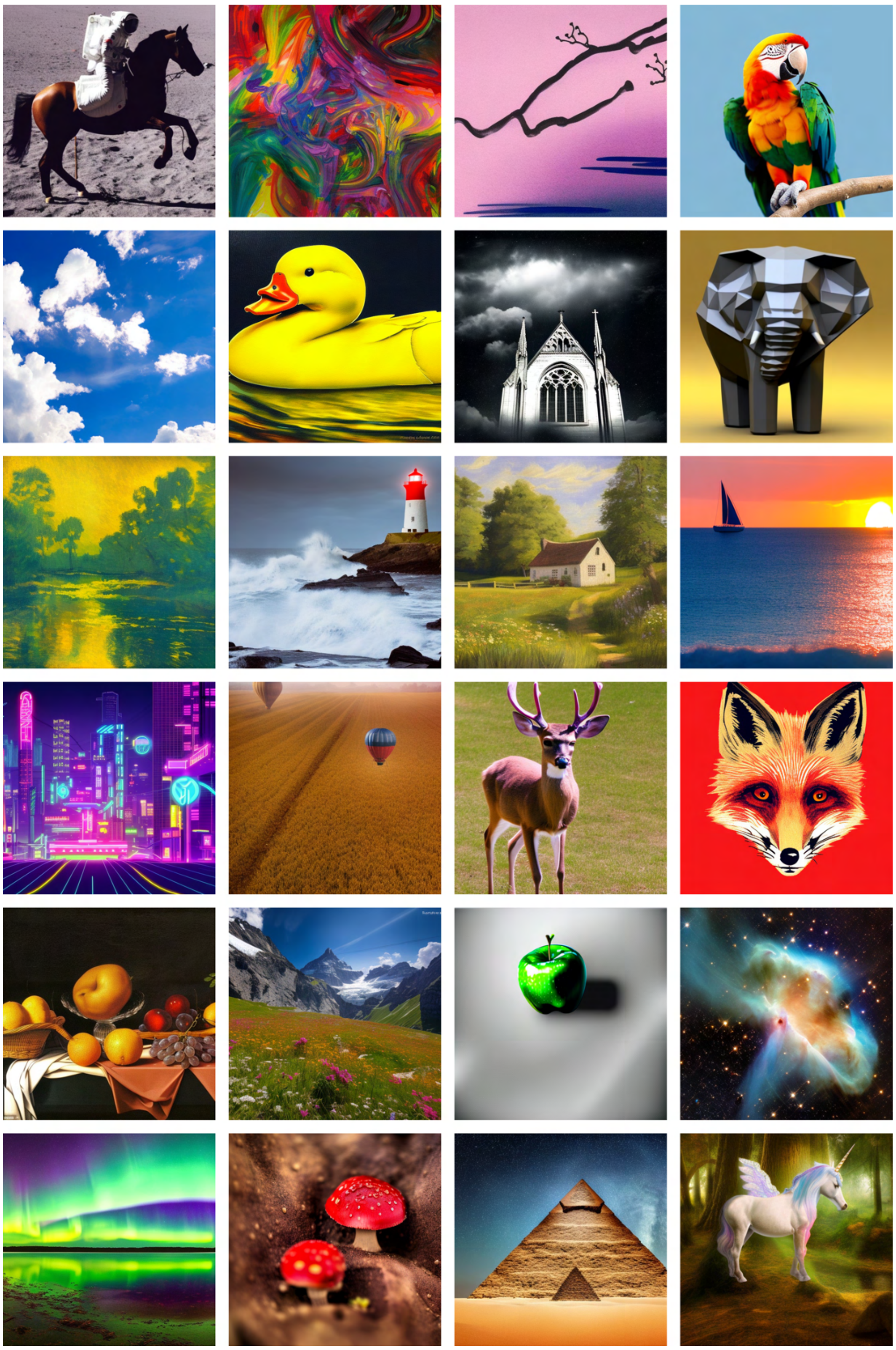


Figure 6.35.: Select diverse samples from the base model for the *PickScore* experiment.

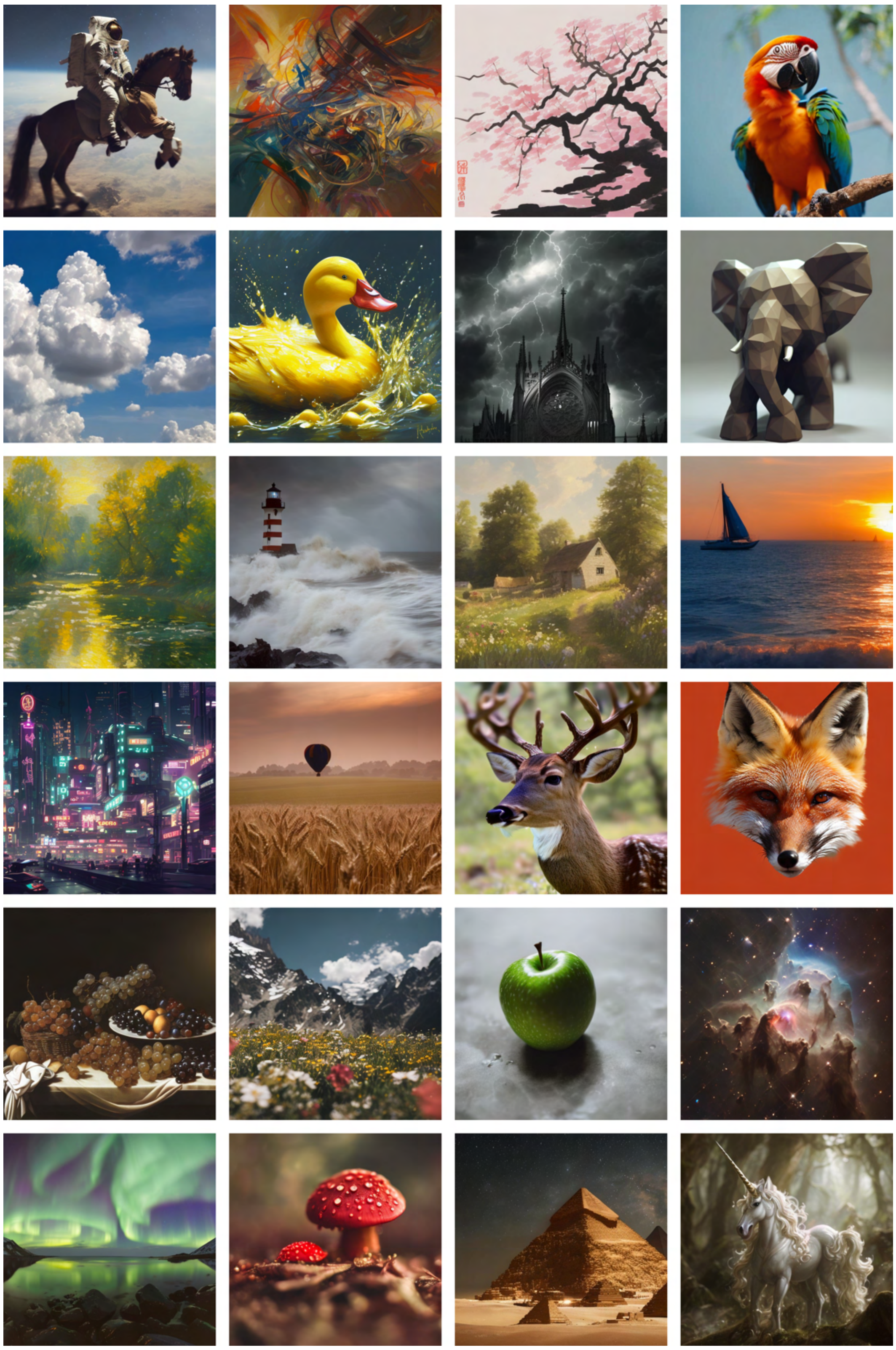


Figure 6.36.: Select diverse samples from the *PickScore* optimized model at epoch 4,000.

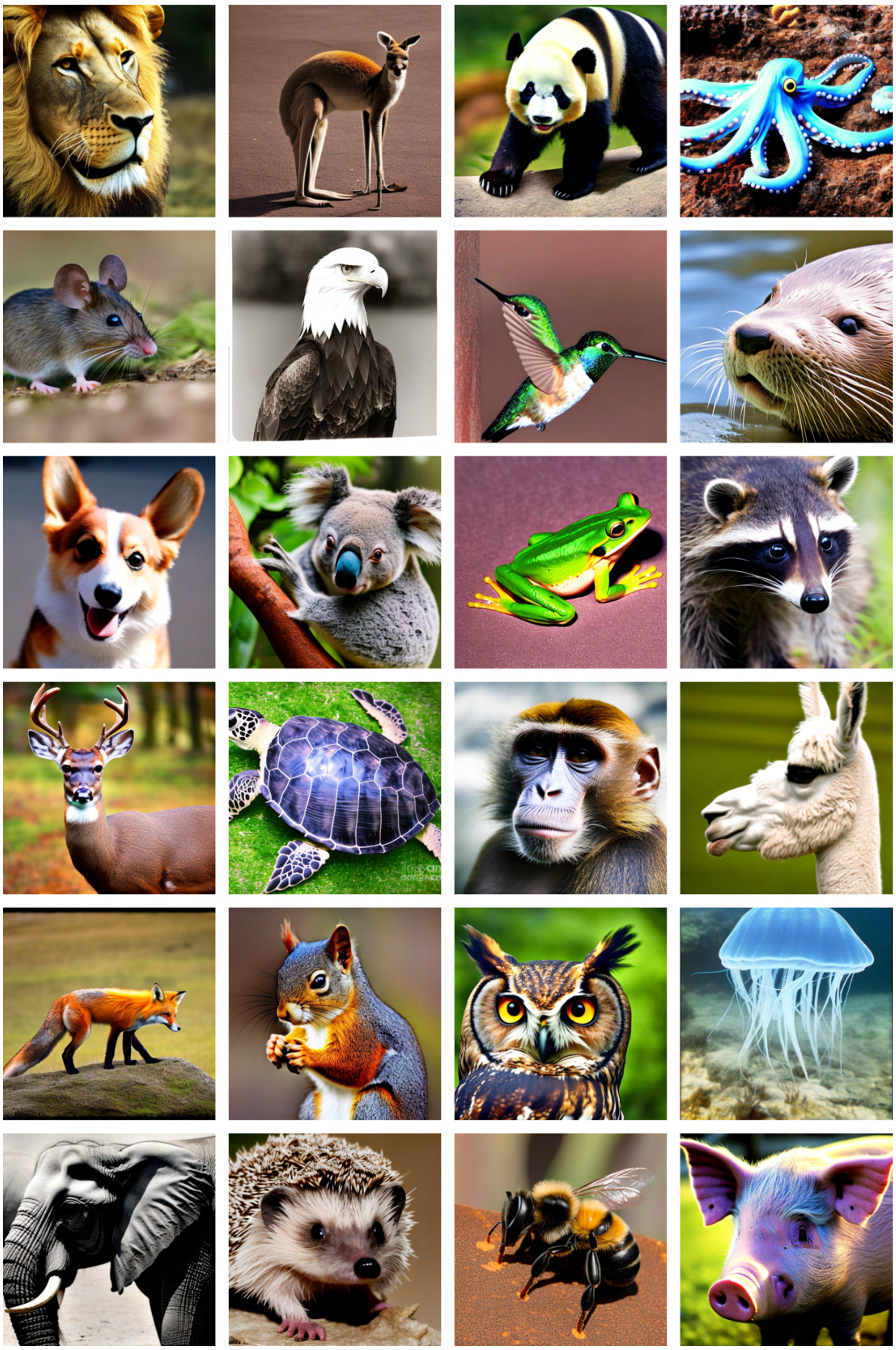


Figure 6.37.: Select animal samples from the base model for the *PickScore* experiment.

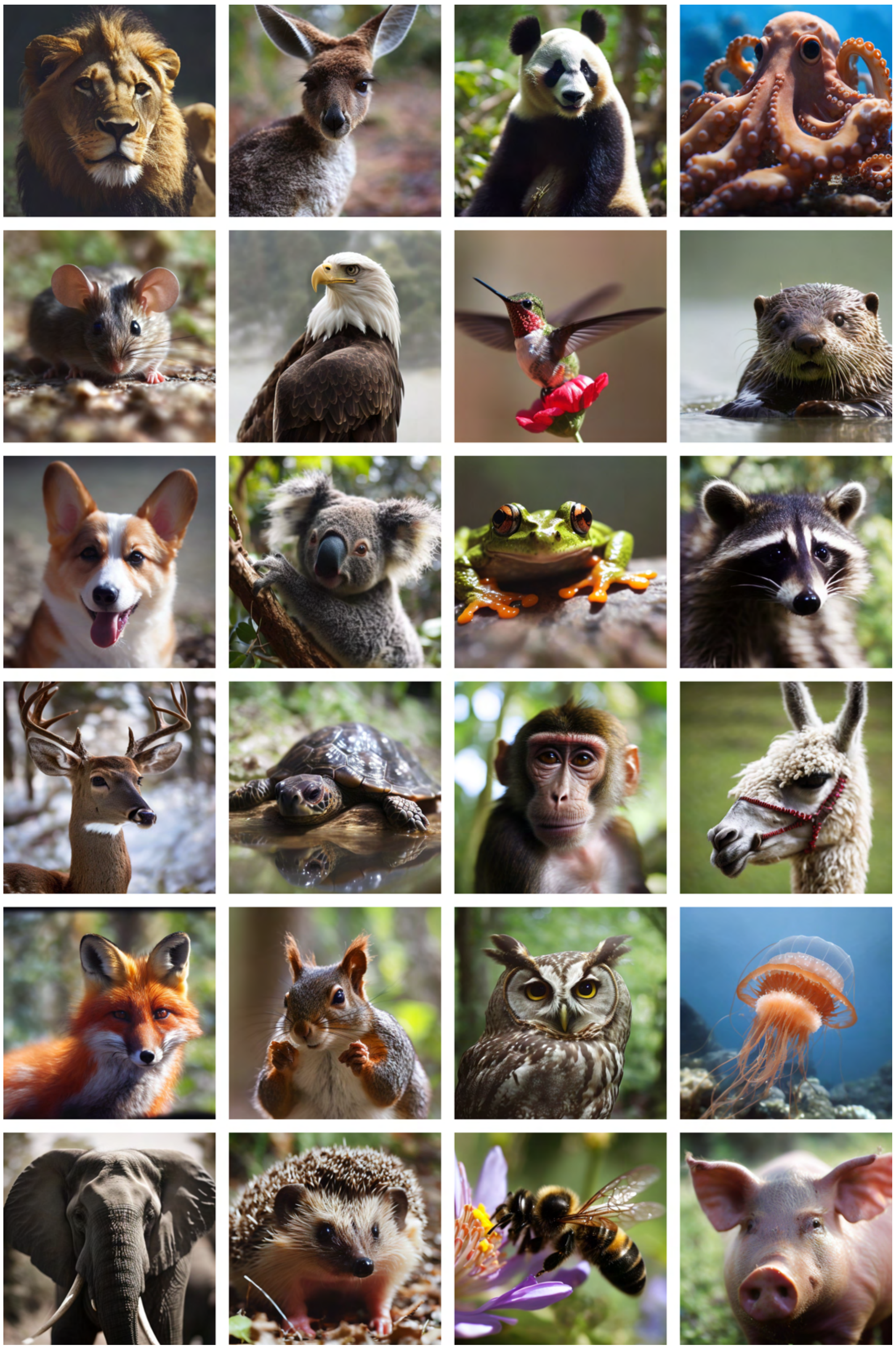


Figure 6.38.: Select animal samples from the *PickScore* optimized model at epoch 5,000.

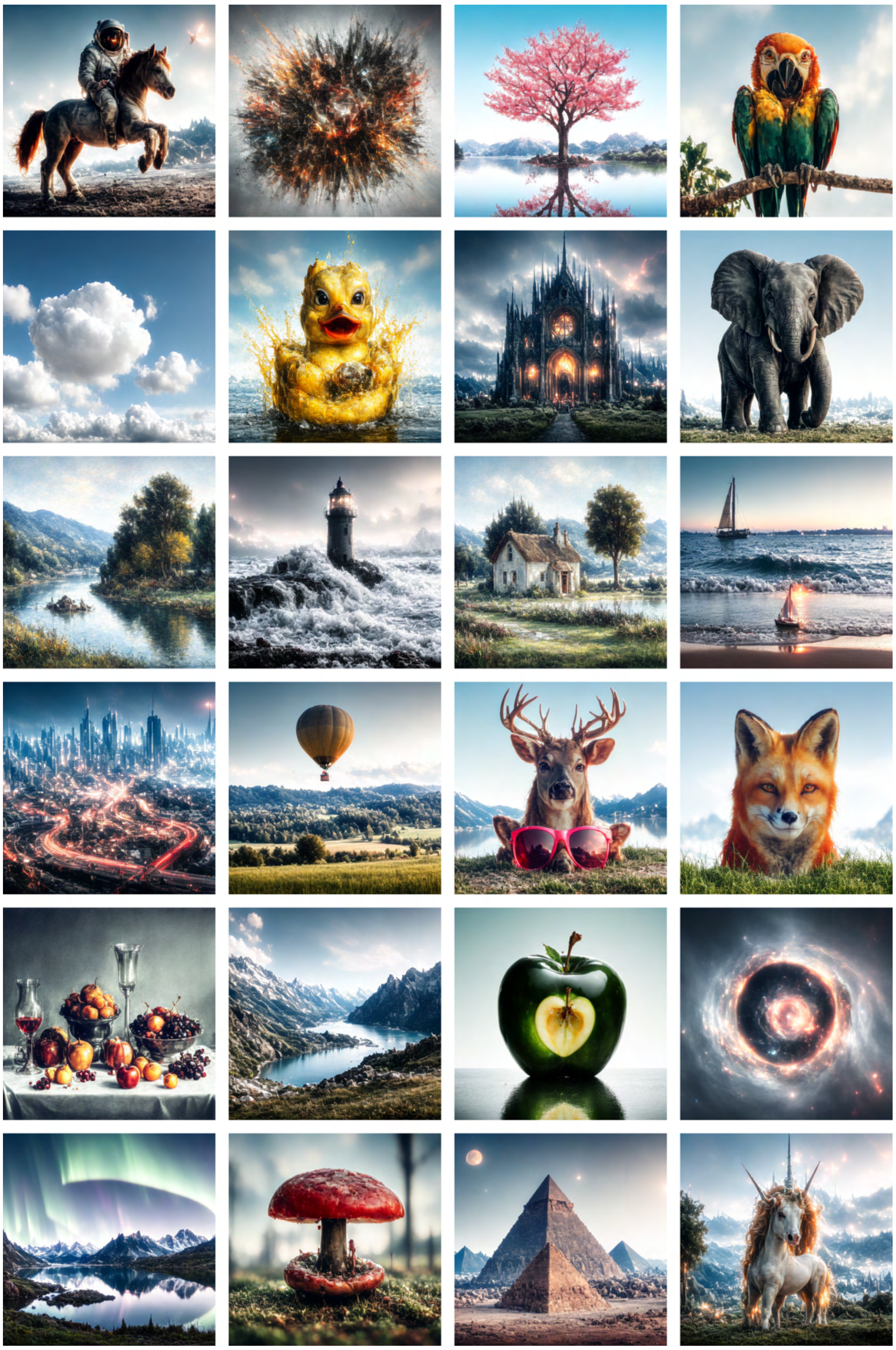


Figure 6.39.: Select diverse samples from the *PickScore* optimized model at epoch 20,000, exhibiting overt distribution degradation.

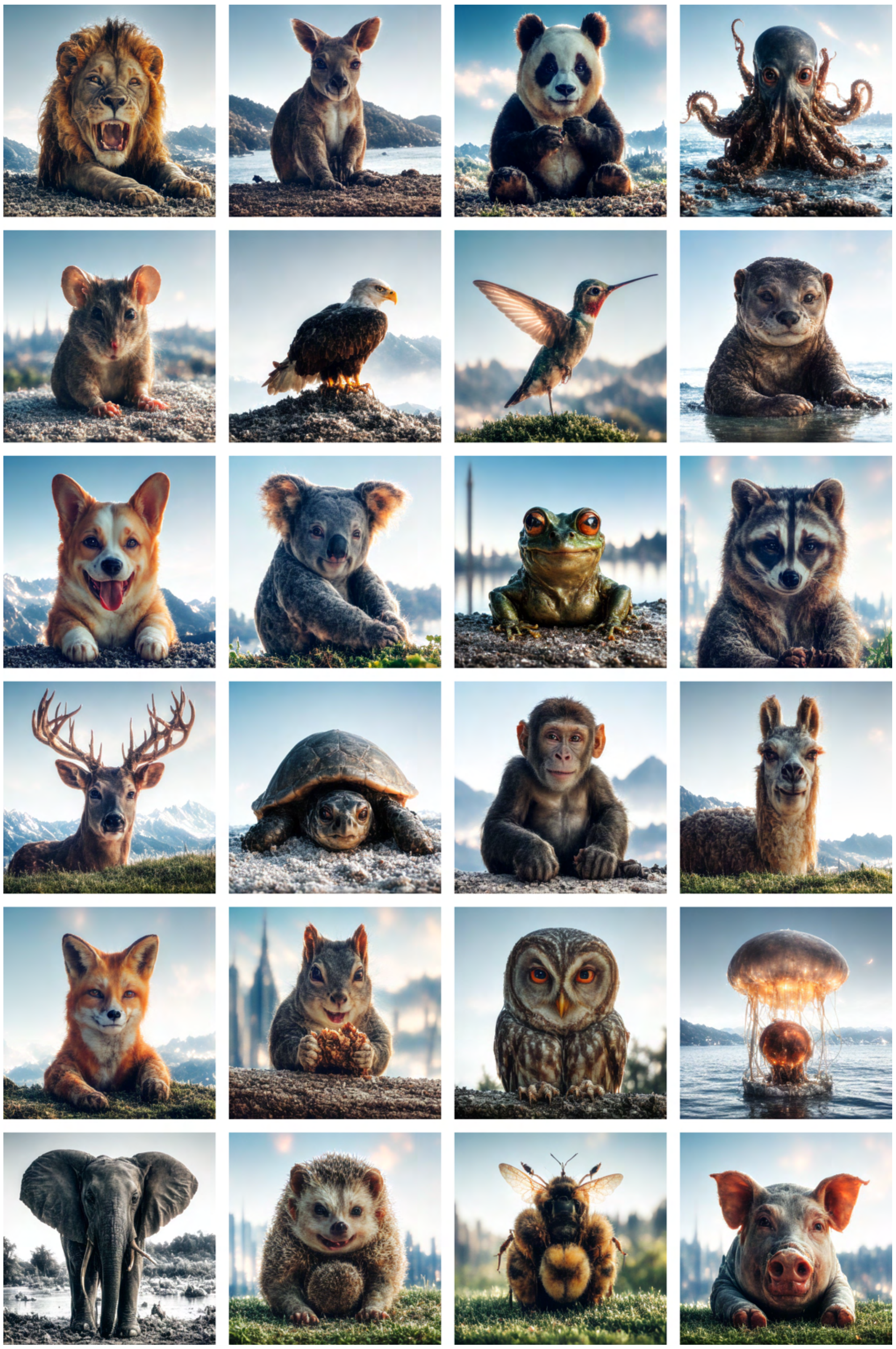


Figure 6.40.: Select animal samples from the *PickScore* optimized model at epoch 20,000, exhibiting clear loss of diversity.

6.10.3. Evaluation

Judging by the reward graph alone, the experiment seems to have been successful at aligning the policy to human preferences. However, the samples show that a high reward does not necessarily equal good-quality images, as the samples with the highest reward would likely not be preferred by humans. For example, images sampled at epoch 20,000 had higher rewards than those sampled at epoch 4,000 but appeared much worse on visual inspection. This result indicates reward model exploitation and highlights the need for a more robust reward model to properly model general human preferences.

The best samples were observed around epochs 4,000 to 6,000, which is before the reward peak. Samples after these epochs show clear signs of degradation. Consequently, there is little point in optimizing past this point with the current setup. This signifies that achieving desirable optimization is feasible within a few thousand epochs.

Without conducting a proper human evaluation trail, it is difficult to gauge whether the optimized policy genuinely generates images that are generally preferred by humans, particularly when assessing improvement or regression in image-text alignment. However, the provided samples give a good indication of the potential of the methodology to align text-to-image models against general human preferences.

It is somewhat surprising how effectively the policy managed to increase the reward considering that PickScore models general human preferences, which are considered challenging to grasp and model. It is possible that the policy achieves this by focusing its attention on certain optimization axes, such as general image brightness and detail level. By exploiting enough of these axes in parallel, the policy may be able to boost the reward significantly.

Since there was no KL regularization, the model was free to optimize the reward at any cost. This lack of constraint can have unknown consequences for the model distribution. Among the observed consequences were the loss of diversity and realism as optimization progressed. Another potentially concerning observation was that since diversity decreased, the images become biased towards the most preferred image features for a specific class. As an example, which was not reported in the results above, it was observed during training that women/men would converge on what the reward model considered an “ideal” woman/man, with certain hair types, facial features, skin colors, and clothes. In many cases, this would be considered unwanted behavior, and thus, it warrants further research into creating robust and bias-free reward models and optimization methods that can better maintain diversity.

In conclusion, this experiment served as a proof of concept for the methodology’s ability to scale beyond the restricted domain of previous experiments and demonstrated its potential as a universally applicable method. Although the experiment achieved decent results in aligning the policy to human preferences, there were indications of reward model exploitation and loss of diversity in the generated samples. This highlights the need for further research into creating robust and bias-free reward models, as well as optimization methods that can maintain diversity better. Conducting a proper human evaluation would be necessary to further validate the results and the effectiveness of the methodology for large-scale alignment.

7. Discussion

The goal of this thesis was to develop an approach to align diffusion-based text-to-image models using Reinforcement Learning from Human Feedback (RLHF) for generating high-fidelity and semantically consistent images. To accomplish this goal, we conducted a series of experiments at increasing scale and complexity, gradually incorporating human preferences to guide the optimization of simple diffusion models and, ultimately, a state-of-the-art text-to-image model.

This chapter discusses the experiment series presented in chapter 6, revisits the research questions posed in chapter 1, and reflects on the implications and limitations of the proposed methodology for aligning diffusion models with human preferences. We also briefly mention and discuss some recent concurrent work. Lastly, we provide a collection of reflections on the work conducted.

7.1. Experiment Series Review

All experiments demonstrated a considerable increase in reward after optimization. However, this increase in reward almost always came at the cost of diversity. The experiments demonstrated that the RLHF methodology can optimize using very few timesteps and that it generalizes to more timesteps during later sampling. This greatly increases the efficiency of the method. The experiments also demonstrated positive reward transfer to other classes and prompt not seen during optimization.

However, it proved to be difficult to balance the parameters to maximize the reward while minimizing model degradation. The experiments showed that KL penalties did not prove to be very successful. The KL penalty value would rise exponentially throughout training and at a critical point, it would jump from being insignificant or only slightly significant to being around 100 % of the total reward. At that point, no further training would be fruitful, as the policy quickly diverged too far from the base model. Manually adjusting the KL penalty coefficient would not have helped, as the penalty increased too rapidly. In conclusion, the experiments were not able to tame the KL mechanism, as it was static. The coefficients we tried were those suggested by Ouyang et al. (2022), but those values seem to be too high in our case, because of the differences between reinforcement learning with diffusion models and Large Language Models (LLMs). Proper ablation studies for this sort of thing are needed, as well as the introduction of more adaptive penalty schemes.

The decision to use a very conservative learning rate throughout the series was primarily based on the nature of the optimization process, which doesn't incorporate any new training data. Instead, the model generates its own training data during rollout. The optimization process is therefore lossy with regard to retaining the original data distribution; there is no direct method for the model to regain any potential conceptual knowledge lost during optimization unless that information is somehow encoded in the reward signal. Hence, for the initial experiments, a lower learning rate was chosen to preserve as much of the original distribution and learned representation as possible.

To facilitate easier transfer of hyperparameter values between experiments, the reward function should have been standardized earlier in the development process. This would have required additional time for sampling and computing rewards from a representative distribution before optimization, which was not prioritized until the methodology had proved itself successful enough.

The experiments also encountered difficulties with distributed training using Distributed Data Parallel (DDP) and mixed precision. DDP had considerable overhead in the beginning, which led to several complete refactors of the codebase, taking a great deal of effort. This was exacerbated by the experiments' unfamiliarity with DDP. Getting Experiment 6: *Pink* and Experiment 7: *PickScore* to work in a timely fashion was also critical, which necessitated the use of distributed training.

In summary, the experiments demonstrated the viability of the RLHF methodology for aligning diffusion models with human preferences. However, they also highlighted challenges in balancing optimization objectives and maintaining model fidelity, as well as difficulties in hyperparameter tuning and evaluating model performance.

7.2. Overview of Results

The experiment series demonstrated the effectiveness of the RLHF methodology for aligning both simple diffusion models and state-of-the-art text-to-image models. The following summarizes the key findings of the experiments:

- Experiments 1 and 2 showed that simple diffusion models can be successfully aligned with simple fixed reward functions, indicating that diffusion models are capable of being trained using reinforcement learning techniques.
- Experiment 3 highlighted challenges related to mismatches between reward models and the image distributions of the diffusion policies being optimized. This mismatch necessitates careful hyperparameter tuning and regularization.
- Experiments 4 to 6 validated the reward modeling components of the methodology. They demonstrated that diffusion policies could learn human preferences captured by reward models based on U-Net encoders.
- Experiments 6 and 7 established the scalability of the methodology to complex

state-of-the-art text-to-image models. They revealed issues like reward exploitation, policy collapse, and potential biases in reward models trained on large datasets.

Taken together, the series progressively validated and improved upon the proposed RLHF methodology, culminating in its application to align human preferences with the state-of-the-art Stable Diffusion model. The series also uncovered many important insights into the challenges of aligning generative models, which inform the following discussion.

7.3. Revisiting the Research Questions

The key findings and lessons learned based on the experiment series help address the three research questions posed in chapter 1.

RQ1: Mapping Diffusion Models to a Reinforcement Environment

Experiments 1 and 2 established that diffusion models can be mapped to Markov Decision Processes (MDPs) and optimized using policy gradient methods. These experiments demonstrated that diffusion states, actions, and rewards could be defined as in section 5.1, enabling the application of reinforcement learning. The main requirement found for successful optimization was a reasonable number of diffusion timesteps for sampling images, which ensured sufficient efficiency and quality to receive meaningful gradients from rewards. Overall, all experiments validated the formulation and general robustness of the proposed reinforcement learning environment for diffusion models.

RQ2: Designing Reward Models

The experiment series highlighted both the importance and challenges of designing effective reward models. Reward model mismatches with diffusion policies (section 6.6) necessitated hyperparameter tuning and KL regularization to avoid rapid divergence or degradation. Integrating human preferences via feedback also required thoughtful modeling decisions, such as splitting feedback into comparison tasks (chapter 4). Experiments 4 and 6 demonstrated that reward models initialized from diffusion model encoders and fine-tuned on human feedback could successfully capture preferences, enabling policy optimization (chapter 5). However, experiments 6 and 7 revealed issues like reward model exploitation and potential bias in models trained on large datasets (section 6.9, section 6.10), motivating future work on robust reward modeling.

RQ3: Evaluating Alignment Performance

Different metrics were used to evaluate alignment performance in the experiments. Quantitative metrics like training loss and reward histograms showed that optimized policies achieved higher rewards according to the modeled preferences. However, samples also needed to be qualitatively assessed to determine if high rewards indeed implied better alignment. Visual inspection revealed issues like policy collapse and degradation in several experiments. Human evaluations of generated samples and comparison tasks emerged as the most reliable evaluation method. Overall, a combination of quantitative metrics, qualitative assessments, and human evaluations was found to be most effective.

7.4. Implications of the Work

The experiment series established the potential viability and impact of the proposed RLHF methodology for aligning diffusion models with complex human preferences. Some key implications are:

- The methodology enables new ways to optimize the behavior of text-to-image models beyond initial training objectives. This optimization can improve user satisfaction, trust, and perception of model capabilities.
- RLHF techniques can enable text-to-image models to learn from limited human feedback and guidance, reducing annotation costs compared to supervised training.
- The generalization capabilities observed in experiments 4 and 6 indicate that RLHF may enable the learning of high-level preferences that transfer across domains and prompts.
- By aligning generated images with human preferences, the methodology could potentially make text-to-image models safer, more ethical, and capable of producing less harmful outputs.

While most experiments focused on relatively simple preferences, they demonstrated the promise of the methodology for aligning diffusion models in real-world settings. With further refinements to reward modeling and regularization, the approach could enable substantial progress in the capabilities and impact of text-to-image generation systems.

7.5. Limitations of the Work

Several limitations are evident from the experiment series, representing opportunities for future improvement:

- Reward models are difficult to design and often exploit weaknesses to provide rewards that do not actually correspond to human preferences. More sophisticated reward modeling schemes are needed.
- Policy optimization led to issues like degradation, collapse, and loss of diversity. Stronger regularization techniques are required to mitigate these problems.

- The methodology places high demands on computational resources, especially for large models. More efficient optimization algorithms and state representations are necessary.
- Human evaluations are required to confirm that generated images actually align with human preferences, but are time-consuming and expensive to obtain at scale.
- The approach does not yet consider potential harms and ethical issues that could arise from optimized generative models.
- The methodology was only evaluated for a small number of reward models. Evaluating the approach for aligning to a wide range of complex preferences remains an open challenge.

Overall, the experiment series demonstrated promising initial progress toward aligning diffusion models through RLHF. However, numerous technical and practical challenges remain in scaling the methodology for real-world impact. Future work will need to confront these limitations head-on if this approach is to realize its potential for generating images that genuinely align with complex human preferences.

7.6. Ethical Considerations

The ability to align generative models with human preferences raises important ethical considerations. Some potential ethical issues include:

- **Biased preferences:** Reward models trained on data from a specific population may exhibit biases that reflect that population's preferences. Optimized generative models can perpetuate such biases. We observed hints of this in experiment 7.
- **Privacy concerns:** Preference data used to train reward models may contain private information that must be properly anonymized and secured.
- **Diversity issues:** Optimized generative models may become aligned with preferences so narrowly that their outputs lack sufficient diversity or creative autonomy.

Addressing these ethical considerations will require careful thinking about how human preferences are modeled and incorporated, as well as checks and safeguards that can be integrated into the methodology itself. Future research on aligning generative models must prioritize ethical considerations from the outset to ensure that this promising technology is developed responsibly.

7.7. Comparison to Concurrent Work

Two recent concurrent works propose methodologies that are closely related to ours. Their findings further reinforce the viability of our independently developed, but similar, methodologies. The two papers are titled *Training Diffusion Models with Reinforcement*

Learning by Black et al. (2023) and *DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models* by Fan et al. (2023). The authors coin the terms *Denoising Diffusion Policy Optimization (DDPO)* and *Diffusion Policy Optimization with KL regularization (DPOK)* to refer to their respective methods; we do not offer an alternative term.

All our methods have in common that we frame the reverse diffusion process as a Markov decision process. Our method differs from that of Black et al. (2023) in that they accumulate gradients over the full rollout trajectory, instead of treating each state in the rollout independently. Thus, all states pertaining to the same trajectory are always part of the same optimization update, instead of being split into different minibatches. This is more consistent with the diffusion theory and previous RLHF efforts in LLMs, but we are not sure if this makes any meaningful difference to the optimization process. Furthermore, Black et al. (2023) does not make use of KL regularization while Fan et al. (2023) and our method does. While we struggled to get KL penalties to work properly, Fan et al. (2023) suggest that they are essential for proper alignment.

Similar to our experimental approach, Black et al. (2023), devise simple functional reward models to verify the reinforcement learning scheme. They also test using the same Aesthetic predictor as us, but using Stable Diffusion. They observed similar results, where the images become simplified with a single-color background with the subject in focus.

Fan et al. (2023) test using basic prompts similar to our experiment involving animals wearing pink sunglasses. They also test using the ImageReward model, but do not provide many samples for analysis.

7.8. Further Reflections

This section provides further reflections on the work conducted in this thesis. These reflections relate to our proposed methodology, the experimental series conducted, applications of our method, what we have learned in the process, and what we would do differently if we were to do it again. This section should be helpful for guiding researchers interested in recreating our method, and to guide future work in the field.

7.8.1. Human Preference Collection Format

We started off by wanting to capture and model general human preferences, inspired by Ziegler et al. (2019)'s work on preference-tuning LLMs, which is why we decided to adopt their labeling format of ranking samples based on preference.

However, while we adapted their human preference collection methodology, we never had the time to conduct a large-scale general human preference collection experiment as we initially wanted. Instead, we opted for simpler experiments designed to prove our methodology, similar to the experiments conducted by Lee et al. (2023), as the closest work to ours in the text-to-image domain. While they also perform preference alignment, their definition of “preference” differs slightly from Ziegler et al.’s in that they primarily look at objective tasks like image-text alignment (correct counting, color binding, backgrounds, etc.), while Ziegler et al. (2019) considers “preference” to include a mix of both text alignment and fidelity.

Because of this, the ranking system which we had adapted to capture subjective, general human preferences, was now being used on more objective tasks, meaning there were instances where “objectively” poor images were rewarded simply because they were superior to other “objectively” poor images. Ideally, both images would be labeled as poor to avoid confusing the reward model.

OpenAI (2023), which builds on Ziegler et al. (2019), combines ranking with auxiliary tasks to tackle this problem. By gathering additional data on toxicity and harmfulness, they can penalize unwanted samples regardless of where they are ranked. While this was not a problem of significance for Experiment 4: *Macaw*, where we primarily valued image fidelity due to the non-complex task, a human preference collection format like this would have enabled us to handle “objectively” poor images a lot better for Experiment 6: *Pink*, where poor images were a lot more common.

Consequently, we believe that our methodology still holds a lot of unexplored potential. And while we have been unable to extensively test this preference collection format, due primarily to our simple experiments, we have no reason to believe that it wouldn’t also work well for general preference collection in the text-to-image domain.

7.8.2. Software

There were numerous shortcomings of our software, some of which we outline below:

Cyclic Labeling. Our labeling software presents the labeler with two and two images within a task, until all images within the task have been compared against one another (in the case that we assume transitivity, some of the images are automatically compared), creating a total ordering within that task. However, by naively comparing every image against one another, cycle-comparisons can appear within the task, breaking the total ordering. This limitation still holds when labeling in transitivity mode, but here such potential cycles are effectively broken by which comparison the labeler is presented with first, which is inherently random.

By only presenting the labeler with two images at a time, our labeling software enables inconsistent human labeling. To combat this, the software would ideally show the entire ordering of images as the labeler annotates the task, naturally forcing the human labeler to break cycles by reconsidering past comparisons instead of arbitrarily breaking them in software.

By the time we realized this, we had already fully developed our labeling software. And because we didn't consider a complete redesign of the software cost effective, we instead tried to naturally guide the labeler towards acyclic labeling by constructing instructions that promote transitivity. We believe it also helped that we were doing the human preference labeling ourselves, since we were already aware of potential cyclic labeling by the time we were performing our larger experiments and could make sure we only labeled data when we could guarantee a level of consistency.

Ties. Our labeling software also assumes that each task makes a total order. This leaves no room for ties if the human labeler believes both presented images are of equal preference. The immediate downside of being unable to declare ties, is how the loss function of our method works. Under training of the reward model, the model tries to maximize the reward difference between the two images that make up a comparison. If the human labeler believes two images to be of equal preference, but is forced to prefer one over the other, it can lead to noisy reward model training as the model tries to maximize their reward difference.

Forcing the labeler to decide a preference among two images in the case of indecision can also slow down labeling. This is something we experienced while labeling for our experiments. The pressure to label as accurately as possible would often lead us to slow down considerably as images of similar quality were presented. Ideally, the human labeler would just be able to select "tie" if the presented images were of sufficiently similar quality. A tied comparison could then either be excluded from the reward maximization in the optimization step, or included in a separate reward minimization step (though this approach would require further engineering work).

Mass Preference Collection. Our labeling software was purposely designed to collect human preferences from "expert" annotators. Relying on data from annotators can be slow and tedious, and easily fall prey to inconsistent labeling from demotivated human labelers (Ziegler et al., 2019; Kirstain et al., 2023). When trying to capture general human preference, you would also ideally try to collect preferences from as many labelers as possible to override any bias not inherently related to the human preference you are trying to capture. Text-to-image services like Midjourney[1], Stable Diffusion's DreamStudio[2] and Pick-a-Pic (Kirstain et al., 2023) use online preference collection to expose their data gathering to a larger group of intrinsically motivated users that can generate their own images. This allows them to build a large dataset of real preferences based on real-world usage of their models.

Others, like Wu et al. (2023), scrape user preference data from Stable Diffusion's online generation forums to gather the same data. While this might have been possible for us too, it was not something we thought of at the time. However, in accordance with our

[1]Available at: `https://www.midjourney.com/`.
[2]Available at: `https://beta.dreamstudio.ai/`.

methodology, scraping the data might not be as effective as running the mass preference collection services themselves. Without the exact knowledge of models and settings used to generate the images, it might be difficult to get the rollout of the policy optimization stage as similar as possible for optimal efficiency. Though to the extent that this is actually needed when building a general preference model is something we have not had the time to test. Regardless, the increase in high-quality data would most likely outweigh this downside and some form of mass preference collection might be needed to gather enough data to accurately capture general human preference.

7.8.3. Experimental Plan.

Unlike Pick-a-Pic (Kirstain et al., 2023), which attempts to capture general human preferences over a wide range of use cases and labelers, we ended up attempting to capture specialized human preferences over a narrow range of use cases and for only one labeler in our main experiments. We also limit our feedback to simple image-to-image comparisons, unlike ImageReward (Xu et al., 2023), which gathers auxiliary data like toxicity and harmfulness to better model a desired behavior.

Our methodology was designed to capture general human preferences, and that is what we intended to do. However, because each experiment took so long to run, including data gathering and model training, and the uncertainty surrounding how much data was needed to capture general human preferences or if our methodology even worked at all, we decided to limit the scope of our experiments. We found it more important to show that the method worked at all than it was to show its full capabilities, on the off-chance that we had gathered too little data or that we proceeded with unknown bugs that prevented it from learning anything.

That said, we believe we have been able to show that our methodology is capable of collecting human preferences and training a reward model that captures these preferences, despite the limited scope of our experiments. We therefore see no reason why our methodology shouldn't also be able to collect general human preferences and train a more general reward model in future, larger experiments.

7.8.4. Reward Model Architecture and Training

While we do not share parameters between models in our experiments, it is possible to do so for the encoder part of the U-Net in the policy, reward model, and value function. This is more memory efficient and could improve model performance by keeping the reward model and value function aligned with the policy, as the policy is fine-tuned. Ziegler et al. (2019) proposed that this approach might help maintain a sufficiently robust reward model that precludes policy exploitation, although they were unable to provide empirical evidence in support of this. Moreover, given that the value function in our diffusion-based framework is required to evaluate intermediate noisy image samples with significant variations in perceptual content, unlike the reward model, parameter sharing for these components seems less logical.

While we initialize the value function from the reward model in our experiments, we conjecture that it could be beneficial to pre-train the value function using the same comparison data used for training the reward model, while introducing noise back into the image samples through forward diffusion (see background in section 2.5). This is similar to the training strategy employed by Dhariwal and Nichol (2021) for their guidance classifiers. However, in our experiments, we observe that the value loss converges early in training, suggesting that the current initialization strategy may already be near-optimal. We leave further exploration of this to future work.

7.8.5. Alternative Reward Models

While our proposed reward model focuses on aligning diffusion-based text-to-image models using human preferences, alternative reward models can also be considered for optimizing diffusion policies against different types of objectives. In this section, we discuss some alternative reward models and their potential use cases. These models represent potential future research directions.

Pre-defined Functional Reward Model. A pre-defined functional reward model can be used to assign higher rewards to images with specific characteristics, such as the colorfulness reward used in Experiment 2: *Colorfulness*. However, such an approach could be problematic if the objective does not take into consideration the text conditioning. In this case, the model might optimize toward the reward model in an undesirable manner by forgetting to pay attention to the text, and hence, the text-related network parameters might be negatively affected. To mitigate this, the KL regularization would have to be sufficiently strong to keep the model from exploiting the reward model too much.

Improving Likelihood of Original Training Dataset. Inspired by Kim et al. (2022), in their work on "Refining Generative Process with Discriminator Guidance in Score-based Diffusion Models," we can train a reward model as a discriminator to discriminate between images from the original dataset and images generated by the model. Optimizing the model against this discriminator should yield improved generative capabilities. This approach aims to "close the gap" between original and synthetic data, thereby improving metrics such as the Fréchet Inception Distance (FID) score (see Background in subsection 2.12.2).

Sampling Distillation. Sampling distillation involves training a reward model as a discriminator to discriminate between high-quality image samples (e.g., 250 sampling steps) and low-quality image samples (e.g., 25 sampling steps). By optimizing the diffusion model using the discriminator, we can produce samples of similar quality to those generated using 250 steps with just 25 steps, ultimately saving on inference time.

Improved Text Rendering. In scenarios where the text prompt involves generating images that include text, a reward model could utilize a text detection or extraction model to evaluate the generated images. This model would provide positive rewards when the generated text in the image matches the expected text, and negative rewards when the generated text does not match, is inaccurate, contains spelling errors, grammatical errors, weird letter shapes, etc. This approach encourages the generation of legible and correct text, which can be particularly useful in applications such as graphic design or logo creation, where accurate text representation is vital. However, integrating a text detection or extraction model can increase the complexity of the model and may require additional training data where text is clearly and accurately labeled within images.

However, such a reward model presents some challenges. First, it would require sophisticated and reliable text detection and recognition capabilities for effective discrimination and reward assignment, in combination with sufficient training data (i.e., prompts for rollout) to capture all potential ways of representing text in images. The inherent subjectivity and variability in text aesthetics (e.g., fonts, sizes, colors, orientations, etc.) also necessitate this. Moreover, as it is specifically tuned for scenarios involving text in images, it might not perform optimally when faced with text prompts that don't involve the generation of text within the image. It could also inadvertently bias the diffusion model to overemphasize text elements in image generation, potentially at the cost of other important visual elements. Additionally, the text recognition model itself could make errors, leading to inappropriate rewards and potentially reducing model performance. Lastly, text rendering in an image is a complex task that might need more specialized treatment than just reward modulation. For example, Saharia et al. (2022a) demonstrated that the degree to which the text-to-image model is capable of rendering text mostly comes down to the text encoder employed, and not the generative image model itself, as one might initially believe.

Multi-objective Reward Models. Multi-objective reward models can assign rewards based on multiple criteria simultaneously, rather than optimizing for a single objective. These models can balance different desired outcomes, such as image quality, alignment with text prompts, and specific image features like colorfulness or text rendering. This concept is similar to the approach used by OpenAI in GPT-4, which employs several small rule-based reward models which provide additional rewards during RLHF fine-tuning (OpenAI, 2023). These individual reward models address specific areas such as detecting harmful or inappropriate content, rambling, and deviations from a desired style.

One approach to implementing such a model could involve assigning weights to the different objectives and combining their rewards linearly. However, determining the appropriate weighting scheme can be a challenging task and might need to be adapted dynamically as the model learns. An alternative approach for multi-objective optimization could be the use of Pareto efficiency or Pareto optimality, which allows for the balancing of multiple conflicting objectives without explicitly defining a weighting scheme (Team, 2022). Pareto efficiency considers a solution to be optimal if there's no way to make a situation better for one objective without making it worse for another.

When considering multi-objective reward models, it's worth noting that there may be a trade-off between the complexity of the model and the quality of the output. Increasing the number of objectives might lead to a more nuanced and balanced image generation model, but it could also make the model more complex and potentially more difficult to train. Moreover, multiple objectives could potentially conflict with one another, making it challenging to find an optimal solution that satisfies all criteria simultaneously.

7.8.6. Evaluating the performance and alignment of diffusion models trained with RLHF

In this work, the performance and alignment of these models are explored using two primary evaluation methods: automatic evaluation and human evaluation, which provide quantitative and qualitative insights, respectively. Automatic evaluation is a convenient way to monitor the development of models as they are trained and to identify any potential issues. In contrast, human evaluation provides a more accurate method for comparing models and qualitatively assessing its ability to capture general preferences.

Automatic Evaluation

This thesis introduces four key metrics used for automatic evaluation: loss, reward, Kullback-Leibler (KL) divergence, and Fréchet Inception Distance (FID) score. These metrics provide a convenient way to monitor the development of models during training and identify potential issues. Each of these components aims to offer a different perspective on model performance.

Loss

Loss is often the primary means of evaluating model training, as it is the loss the model tries to minimize during the learning process. However, while a decrease in loss indicates that the model is learning, it does not reveal *what* the model is learning or whether what it is learning aligns with the intended objectives. As such, the loss graph is a good first indicator of model improvement, but other metrics are necessary for evaluating performance and alignment.

Reward

An immediately better quantitative evaluation metric than loss, is reward. While the loss is the metric the model tries to minimize, the reward is the objective which it indirectly tries to maximize. Reward is a great evaluation metric because it can encompass information about both image fidelity and image-text alignment. By utilizing a reward function or reward model, practically any objective can be encoded for model evaluation. For example, if the reward function measures "colorfulness," as presented in Experiment 2: *Colorfulness*, a higher mean reward indicates that the trained model has a tendency to produce more colorful images as defined by the reward function. Similarly, if the reward model measures "animals wearing pink sunglasses," as presented in Experiment 6: *Pink*, it can be an evaluation metric for human preference on the task, measuring both image fidelity and image-text alignment of animals wearing pink sunglasses.

However, it can be problematic to use the same reward model for both training and evaluation. If the reward model is a bad approximation of the alignment objective, the reward can still be high despite poor image fidelity or image-text alignment as rated by humans. We experienced this during Experiment 6: *Pink*, where the reward monotonically increased, even past the point where the samples started deteriorating.

Another problem of using the same reward model for both training and evaluation is mode collapse. Mode collapse is another symptom of aggressive policy optimization, where the model is encouraged to maximize the mean reward, even when the peak reward stops increasing. This encourages the worse-rewarded samples to become more similar to the best-rewarded samples until the mean reward matches the peak reward. Mode collapse was very noticeable towards the end of Experiment 6: *Pink*, due to the prolonged training, as the animals in the images converged towards the end.

If we were to naively evaluate performance and alignment using the reward model alone, we could be deceived by increasing rewards despite degrading image quality, as the variety decreases and the model exploits the reward model's weaknesses. To better automatically evaluate the performance and alignment of a trained model using a scalar reward, a separate reward model could be used for evaluation that would not be prone to the reward exploitation. However, this would not guard against mode collapse.

The episodic reward graph was among the metrics actively used to evaluate the performance and alignment throughout training in this Thesis (together with human evaluation) until it was obvious that the model had overtrained and that the reward model was no longer a reliable approximation of human preferences for the trained model.

KL-divergence

KL-divergence is a useful metric for evaluating the difference between two parametrically related models and provides insights into alignment. A model that deviates significantly from the base model may experience decreased image quality as the methodology shifts from alignment to training. However, determining what constitutes an acceptable level of sample-deviation depends on a number of factors, including the model, the training objective, and the task at hand. Since this has not been empirically studied in this thesis, the degree to which KL-divergence can be used to evaluate performance or alignment of text-to-image models trained with RLHF is uncertain.

FID Score

The FID score (see Background in subsection 2.12.2) belongs to the same category as KL-divergence and is useful for comparing the performance of the base and aligned models on a common benchmark dataset like MS-COCO (presented in Background in subsection 2.11.4). This comparison can determine whether the model retains knowledge of the elements it was not aligned on and ensure the model does not interfere with unrelated aspects. Consequently, it is most interesting for observing if the model's performance worsens. While it is possible to calculate a FID score for seen prompts between the two models, it is more closely paralleled with KL-divergence and might not be necessary.

7.8.7. Human Evaluation

Human evaluation is necessary to gain a precise understanding of how well models trained with RLHF perform in terms of image fidelity and image-text alignment. The main way this study conducted human evaluation was through ad hoc observation of sample images. The ad hoc observation were conducted on related, partially related, and unrelated prompts to better evaluate image fidelity and image-text alignment, and to understand how well the alignment generalizes across domains while maintaining the original knowledge. Preferably, blind human evaluation trials should also be carried out.

Ad Hoc Observation We performed ad hoc observation of the trained models by examining randomly sampled images from rollout and same-seed validation images to get a sense of the model's capabilities and shortcomings. This evaluation method was especially useful for Experiment 4: *Macaw* and Experiment 6: *Pink*, where we could see the reward increase during training, and then validate the model's improved performance and alignment by comparing sample images. This made it possible to pinpoint where model fidelity and image-text alignment started to deteriorate, despite the increasing reward. While the ad hoc observation performed in this work does not provide a quantifiable evaluation of model performance or alignment, it made it easier to select a good model from training which we could then perform more extensive and laborious evaluation on, including a potential future blind experiment. As such, we consider it an important part of evaluating models trained with RLHF.

8. Conclusion and Future Work

This chapter concludes the thesis by providing several directions for future work and some brief final words.

8.1. Future Work

In this thesis, we have introduced a novel approach to optimize text-to-image models by leveraging human feedback. Although our initial results are promising, there are many areas where future research can further improve and refine this methodology. We outline several directions that could be pursued to address current limitations and strengthen the potential impact of this work.

Aligning to general human preferences. To enhance the generalization of our models to a wider range of preferences, it is necessary to gather feedback from multiple human labelers. *PickScore* (experiment 7) presented preliminary evidence that incorporating diverse opinions can improve overall alignment. However, more research is warranted to determine the possible extent of this improvement.

More nuanced human feedback. We acknowledge that merely collecting a preference for one image over another lacks sufficient nuance. Future work could involve collecting preferences for multiple objectives, such as perceived fidelity and toxicity, and then aligning policies to a multi-objective reward model.

Developing stronger reward models. The reward models explored in the presented experiments were prone to reward exploitation by the reinforcement learning algorithm. Potential avenues for improvement include exploring new architectures, employing ensemble methods to increase stability, and gathering more training data to enhance generalization performance.

Investigating multi-iterative RLHF. One potential way to mitigate issues stemming from overoptimization is to iteratively optimize policies and retrain reward models in multiple iterations. This approach might help prevent artifacts from arising during later stages of training.

Exploring alternative optimization strategies. Our work has shown that non-RL-based methods, such as simply aligning the policy using supervised training, hold promise as alternative training strategies. Additionally, investigating self-supervised RL techniques that do not require human feedback, like Reinforcement Learning from AI Feedback (RLAIF) by Bai et al. (2022), could offer valuable insights into further improving model text-to-image models using less data.

Incorporating complementary training techniques. As demonstrated by Ouyang et al. (2022) with Proximal Policy Optimization (PPO)-ptx, alternating the policy optimization with training on the original objective can help maintain the base distribution while aligning the model. Future work involves adapting and testing such strategies for aligning text-to-image models.

Blind Human Evaluation. To better understand the impact of our methodology, it is essential to conduct blind human evaluations of the optimized models. These assessments would provide useful insights into the strengths and weaknesses of our approach, as well as its overall effect on model performance as perceived by actual humans, not just proxy reward models.

Improving KL regularization. A productive line of future work would be to conduct ablation studies to determine appropriate KL penalty coefficients for different policy and reward model pairings. Furthermore, additional methods for KL regularization, such as dynamic KL penalties, could be explored to better preserve the original distribution throughout the training process.

Ethical considerations. As we advance these techniques, a more profound exploration into the ethical implications of aligning text-to-image models based on human preferences should be conducted. Potential risks related to biases and harmful depictions should be studied and mitigated in order to develop models that are beneficial to everyone and adhere to ethical principles.

Addressing these opportunities would advance our methodology towards applicability in real-world text-to-image generation systems. It would also provide a more comprehensive understanding of the benefits, limitations, and risks involved in RLHF alignment for generative models.

8.2. Conclusion

Our experiments have shown that RLHF is a viable and effective approach to align diffusion-based text-to-image models with human preferences. We successfully optimized diffusion policies using fixed reward functions, existing reward models training using human feedback, and our own reward models based on human preferences.

We recognize that there is still substantial room for further investigation and improvement in our proposed methodology. First, our work has identified a number of technical challenges and limitations that need to be addressed in future research, such as KL regularization, reward model exploitation, and mode collapse. Additionally, we have only begun to explore the vast potential of capturing general human preferences through large-scale data collection and reward modeling. Furthermore, our work has touched upon various ethical considerations that need to be addressed when deploying this methodology in real-world applications, such as bias mitigation, unintended consequences, and risks of misuse.

In conclusion, our work represents an essential step forward in aligning complex generative models with human values and demonstrating the viability of RLHF in the text-to-image domain. We hope that our method serves as a foundation upon which future research can build more robust, controllable, and ethically grounded generative AI systems.

Bibliography

Amit Auddy. Binary insertion sort. https://www.geeksforgeeks.org/binary-insertion-sort/, 2023. Accessed: 2023-06-24.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL https://arxiv.org/abs/1409.0473.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2, 1989.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space, 2022. URL https://arxiv.org/abs/2209.02535.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2305.16381*, 2023.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. URL https://mitpress.mit.edu/9780262561167/.

Yasutaka Furusho and Kazushi Ikeda. Theoretical analysis of skip connections and batch normalization from generalization and optimization perspectives. *APSIPA Transactions on Signal and Information Processing*, 9, 02 2020. doi:10.1017/ATSIP.2020.7.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. *arXiv preprint arXiv:2210.10760*, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. *arXiv preprint arXiv:2212.09611*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47–1, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

Dongjun Kim, Yeongmin Kim, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. *arXiv preprint arXiv:2211.17091*, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *arXiv preprint arXiv:2305.01569*, 2023.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017. URL `http://arxiv.org/abs/1703.03130`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL `https://arxiv.org/abs/1301.3781`.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

André Susano Pinto, Alexander Kolesnikov, Yuge Shi, Lucas Beyer, and Xiaohua Zhai. Tuning computer vision models with task rewards. *arXiv preprint arXiv:2302.08242*, 2023.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge (2014). *arXiv preprint arXiv:1409.0575*, 2(3), 2014.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022a.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022b.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

John Schulman. Approximating kl divergence. http://joschu.net/blog/kl-approx.html, 2020. Accessed: 2023-06-24.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, Rapha Gontijo Lopes, Shengjia Zhao, Arun Vijayvergiya, Eric Sigler, Adam Perelman, Chelsea Voss, Mike Heaton, Joel Parish, Dave Cummings, Rajeev Nayak, Valerie Balcom, David Schnurr, Tomer Kaftan, Chris Hallacy, Nicholas Turley, Noah Deutsch, Vik Goel, Jonathan Ward, Aris Konstantinidis, Wojciech Zaremba, Long Ouyang, Leonard Bogdonoff, Joshua Gross, David Medina, Sarah Yoo, Teddy Lee, Ryan Lowe, Dan Mossing, Joost Huizinga, Roger Jiang, Carroll Wainwright, Diogo Almeida, Steph Lin, Marvin Zhang, Kai Xiao, Katarina Slama, Steven Bills, Alex Gray, Jan Leike, Jakub Pachocki, Phil Tillet, Shantanu Jain, Greg Brockman, Nick Ryder, Alex Paino, Qiming Yuan, Clemens Winter, Ben Wang, Mo Bavarian, Igor Babuschkin, Szymon

Sidor, Ingmar Kanitscheider, Mikhail Pavlov, Matthias Plappert, Nik Tezak, Heewoo Jun, William Zhuk, Vitchyr Pong, Lukasz Kaiser, Jerry Tworek, Andrew Carr, Lilian Weng, Sandhini Agarwal, Karl Cobbe, Vineet Kosaraju, Alethea Power, Stanislas Polu, Jesse Han, Raul Puri, Shawn Jain, Benjamin Chess, Christian Gibson, Oleg Boiko, Emy Parparita, Amin Tootoonchian, Kyle Kosic, and Christopher Hesse. Introducing chatgpt. `https://openai.com/blog/chatgpt`, 2022. Accessed: 2023-06-24.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL `http://arxiv.org/abs/1508.07909`.

Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

CFI Team. Pareto efficiency. `https://corporatefinanceinstitute.com/resources/economics/pareto-efficiency/`, 2022. Accessed: 2023-06-24.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Better aligning text-to-image models with human preference. *arXiv preprint arXiv:2303.14420*, 2023.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*, 2023.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Appendices

A. Colorfulness Reward Function Code

Python code for the *colorfulness* reward function used in section 6.5:

```python
import numpy as np
from skimage.color import rgb2lab
import torch

def colorfulness_reward(images, vivid_threshold=10, n_hue_bins=7,
↪   **kwargs):
    """
    Compute a colorfulness reward for a batch of images.
    Input:
        images: a numpy array of shape (N, H, W, C) representing N
↪   images with
            height H, width W, and C color channels.
    Output:
        A numpy array of shape (N,) representing the reward for each
↪   image.
    """

    device = images.device  # Save device for output

    # Convert to Lab colorspace
    images = (images / 2 + 0.5).clamp(0, 1)  # Move images to a [0, 1]
    ↪   distribution
    images = images.detach().cpu().permute(0, 2, 3, 1).numpy()  #
    ↪   Convert to numpy

    lab_images = np.stack([rgb2lab(image) for image in images])

    # Compute the ratio of pixels with vivid color (vividness)
    a = lab_images[..., 1]
    b = lab_images[..., 2]
    vivid_mask = a ** 2 + b ** 2 > vivid_threshold
    vivid_ratio = np.mean(vivid_mask, axis=(1, 2))

    # Compute the entropy of hue values (diversity)
    hue = np.arctan2(b, a)
    hue_bins = np.linspace(-np.pi, np.pi, n_hue_bins + 1)
    hue_discrete = np.digitize(hue, hue_bins) - 1
    hue_discrete[hue_discrete == n_hue_bins] = n_hue_bins - 1  # replace
     ↪  the last bin with the second to last bin
    hue_counts = np.apply_along_axis(np.bincount, axis=-1,
     ↪  arr=hue_discrete.reshape(hue_discrete.shape[0], -1),
     ↪  minlength=n_hue_bins)
    hue_probs = hue_counts / np.sum(hue_counts, axis=-1, keepdims=True)
    hue_entropy = -np.sum(hue_probs * np.log2(hue_probs +
     ↪  np.finfo(float).eps), axis=-1)  # add epsilon for numerical
     ↪  stability

    # Combine terms and return reward
    reward = vivid_ratio * hue_entropy

    return torch.tensor(reward).float().to(device).squeeze(-1)
```

B. Value Function Architecture for Experiment 1: *Target Image*

```
import torch.nn as nn
import torch.nn.functional as F

class CriticModel(nn.Module):
    """
    Inspired by:
    https://nextjournal.com/gkoehler/pytorch-mnist

↪   https://godatadriven.com/blog/how-to-build-your-first-image-classifier-using-pytorch
    """
    def __init__(self, in_channels=1):
        super().__init__()
        self.channels = in_channels
        self.out_dim = 10
        self.conv = nn.Conv2d(self.channels, self.channels * 6,
        ↪  kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.fc1 = nn.Linear(self.channels * 6 * 16 * 16, 24)
        self.fc2 = nn.Linear(24, self.out_dim)

    def forward(self, x):
        x = F.relu(self.conv(x))
        x = self.pool(x)
        x = x.view(-1, self.channels * 6 * 16 * 16)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

C. Data for Generating Modified Prompts in Experiment 6: *Pink*

C.1. Animals

elephant	dolphin	parrot	giraffe
koala	turtle	squirrel	hippopotamus
rabbit	hedgehog	goose	lemur
bear	sheep	cat	pig
tiger	panda	horse	hummingbird
penguin	fox	wolf	monkey
frog	chicken	duck	zebra
rhinoceros	kangaroo	crocodile	snake
spider	shark	whale	antelope
butterfly	eagle	owl	bee
jellyfish	lobster	octopus	crab
moose	seahorse	gorilla	sloth
otter	walrus	raccoon	skunk
bat	armadillo	beaver	cow
donkey	camel	llama	goat
swan	flamingo	lizard	chameleon

C.2. Colors

red
orange
yellow
green
blue
purple
black
white

D. Prompts Used in Experiment 7: *PickScore*

D.1. Diverse Prompts

The 24 prompts below were used to generate the diverse samples in Figure 6.35, Figure 6.36, and Figure 6.39. They are given in column-major order relative to the figures.

1. *"a photo of an astronaut riding a horse"*
2. *"a blue sky with fluffy white clouds"*
3. *"a serene landscape in the style of Impressionism, filled with bright yellows and deep blues, featuring a river reflecting the afternoon sun"*
4. *"a futuristic cityscape at night, bustling with neon signs, hyperloops, and flying cars, in the vein of cyberpunk aesthetics"*
5. *"a simple still life composition of assorted fruits on a tablecloth, reminiscent of Caravaggio's use of chiaroscuro"*
6. *"a dreamy representation of the Aurora Borealis, vibrant with green and purple hues, reflecting in a still, clear lake"*
7. *"the concept of 'chaos' portrayed through an abstract expressionist painting, using a riot of swirling colors and brushstrokes"*
8. *"yellow duck, colored, splash, high detailed, painting"*
9. *"a lighthouse on a stormy night, waves crashing against the rocks"*
10. *"a photo of a hot air balloon over a wheat field"*
11. *"landscape photography of swiss alps, mountains, nature, high quality, summer, bloom"*
12. *"macro photography of a red mushroom, morning dew, nature, high quality, bokeh, dslr"*
13. *"a cherry blossom tree by a tranquil pond, pink, expressed through minimalist Japanese ink painting"*

14. *"a grand Gothic cathedral under a stormy sky, with dramatic shadows and highlights, inspired by Romanticism"*
15. *"a tranquil rural scene of a cottage amidst a blooming meadow under the midday sun, painted in the style of American Regionalism"*
16. *"a photograph of a deer wearing pink sunglasses"*
17. *"green apple, shiny surface, vivid colors"*
18. *"an ancient Egyptian pyramid under a starry night sky."*
19. *"a macaw parrot with a colorful plumage perched on a branch"*
20. *"low poly elephant figure"*
21. *"sunset over the ocean, with a sailboat in the distance"*
22. *"logo of a red fox"*
23. *"photo of a nebula by the Hubble space telescope"*
24. *"a mystical depiction of a unicorn in an enchanted forest"*

D.2. Animal Prompts

The 24 prompts below were used to generate the animal samples in Figure 6.37, Figure 6.38, and Figure 6.40. They are given in column-major order relative to the figures.

1. *"a photograph of a lion"*
2. *"a photograph of a mouse"*
3. *"a photograph of a corgi"*
4. *"a photograph of a deer"*
5. *"a photograph of a fox"*
6. *"a photograph of an elephant"*
7. *"a photograph of a kangaroo"*
8. *"a photograph of a eagle"*
9. *"a photograph of a koala"*
10. *"a photograph of a turtle"*
11. *"a photograph of a squirrel"*
12. *"a photograph of a hedgehog"*
13. *"a photograph of a panda"*

14. *“a photograph of a hummingbird”*
15. *“a photograph of a frog”*
16. *“a photograph of a monkey”*
17. *“a photograph of an owl”*
18. *“a photograph of a bee”*
19. *“a photograph of an octopus”*
20. *“a photograph of an otter”*
21. *“a photograph of a raccoon”*
22. *“a photograph of a llama”*
23. *“a photograph of a jellyfish”*
24. *“a photograph of a pig”*

E. Labeling Software

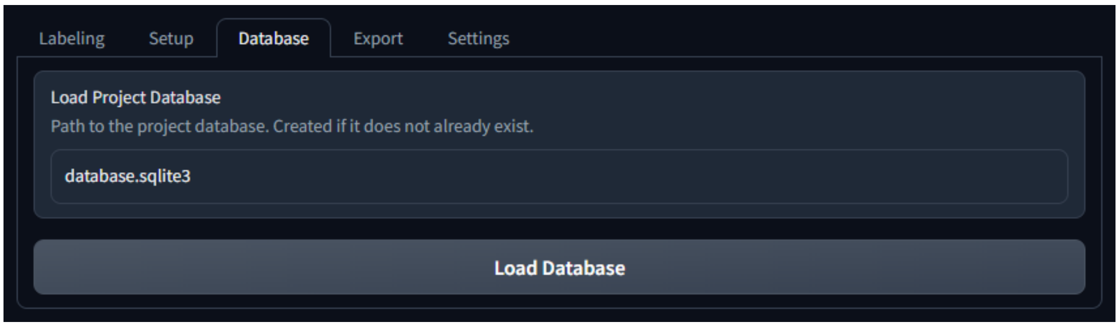
The main labeling pane for preference collection is explained in section 4.2. The remaining panes are presented here.

Figure E.1 shows the pane used to select which database the projects should read from and write to. If a database does not exist at the target location, a new database is automatically configured.

Figure E.2 shows the settings pane. This pane is used to quickly configure the settings of the software. Here, the keys to select preferred image can be changed, as well as the keys to navigate back and forth between image comparisons and tasks. It is also here transitivity is enabled or disabled in the software.

Figure E.3 shows the pane for loading or creating a project. On project creation, the project settings are specified, including the location of the images being preference rated, the task size, what type of conditioning to use (single-class, multi-class, prompts) and the project instructions.

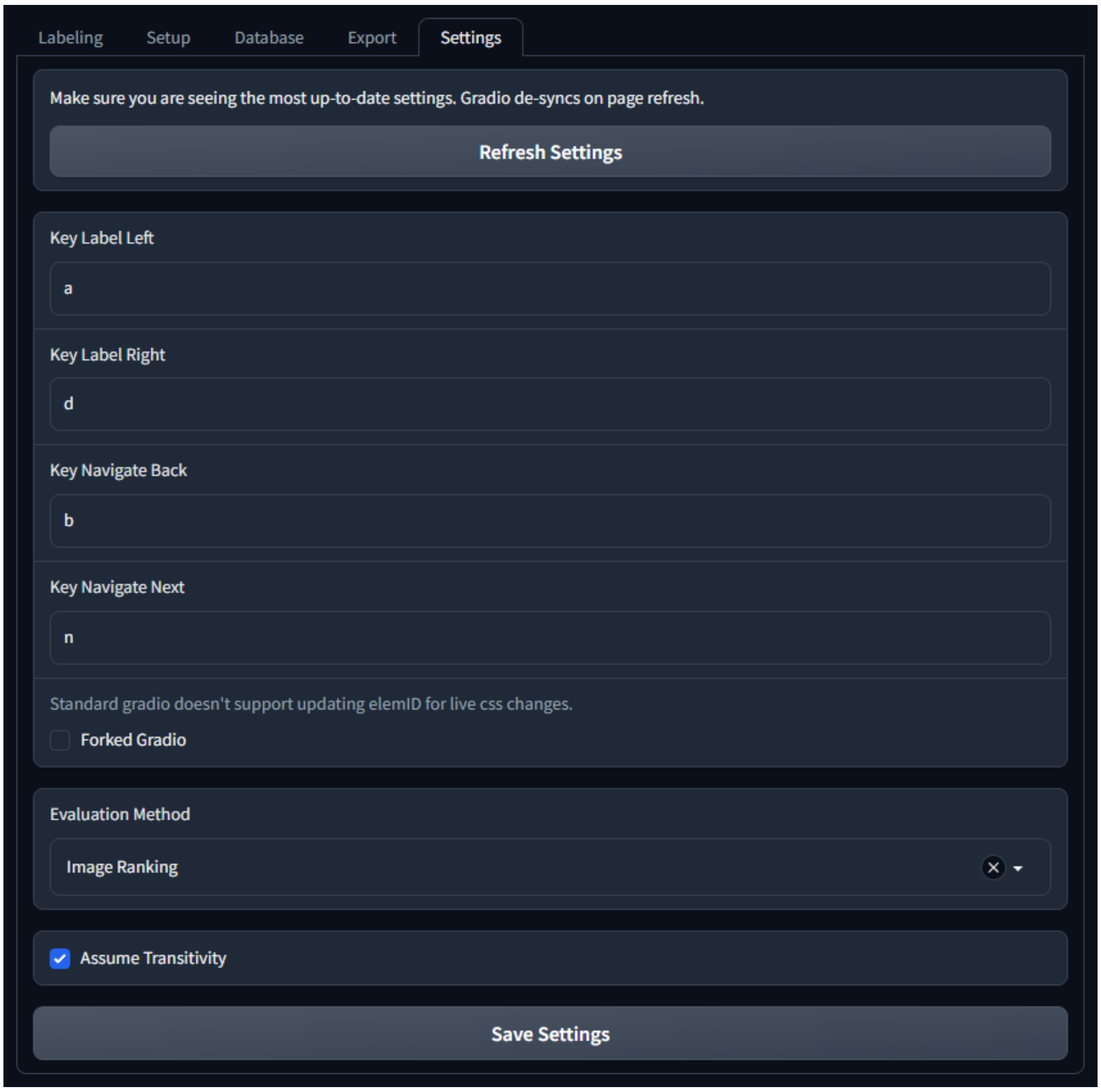
Figure E.4 shows the pane used to export labeled projects for reward model training. The pane supports splitting the tasks into training and validation tasks to train and evaluate reward models.



Path to the project database. Created if it does not already exist.

database.sqlite3

Figure E.1.: The pane used to load project databases in the custom labeling software.



Make sure you are seeing the most up-to-date settings. Gradio de-syncs on page refresh.

Key Label Left: a

Key Label Right: d

Key Navigate Back: b

Key Navigate Next: n

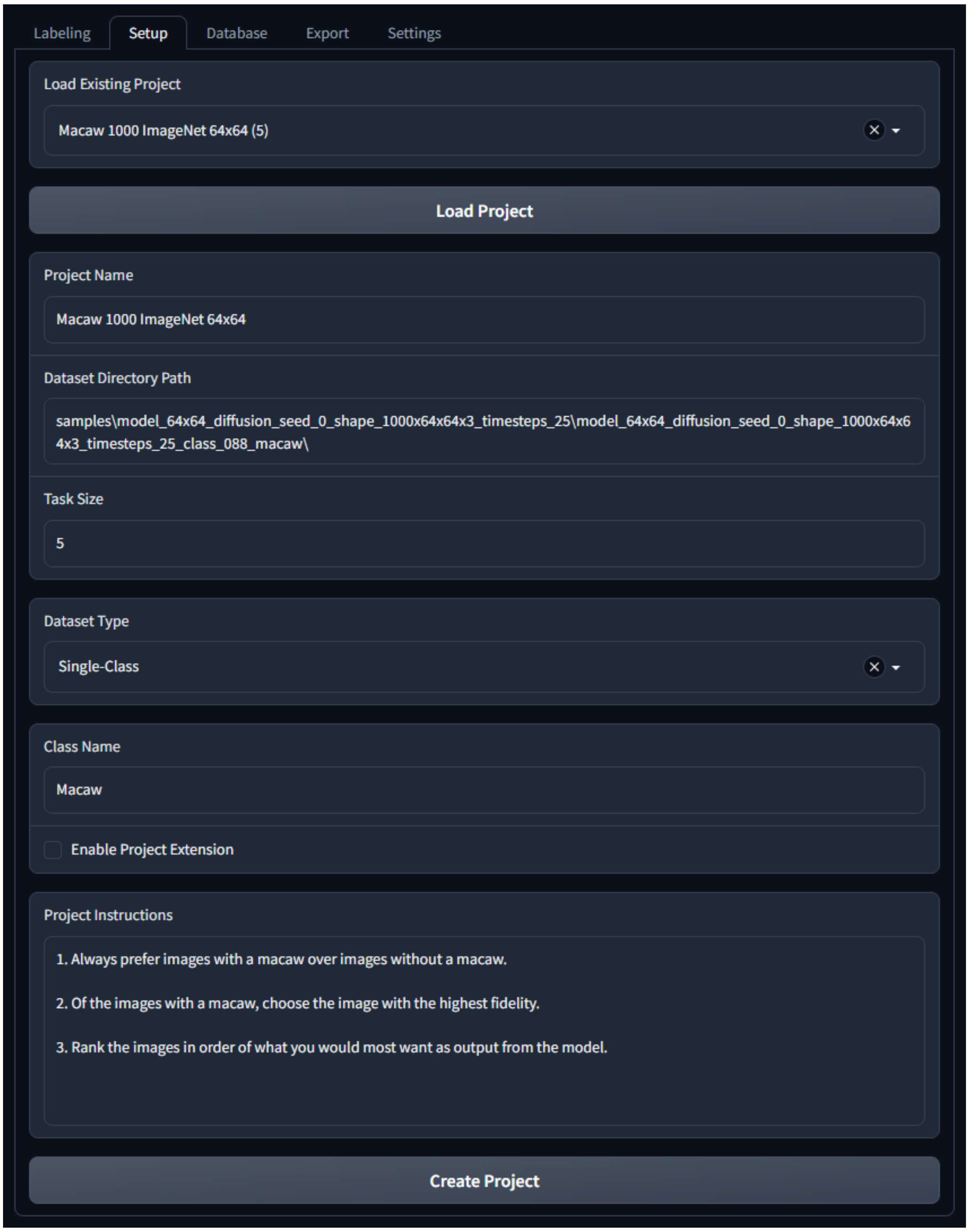
Standard gradio doesn't support updating elemID for live css changes.

☐ Forked Gradio

Evaluation Method: Image Ranking

☑ Assume Transitivity

Figure E.2.: The pane used to configure the custom labeling software's settings.



Load Existing Project

Macaw 1000 ImageNet 64x64 (5)

Project Name

Macaw 1000 ImageNet 64x64

Dataset Directory Path

samples\model_64x64_diffusion_seed_0_shape_1000x64x64x3_timesteps_25\model_64x64_diffusion_seed_0_shape_1000x64x64x3_timesteps_25_class_088_macaw\

Task Size

5

Dataset Type

Single-Class

Class Name

Macaw

☐ Enable Project Extension

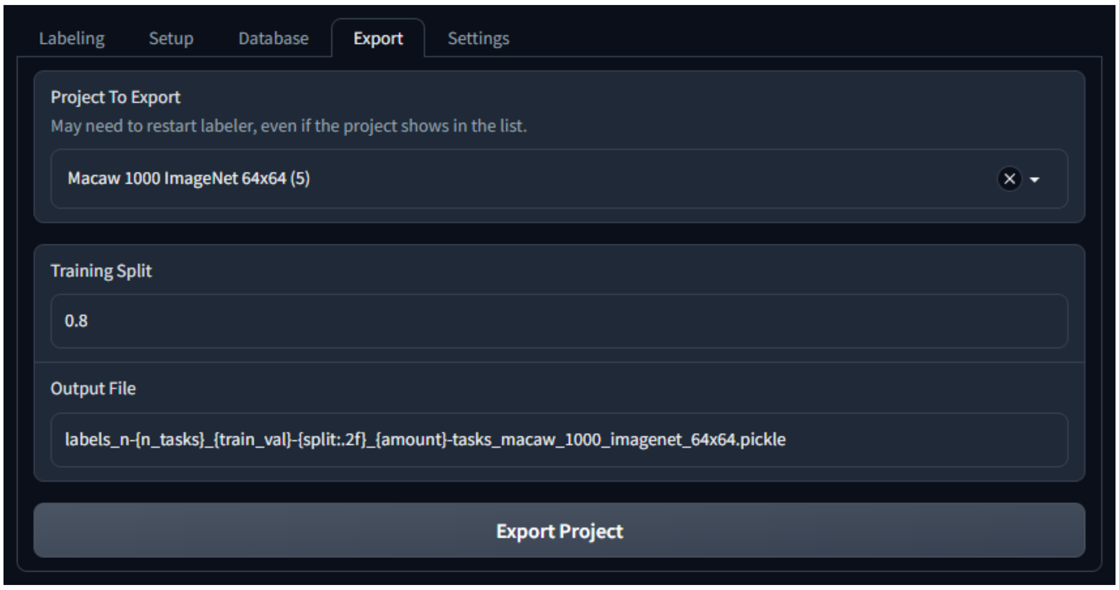
Project Instructions

1. Always prefer images with a macaw over images without a macaw.

2. Of the images with a macaw, choose the image with the highest fidelity.

3. Rank the images in order of what you would most want as output from the model.

Figure E.3.: The pane used to create or load a project in the custom labeling software.



Project To Export
May need to restart labeler, even if the project shows in the list.
Macaw 1000 ImageNet 64x64 (5)

Training Split
0.8

Output File
labels_n-{n_tasks}_{train_val}-{split:.2f}_{amount}-tasks_macaw_1000_imagenet_64x64.pickle

Figure E.4.: The pane used to export a labeled project in the custom labeling software.