Using Style Ambiguity Loss to Improve Aesthetics of Diffusion Models

Anonymous authors
Paper under double-blind review

Abstract

Teaching text-to-image models to be creative involves using style ambiguity loss. In this work, we explore using the style ambiguity training objective, used to approximate creativity, on a diffusion model. We then experiment with forms of style ambiguity loss that do not require a labeled dataset, and find that the models trained with style ambiguity loss can generate better images than the baseline diffusion models.

1 Introduction

With every new invention comes a new wave of possibilities. Humans have been making pictures since before recorded history, so its only natural that there would be interest in computational image generation. Artificially generating photographs that are indistinguishable from real ones has become so easy and effective that there is even concern over "deepfakes" being used for propaganda or illicit purposes (Pawelec, 2022). However, there is also demand for machine-generated images that are not just realistic but artistic, as exemplified by the picture that sold for nearly half a million dollars (Christie's, 2018) at auction, or the picture used in the sitcom *Silicon Valley* (AICAN, 2024). While the definition of art is somewhat philosophical and beyond the scope of this paper, it most certainly has to be creative. Creative assets are usually defined as being "novel and useful" (Diedrich et al., 2015). A string of random characters is novel in that we cannot predict the next characters. However, if the purpose of characters is to compose words that compose sentences that communicate a message, then a string of random characters is not useful. Most people would not consider a string of random characters to be creative or art. If the utility of an image is to depict objects and scenes that humans understand and can recognize, then generative models perform well in that regard. Novelty, on the other hand has been underexplored. A breakthrough was the invention of the Creative Adversarial Network (Elgammal et al., 2017), which used a style ambiguity loss to train a generator network to generate images that could not be classified as belonging to a particular style. However, this style ambiguity loss requires a pretrained classifier: Every set of styles or concepts requires training a classifier before even training a model to generate images, which itself takes time and requires a labeled dataset, the collection and curation of which can be expensive. To circumvent these issues, we propose using a classifier for style ambiguity that does not require any additional training and can be easily applied to any dataset, labeled or unlabeled. Additionally, the Creative Adversarial Network was based off of the GAN framework, which have fallen out of favor compared to the more powerful diffusion models (Luo, 2022), so we propose training a diffusion model with style ambiguity loss. Our contributions are as follows:

- We applied style ambiguity loss to diffusion models via reinforcement learning
- We developed versatile CLIP-based and K-Means-based creative style ambiguity losses that do not require training a separate style classifier on a labeled dataset.
- Empirically, we find that training a diffusion model with style ambiguity loss teaches the model to generate very novel outputs that may score higher on empirical metrics than models trained without style ambiguity loss

2 Related Work

2.1 Creativity

Creative work has been formulated as work having novelty, in that it differs from other similar objects, and also utility, in that it still performs a function (Cropley, 2006). For example, a Corinthian column has elaborate, interesting, unexpected adornments (novelty) but still holds up a building (utility). A distinction can also be made between "P-creativity", where the work is novel to the creator, and "H-creativity" where the work is novel to everyone (Boden, 1990). Computational techniques to be creative include using genetic algorithms (DiPaola & Gabora, 2008), reconstructing artifacts from novel collections of attributes (Iqbal et al., 2016), and most relevantly to this work, using Generative Adversarial Networks (Elgammal et al., 2017) with a style ambiguity loss.

2.2 Computational Art

One of the first algorithmic approaches dates back to the 1970s with the now primitive AARON (McCorduck, 1991), which was initially only capable of drawing black and white sketches. Generative Adversarial Networks (Goodfellow et al., 2014), or GANs, were some of the first models to be able to create complex, photorealistic images and seemed to have potential to be able to make art. Despite many problems with GANs, such as mode collapse and unstable training (Saxena & Cao, 2023), GANs and further improvements (Arjovsky et al., 2017; Karras et al., 2019; 2018) were state of the art until the introduction of diffusion Sohl-Dickstein et al. (2015). Diffusion models such as IMAGEN (Saharia et al., 2022) and DALLE-3 (Betker et al.) have attained widespread commercial success (and controversy) due to their widespread adoption.

2.3 Reinforcement Learning

Reinforcement learning (RL) is a method of training a model by having it take actions that generate a reward signal and change the environment, thus changing the impact and availability of future actions Qiang & Zhongli (2011). RL has been used for tasks as diverse as playing board games (Silver et al., 2017), protein design (Lutz et al., 2023), self-driving vehicles (Kiran et al., 2021) and quantitative finance (Sahu et al., 2023). Policy-gradient RL (Sutton et al., 1999) optimizes a policy π that chooses which action to take at any given timestep, as opposed to value-based methods that may use a heuristic to determine the optimal choice. Examples of policy gradient methods include Soft Actor Critic (Haarnoja et al., 2018), Deep Deterministic Policy Gradient (Lillicrap et al., 2019) and Trust Region Policy Optimization (Schulman et al., 2017a).

3 Background

3.1 Creative Adversarial Network

A Generative Adversarial Network, or GAN (Goodfellow et al., 2014), consists of two models, a generator and a discriminator. The generator generates samples from noise, and the discriminator detects if the samples are drawn from the real data or generated. During training, the generator is trained to trick the discriminator into classifying generated images as real, and the discriminator is trained to classify images correctly. Given a generator $G : \mathbb{R}^{noise} \to \mathbb{R}^{h \times w \times 3}$, a discriminator $D : \mathbb{R}^{h \times w \times 3} \to [0, 1]$ real images $x \in \mathbb{R}^{h \times w \times 3}$, and noise $\mathcal{Z} \in \mathbb{R}^{noise}$, the objective is:

$$\min_G \max_D \mathbb{E}_x[log(D(x)] + \mathbb{E}_{\mathcal{Z}}[log(1 - D(G(\mathcal{Z}))]$$

Elgammal et al. (2017) introduced the Creative Adversarial Network, or CAN, which was a DCGAN (Radford et al., 2016) where the discriminator was also trained to classify real samples, minimizing the style classification loss. Given N classes of image (such as ukiyo-e, baroque, impressionism, etc.), the classification modules of the Discriminator $D_C : \mathbb{R}^{h \times w \times 3} \to \mathbb{R}^N$ that returns a probability distribution over the N_s style classes for an image and the real labels $\ell \in \mathbb{R}^N$, the style classification loss was:

$$L_{SL} = \mathbb{E}_{x,\ell}[\mathbf{CE}(D_C(x), \ell)]$$

Where **CE** is the cross entropy function.

The generator was also trained to generate samples that could not be easily classified as belonging to one class. This stylistic ambiguity is a proxy for creativity or novelty. Given a vector $U \in \mathbb{R}^N$, where each entry $u_1, u_2, , u_N = \frac{1}{N}$, and the classification modules of the discriminator D_C the style ambiguity loss is:

$$L_{SA} = \mathbb{E}_{\mathcal{Z}}[\mathbf{CE}(C(G(\mathcal{Z})), U)]$$

The discriminator was additionally trained to minimize L_{SL} and the generator was additionally trained to minimize L_{SA}.

3.2 Diffusion

A diffusion model aims to learn to iteratively remove the noise from a corrupted sample to restore the original. Starting with x_0, the forward process q iteratively adds Gaussian noise to produce the noised version x_T, using a noise schedule $\beta_1 ... \beta_T$, which can be learned or manually set as a hyperparameter:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I})$$

More importantly, we also want to model the reverse process p, that turns a noisy sample x_T back into x_0, conditioned on some context c. As x_T is the fully noised version, $p(x_T|c) = \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$

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

Once the model has been trained, the reverse process, aka inference, generates a sample from noise $x_T \sim \mathcal{N}(0, 1)$. We use the DDIM technique (Song et al., 2022) for sampling. In this work, we use a variant of diffusion known as Stable Diffusion (Rombach et al., 2022), where x is replaced with a latent embedding $\mathcal{E}(x)$, where $\mathcal{E} : \mathbb{R}^{h \times w \times 3} \to \mathbb{R}^{h_z \times w_z \times c_z}$ is a frozen autoencoder (Kingma & Welling, 2022), and $h_z < h, w_z < w, c_z > 3$.

4 Methods

4.1 Denoising Diffusion Proximal Optimisation

Introduced by Black et al. (2023), Denoising Diffusion Proximal Optimisation, or DDPO, represents the reverse Diffusion Process as a Markov Decision Process (Bellman, 1957). A similar method was also pursued by Fan et al. (2023). Reinforcement learning training was then applied to a pretrained diffusion model. Following Schulman et al. (2017b), Black et al. (2023) also implemented clipping to protect the policy gradient $\nabla_\theta \mathcal{J}_{DDRL}$ from excessively large updates, and per prompt stat tracking to normalize rewards. We largely follow their method but use a different reward function. We fine-tune off of the pre-existing **stabilityai/stable-diffusion-2-base** checkpoint (Rombach et al., 2022) downloaded from `https://huggingface.co/stabilityai/stable-diffusion-2-base`.

4.2 Text Prompts

DDPO does not require any new data, given that we are fine-tuning off of a pretrained checkpoint. However, each time we train the model, we must decide which text prompts to use to condition the generation of images. We used the set of (painting, drawing, art) as our text prompts

4.3 Datasets and Labels

Training the CAN, of course, requires a labeled dataset. We used two different real datasets based off of WikiArt (Saleh & Elgammal, 2015):

1. **Full:** the WikiArt dataset as is. Consists of roughly 80k images.

2. **Mediums:** given the the text prompt set (painting, drawing, art), we used BLIP (Li et al., 2022) to generate captions, and then selected the 10 classes that had the highest fraction of their descriptions containing any of the words in **Mediums**. We then used the images in WikiArt that belonged to those classes. Consists of roughly 20k images.

This also meant we had two different sets of style class labels: L_{full}, all 27 class labels, used with **Full** and L_{med}, the 10 labels used in **Mediums**. For implementation details regarding the captions, and a list of the class labels used for **Mediums** refer to appendix C

4.4 DDPO Reward Function

In the original DDPO paper, the authors used four different reward functions for four different tasks. For example, they used a scorer trained on the LAION dataset (Schuhmann & Beaumont, 2022) as the reward function to improve the aesthetic quality of generated outputs. In this paper, we use the reward model based on Elgammal et al. (2017), where the model is rewarded for stylistic ambiguity, combined with a reward for utility. Given a pretrained CLIP (Radford et al., 2021) model, that can return a similarity score for each image-text pair: $CLIP : \mathbb{R}^{text} \times \mathbb{R}^{h \times w \times 3} \rightarrow \mathbb{R}$, image $x_0 \in \mathbb{R}^{h \times w \times 3}$ generated with text prompt s, cross entropy **CE**, uniform distribution $U \in \mathbb{R}^N$ and a classifier $C : \mathbb{R}^{h \times w \times 3} \rightarrow \mathbb{R}^N$ we want to maximize:

$$R(x_0) = -\lambda_{novelty}\mathbf{CE}(C(x_0), U) + \lambda_{utility}CLIP(s, x_0)$$

The first term on the left side of the equation represents style ambiguity loss, and the second term maintains alignment between text and image, essentially keeping the model from straying "too far" from the text prompt; these terms approximate novelty and utility, respectively. We actually have *multiple* choices of classifier, which we dicuss.

4.4.1 Discriminator Classifier

We can use the classification module of the CAN discriminator as the classifier in the reward function, setting $C = D_C$. This is the traditional method of style ambiguity loss, and the baseline against which we are comparing the other two types of classifier with.

4.4.2 CLIP-Based Style Classifier

For each generated image x_0, for each class name $s_i, 1 \leq i \leq N_s$ in the style class label set we want to use, we find $CLIP(s_i, x_0)$. We can then create a vector $(CLIP(s_1, x_0), CLIP(s_2, x_0), , , CLIP(s_{N_s}, x_0))$ and then use softmax to normalize the vector and define the result as $CC(x_0)$. Formally:

$$CC(x_0) = \mathbf{softmax}((CLIP(s_1, x_0), CLIP(s_2, x_0), , , CLIP(s_{N_s}, x_0))$$

Then we set $C = CC$. Given the four sets of labels, we had four different types of CLIP-Based classifier $CC_{full}, CC_{med}, CC_{sub}$ and CC_{syn}, using $L_{full}, L_{med}, L_{sub}$ and L_{syn} respectively.

4.4.3 K-Means Image Based Classifiers

Alternatively, N_I source images in a dataset, we can embed the labels or images into the CLIP embedding space $\in \mathbb{R}^{768}$ and perform k-means clustering to generate k centers. Given a CLIP Embedder $E : \mathbb{R}^{h \times w \times 3} \rightarrow \mathbb{R}^{768}$ mapping images to embeddings, and the k centers $c_1, c_2, , , c_k$ we can create a vector $(\frac{1}{||E(x_0) - c_1||}, \frac{1}{||E(x_0) - c_2||}, , , , \frac{1}{||E(x_0) - c_k||}$ and then use softmax to normalize the vector and define the result as $KM(x_0))$. Formally:

$$KMEANS(x_0) = \mathbf{softmax}(\frac{1}{||E(x_0) - c_1||}, \frac{1}{||E(x_0) - c_2||}, , , , \frac{1}{||E(x_0) - c_k||})$$

Then we set $C = KM$.

Letter	$\lambda_{novelty}$	$\lambda_{utility}$	Classifier?	Dataset	Inference Steps
M_0	1	0.25	Discriminator	Full	30
M_1	1	0.25	CLIP-Based	Full	30
M_2	1	0.25	K-Means Based	Full	30
M_3	1	0.25	Discriminator	Mediums	30
M_4	1	0.25	CLIP-Based	Mediums	30
M_5	1	0.25	K-Means Based	Mediums	30
M_6	0	1	None	N/A	30
M_7	0	1	None	N/A	10
M_8	0	0	None	N/A	30
M_9	0	0	None	N/A	10

Table 1: Diffusion Methods

5 Results

Given three types of classifier and two datasets, we had six models. We refer to those models as the "creative" models. We also wanted to compare these models to a model trained with *just* utility loss and a model that was not trained at all off of the benchmark. For the non-creative models, we also generate samples with a smaller amount of inference steps, to dispel the notion that the "creative" models are just learning blurrier, noisier versions of normal models. A breakdown on all the models is in table 1. Note that for methods M_1 and M_4, we don't use the data in the data for the classifier per se; however, we use the style class labels, which are unique to the datasets in question. M_1 uses L_{full} and M_3 uses $L_{mediums}$.

We generated all images with width and height = 512. The authors used width and height = 256 in the original CAN paper. However, given that larger, more detailed images are preferred by most people, we thought it more relevant to focus on larger images.

5.1 Quantitative Evaluation

For each dataset, for each choice of classifier, we an evaluation dataset of 100 images. We used the exact same prompts and random seeds for each. I.e. if the nth image generated by M_0 used prompt "painting" and random seed = $z \in \mathbb{Z}$, so would the nth image generated by M_1, M_2, etc. We used two scoring metrics to score the models:

- **AVA Score: (AVA)** Consisting of CLIP+Multi-Layer Perceptron (Haykin, 2000), the AVA model was trained on the AVA dataset (Murray et al., 2016) of images and average rankings by human subjects, in order to learn to approximate human preferences given an image. We used the CLIP model weights from the **clip-vit-large-patch14** checkpoint and the Multi-Layer Perceptron weights downloaded from `https://huggingface.co/trl-lib/ddpo-aesthetic-predictor`.

- **Image Reward: (IR)** The image reward model (Xu et al., 2023) was trained to score images given their text description based on a dataset of images and human rankings. We used the **image-reward** python library found at `https://github.com/THUDM/ImageReward/tree/main`.

Results of our experiments are shown in tables 2. Each cell contains the mean (and standard deviation). Evidently, some creative models perform better than the uncreative models; the highest IR score was attained by M_0, and the highest AVA score was attained by M_5. However, we note that using fewer labels dramatically improves the IR of the CLIP classifier (comparing M_1 to $M4$, and using a smaller dataset improves the IR *and* AVA of the K Means classifier (comparing M_2 to M_5). We see the opposite trend with using a discriminator; M_0 scores higher than M_3 for both. We further visualize these results in table 3. We see that IR tends to be rather skewed.

Model	AVA	IR
M_0	5.49 (0.47)	1.33 (0.34)
M_1	5.49 (0.38)	0.84 (1.06)
M_2	4.98 (0.4)	-0.88 (0.31)
M_3	5.12 (0.31)	-0.84 (0.31)
M_4	5.39 (0.38)	1.08 (0.87)
M_5	5.89 (0.39)	1.27 (0.37)
M_6	5.12 (0.29)	0.85 (0.87)
M_7	5.18 (0.33)	0.73 (0.78)
M_8	4.12 (0.88)	-1.87 (0.58)
M_9	4.22 (0.84)	-2.07 (0.43)

Table 2: Aesthetic Scores by Models

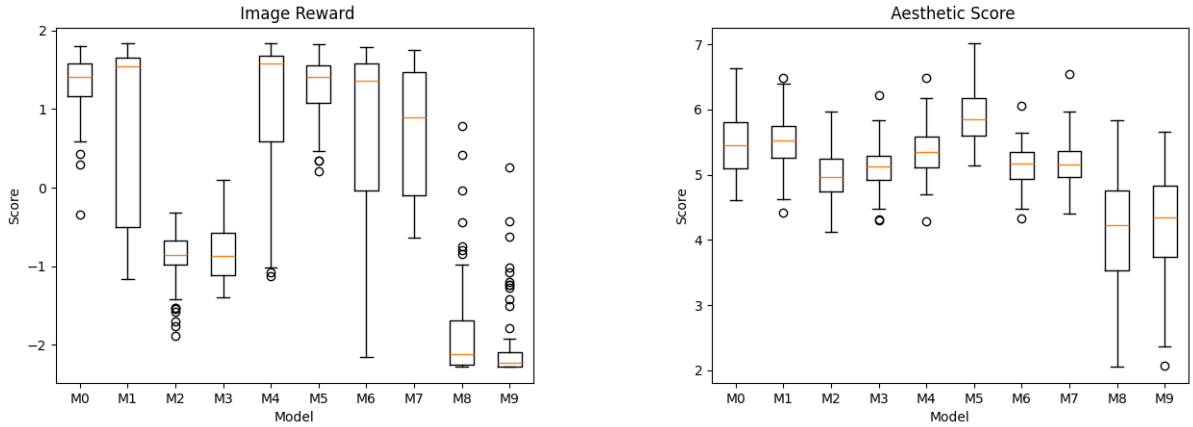

Table 3: Score Box Plots

5.2 Visual Results

We also provide a few visual results in figure 2. Each image in each row was generated with the same prompt and random seed. For more pictures, consult appendix A.

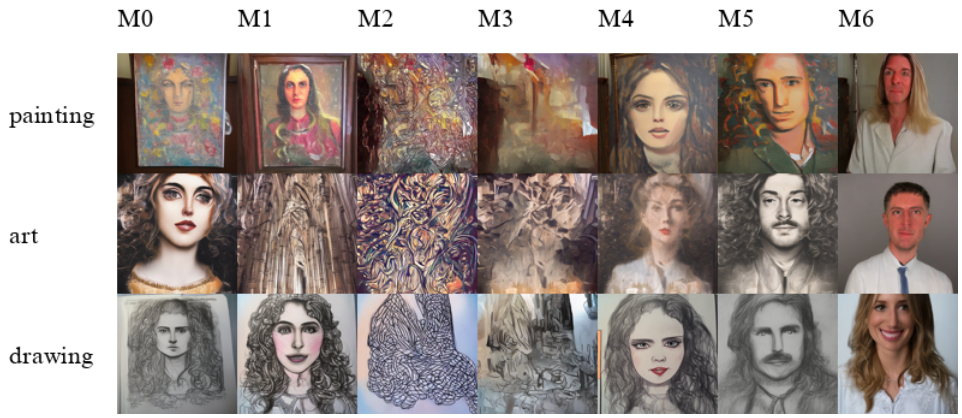

Figure 1: Image Comparisons (Only Creative Models and M_6)

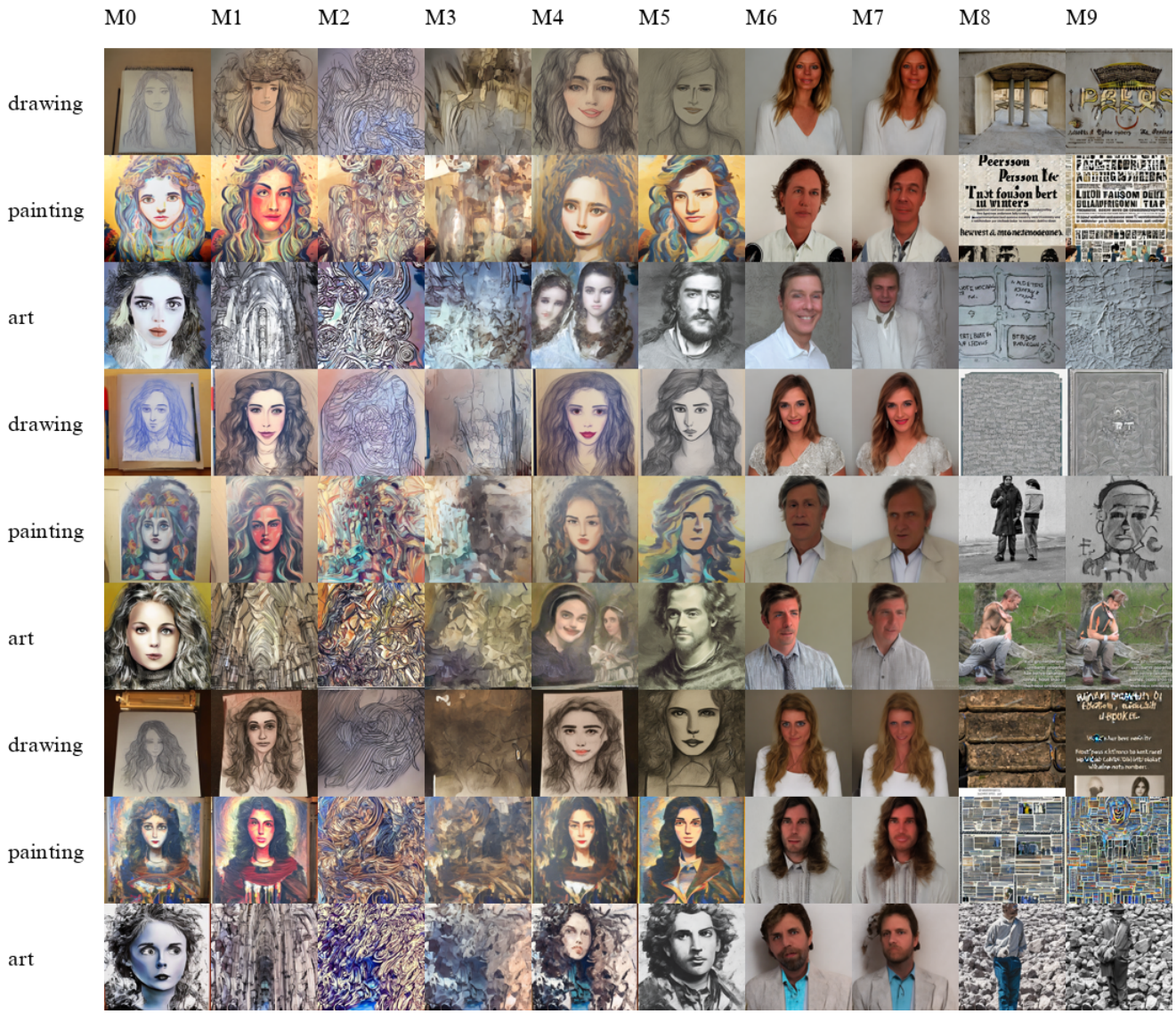

Figure 2: Image Comparisons

Curiously, many of the creative models generate faces. However, given that M_6 and M_7 do this as well, we suspect that may be a product of the utility function. On the other hand, the M_2 and M_3 models tend to mostly generate abstract noise. This is somewhat similar to the outputs generated in the original CAN paper.

5.3 Content and Style Similarities

We also compared the similarities between each model. Given that the nth image generated by each model used the same prompt and seed across all models, we could compare the cosine distance of the embeddings of the nth image for each model with the embeddings of the nth image of every other model, and average them. We used style and content embeddings from the **dino-vits16** checkpoint (Caron et al., 2021) from `https://huggingface.co/facebook/dino-vits16`. Many other works have commented on the ability to extract separable style and content from vision transformers (Tumanyan et al., 2022; Kwon & Ye, 2023).

Unsurprisingly, M_6 and $M7$, as well as M_8 and M_9, have very high similarities, given that they are the same models. The creative models are generally more similar to each other, and more distinct from the uncreative models.

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
M_0	1.0	0.4	0.27	0.26	0.43	0.41	0.35	0.35	0.23	0.23
M_1	0.4	1.0	0.31	0.29	0.43	0.39	0.34	0.34	0.22	0.23
M_2	0.27	0.31	1.0	0.4	0.26	0.24	0.18	0.19	0.22	0.24
M_3	0.26	0.29	0.4	1.0	0.26	0.23	0.19	0.19	0.24	0.26
M_4	0.43	0.43	0.26	0.26	1.0	0.45	0.35	0.35	0.23	0.22
M_5	0.41	0.39	0.24	0.23	0.45	1.0	0.42	0.41	0.22	0.21
M_6	0.35	0.34	0.18	0.19	0.35	0.42	1.0	0.85	0.24	0.22
M_7	0.35	0.34	0.19	0.19	0.35	0.41	0.85	1.0	0.24	0.22
M_8	0.23	0.22	0.22	0.24	0.23	0.22	0.24	0.24	1.0	0.53
M_9	0.23	0.23	0.24	0.26	0.22	0.21	0.22	0.22	0.53	1.0

Table 4: Average Content Similarities

	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
M_0	1.0	0.54	0.35	0.3	0.65	0.64	0.34	0.37	0.18	0.19
M_1	0.54	1.0	0.41	0.35	0.58	0.54	0.33	0.35	0.18	0.19
M_2	0.35	0.41	1.0	0.41	0.29	0.3	0.12	0.13	0.14	0.18
M_3	0.3	0.35	0.41	1.0	0.31	0.29	0.15	0.18	0.19	0.22
M_4	0.65	0.58	0.29	0.31	1.0	0.65	0.36	0.39	0.2	0.19
M_5	0.64	0.54	0.3	0.29	0.65	1.0	0.44	0.45	0.2	0.19
M_6	0.34	0.33	0.12	0.15	0.36	0.44	1.0	0.91	0.14	0.11
M_7	0.37	0.35	0.13	0.18	0.39	0.45	0.91	1.0	0.15	0.12
M_8	0.18	0.18	0.14	0.19	0.2	0.2	0.14	0.15	1.0	0.63
M_9	0.19	0.19	0.18	0.22	0.19	0.19	0.11	0.12	0.63	1.0

Table 5: Average Style Similarities

5.4 Visualizing The Possibility Space

For each model, we combined each pair of evaluation datasets and then then performed dimensionality reduction using PCA and TSNE to embed images into 2 dimensions. Thus, we could visualize the overlap, or lack thereof, between the possibility space of each model. We see that there are some stark contrasts between the uncreative and creative models, often breaking into very distant clusters.

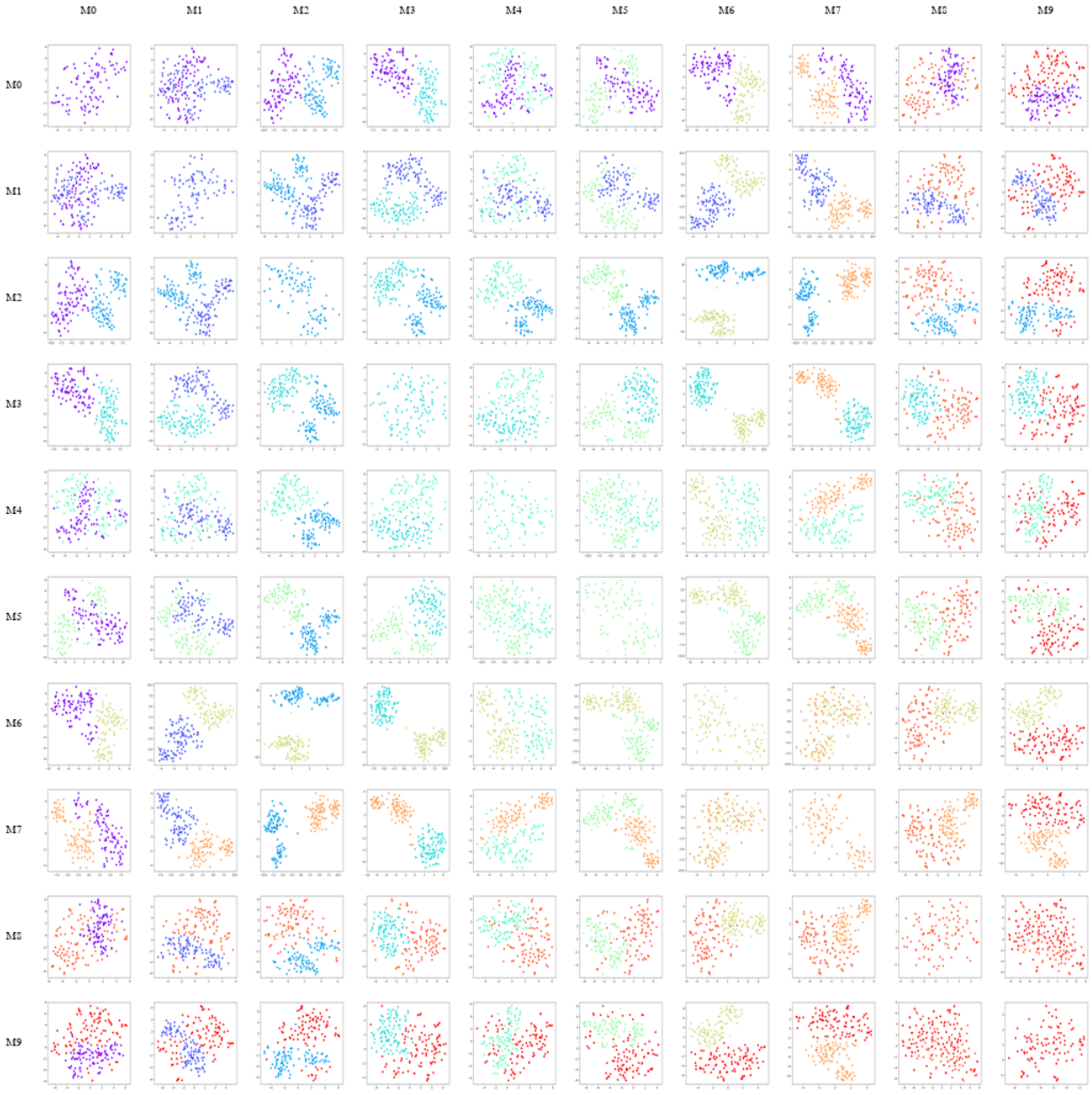

Figure 3: Possibility Space

6 Conclusion

In this paper, we introduced innovative techniques to incorporate style ambiguity loss into diffusion models through reinforcement learning. Our primary contributions include the development of versatile CLIP-based and K-Means-based creative style ambiguity losses, which eliminate the need for training a separate style classifier on labeled datasets. Our empirical results demonstrate that training diffusion models with style ambiguity loss significantly enhances their ability to generate novel outputs. These models consistently achieve higher scores on empirical metrics compared to models trained without style ambiguity loss. Our findings suggest that incorporating style ambiguity can be a powerful approach to foster creativity and diversity in generated content, opening new avenues for future research in the field of generative models. Further work remains. As noted, the use of the CLIP alignment function for utility tends to product a lot of faces for the prompts we used. Use of another alignment technique may be better, or it could be supplemented with some sort of method to increase output diversity (Sadat et al., 2024). Furthermore, this method can likely be applied to different datasets, consisting of different images or even different modalities like audio or video. Using the K-Means Classifier for style ambiguity loss is particularly adept at this, given there is no need for *any* labels.

Broader Impact Statement

Many are concerned about the impacts of generative AI. By making art, this work infringes upon a domain once exclusive to humans. Companies have faced scrutiny for possibly using AI (Gutierrez, 2024), and many creatives, such as screenwriters and actors, have voiced concerns about whether their jobs are safe (del Barco, 2023). Nonetheless, using AI can help humans by making them more efficient, providing inspiration, and generating ideas (Fortino, 2023; Campitiello, 2023; Darling, 2022). It's also not certain how copyright protection will function for AI-generated art (Watiktinnakorn et al., 2023), given copyright law is based on the premise that creative works originate solely from human authorship. Clear, consistent policies, both at the government level and by industry and/or academic groups, will be needed to mitigate the harm and maximize the benefits for all members of society.

7 Assistance

Author Contributions

Acknowledgements

References

AICAN. The birth of venus: Aican first collection, 2024. URL https://www.aican.io/store/products/the-birth-of-venus-aican-first-collection. Accessed: 2024-07-16.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. URL http://www.jstor.org/stable/24900506.

James Betker, Gabriel Goh, Li Jing, † TimBrooks, Jianfeng Wang, Linjie Li, † LongOuyang, † JuntangZhuang, † JoyceLee, † YufeiGuo, † WesamManassra, † PrafullaDhariwal, † CaseyChu, † YunxinJiao, and Aditya Ramesh. Improving image generation with better captions. URL https://api.semanticscholar.org/CorpusID:264403242.

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

Margaret Boden. *The Creative Mind.* Abacus, 1990.

Jess Campitiello. Ai vs. artist: The future of creativity, 2023. URL https://tech.cornell.edu/news/ai-vs-artist-the-future-of-creativity/.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

Christie's. Is artificial intelligence set to become art's next medium? *Christie's*, 2018. URL https://www.christies.com/en/stories/a-collaboration-between-two-artists-one-human-one-a-machine-0cd01f4e232f4279a525a446d60d4cd1.

Arthur Cropley. In praise of convergent thinking. *Creativity Research Journal*, 18(3):391–404, 2006. doi: 10.1207/s15326934crj1803_13. URL https://doi.org/10.1207/s15326934crj1803_13.

Kate Darling. Ai image generators will help artists, not replace them, 2022. URL https://www.sciencefocus.com/news/ai-image-generators-will-help-artists-not-replace-them.

Mandalit del Barco. Some sag-aftra members are concerned about ai provisions in tentative deal, 2023. URL https://www.npr.org/2023/11/30/1216005659/some-sag-aftra-members-are-concerned-about-ai-provisions-in-tentative-deal.

Jennifer Diedrich, Mathias Benedek, Emanuel Jauk, and Aljoscha Neubauer. Are creative ideas novel and useful? *Psychology of Aesthetics, Creativity, and the Arts*, 9:35–40, 02 2015. doi: 10.1037/a0038688.

Steve DiPaola and Liane Gabora. Incorporating characteristics of human creativity into an evolutionary art algorithm. *Genetic Programming and Evolvable Machines*, 10(2):97–110, December 2008. ISSN 1573-7632. doi: 10.1007/s10710-008-9074-x. URL http://dx.doi.org/10.1007/s10710-008-9074-x.

Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark, 2022.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

Ahmed M. Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. CAN: creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *CoRR*, abs/1706.07068, 2017. URL http://arxiv.org/abs/1706.07068.

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, 2023.

Andres Fortino. Embracing creativity: How ai can enhance the creative process, 2023. URL https://www.sps.nyu.edu/homepage/emerging-technologies-collaborative/blog/2023/embracing-creativity-how-ai-can-enhance-the-creative-process.html.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

Luis Joshua Gutierrez. Wizards of the coast repeats anti-ai stance, fights accusation against latest magic the gathering promo, 2024. URL https://www.gamespot.com/articles/wizards-of-the-coast-repeats-anti-ai-stance-fights-accusation-against-latest-magic-the-gathering-pro1100-6520153/.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Simon Haykin. Neural networks: A guided tour. 2000.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Azlan Iqbal, Matej Guid, Simon Colton, Jana Krivec, Shazril Azman, and Boshra Haghighi. The digital synaptic neural substrate: A new approach to computational creativity, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey, 2021.

Gihyun Kwon and Jong Chul Ye. Diffusion-based image translation using disentangled style and content representation, 2023.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

Calvin Luo. Understanding diffusion models: A unified perspective, 2022.

Isaac D Lutz, Shunzhi Wang, Christoffer Norn, Alexis Courbet, Andrew J Borst, Yan Ting Zhao, Annie Dosey, Longxing Cao, Jinwei Xu, Elizabeth M Leaf, et al. Top-down design of protein architectures with reinforcement learning. *Science*, 380(6642):266–273, 2023.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3. Atlanta, GA, 2013.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

P. McCorduck. *Aaron's Code: Meta-art, Artificial Intelligence, and the Work of Harold Cohen*. W.H. Freeman, 1991. ISBN 9780716721734. URL `https://books.google.com/books?id=r3UyBgAAQBAJ`.

Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. 2016. URL `https://github.com/imfing/ava_downloader`.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Maria Pawelec. Deepfakes and democracy (theory): How synthetic audio-visual media for disinformation and hate speech threaten core democratic functions. *Digital Society*, 1, 09 2022. doi: 10.1007/s44206-022-00010-6.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL `http://jmlr.org/papers/v12/pedregosa11a.html`.

Wang Qiang and Zhan Zhongli. Reinforcement learning model, algorithms and its application. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pp. 1143–1146, 2011. doi: 10.1109/MEC.2011.6025669.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

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)*, pp. 10684–10695, June 2022.

Seyedmorteza Sadat, Jakob Buhmann, Derek Bradley, Otmar Hilliges, and Romann M. Weber. Cads: Unleashing the diversity of diffusion models through condition-annealed sampling, 2024. URL `https://arxiv.org/abs/2310.17347`.

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, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

Santosh Kumar Sahu, Anil Mokhade, and Neeraj Dhanraj Bokde. An overview of machine learning, deep learning, and reinforcement learning-based techniques in quantitative finance: Recent progress and challenges. *Applied Sciences*, 13(3):1956, 2023.

Babak Saleh and Ahmed M. Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *CoRR*, abs/1505.00855, 2015. URL `http://arxiv.org/abs/1505.00855`.

Divya Saxena and Jiannong Cao. Generative adversarial networks (gans survey): Challenges, solutions, and future directions, 2023.

Christoph Schuhmann and Romain Beaumont, 2022. URL `https://laion.ai/blog/laion-aesthetics/`.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017. URL `https://api.semanticscholar.org/CorpusID:205261034`.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. URL `http://arxiv.org/abs/1503.03585`.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL `https://arxiv.org/abs/2010.02502`.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf`.

Narek Tumanyan, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Splicing vit features for semantic appearance transfer, 2022.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Chawinthorn Watiktinnakorn, Jirawat Seesai, and Chutisant Kerdvibulvech. Blurring the lines: how ai is redefining artistic ownership and copyright. *Discover Artificial Intelligence*, 3, 11 2023. doi: 10.1007/s44163-023-00088-y.

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, 2023.

A Additional Images

We display a few more images in figures 4, 5 and 6. Once again, each image in each row was generated with the same seed and same prompt.

B WikiArt Classes

We host the original **WikiArt-Full** dataset on `REDACTEDWHILEUNDERBLINDREVIEW`. The 27 WikiArt style classes, L_{full} are listed in table 6

contemporary-realism	art-nouveau-modern	abstract-expressionism
northern-renaissance	mannerism-late-renaissance	early-renaissance
realism	action-painting	color-field-painting
pop-art	new-realism	pointillism
expressionism	analytical-cubism	symbolism
fauvism	minimalism	cubism
romanticism	ukiyo-e	high-renaissance
synthetic-cubism	baroque	post-impressionism
impressionism	rococo	na-ve-art-primitivism

Table 6: Styles

C Medium Subset

C.1 Captions

All captions were generated using the **Salesforce/blip-image-captioning-base** checkpoint downloaded from `https://huggingface.co/Salesforce/blip-image-captioning-base`.

C.2 Subset

The categories of images that had the highest percentage of images in the categories with text captions that contained one of the words in the relevant text prompt set, as well as the percentage of images that did so, and the quantity of images in said category are shown in table 7 . We host the **Mediums** datasets at `REDACTEDWHILEUNDERBLINDREVIEW`.

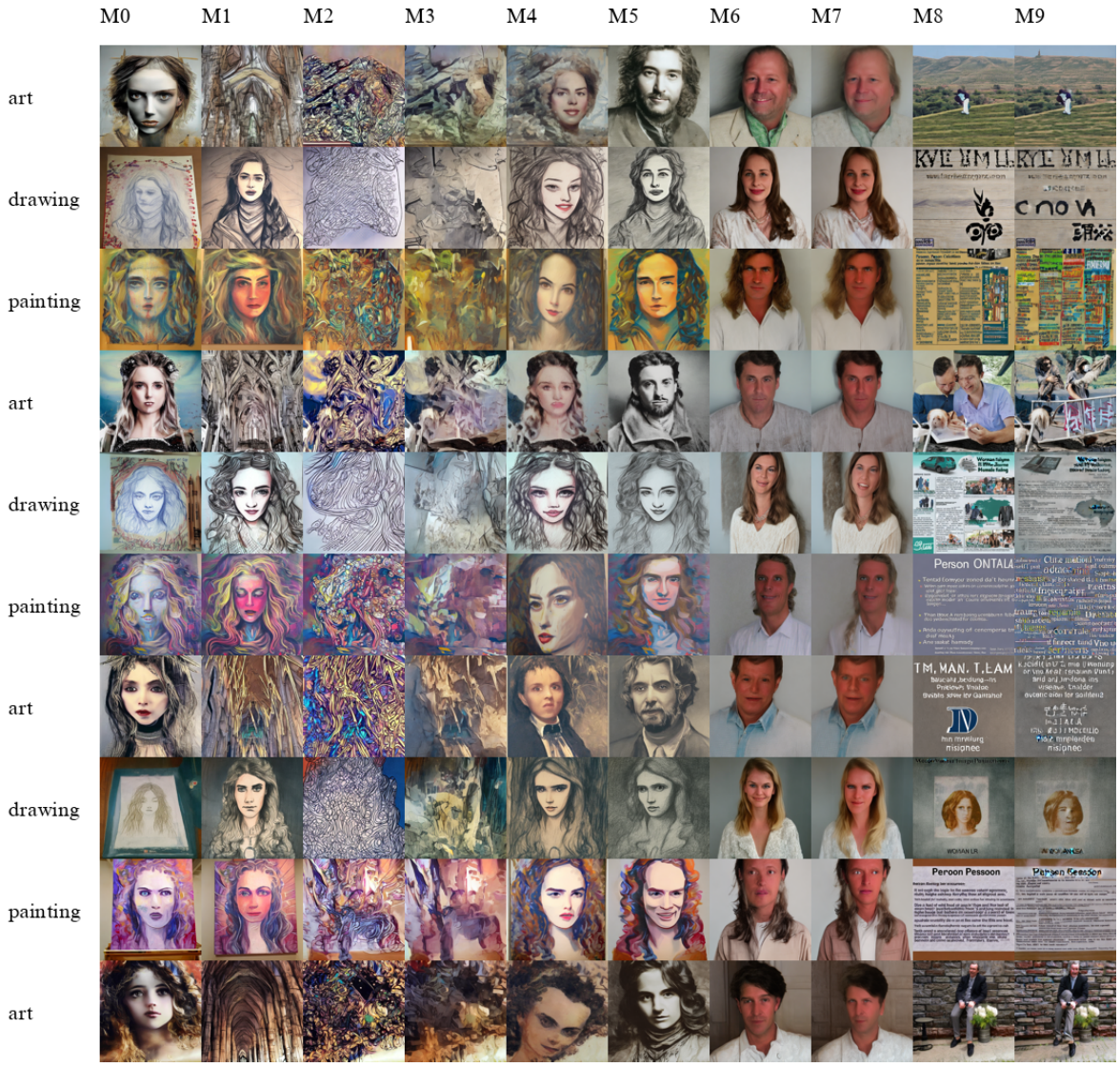

Figure 4: More Images

style class	quantity	percentage
expressionism	6054	91.56
post-impressionism	5832	89.25
fauvism	841	96.08
abstract-expressionism	2518	89.95
na-ve-art-primitivism	2148	93.39
cubism	2027	91.61
synthetic-cubism	197	89.85
analytical-cubism	105	91.43
new-realism	280	96.07
action-painting	93	93.55

Table 7: Mediums

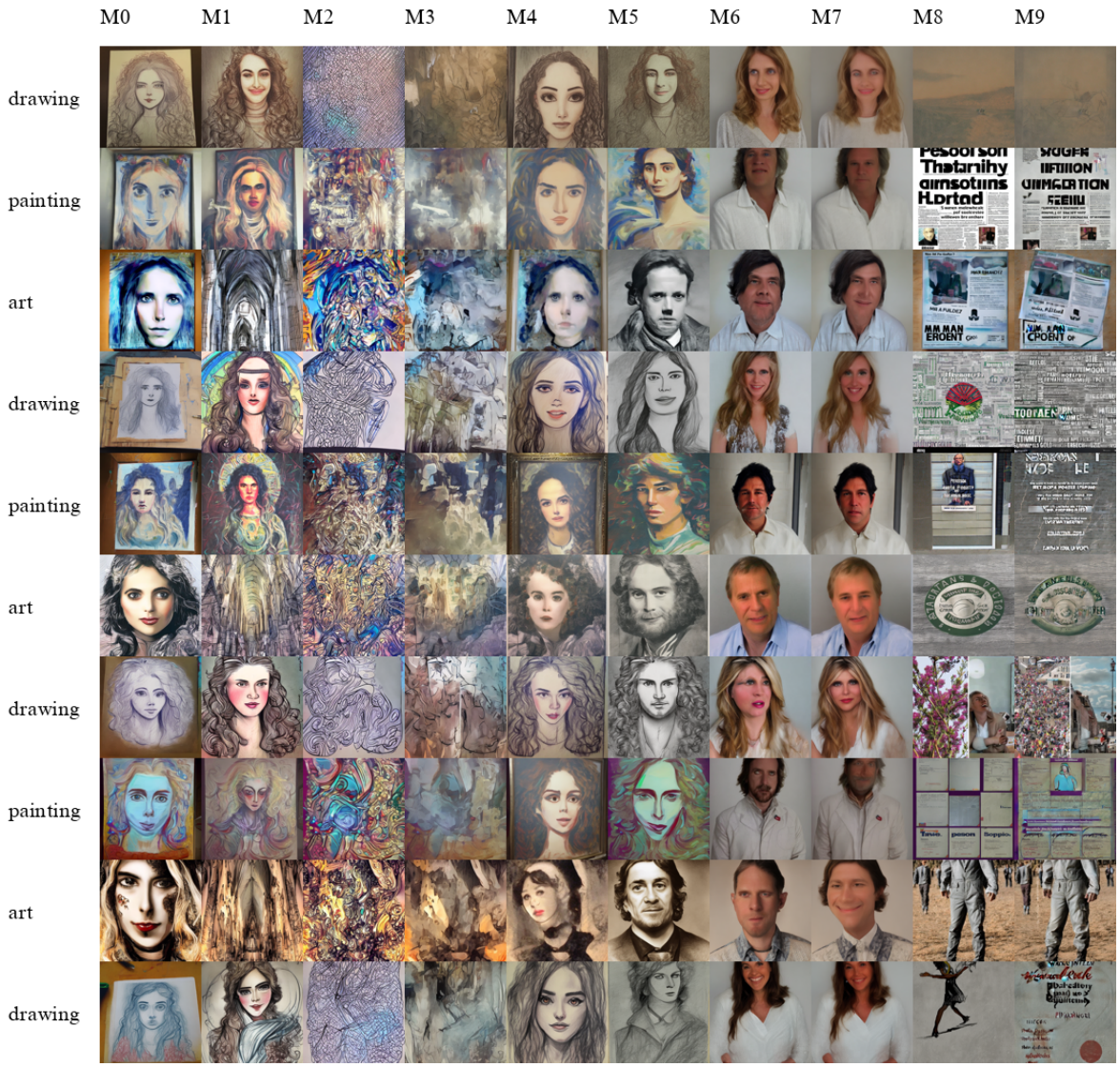

Figure 5: More Images

Figure 6: More Images

D Training

For reproducibility and transparency, the hyperparameters are listed in table 8 and table 9. All experiments were implemented in Python, building the models in **pytorch** (Paszke et al., 2017) using **accelerate** (Gugger et al., 2022) for efficient training. The diffusion models also relied on the **trl** (von Werra et al., 2020), **diffusers** (von Platen et al., 2022) and **peft** (Mangrulkar et al., 2022) libraries. The K-Means clustering was done using the k means implementation from **scikit-learn** (Pedregosa et al., 2011). A repository containing all code can be found on github at `REDACTEDWHILEUNDERREVIEW`. Each experiment was run using two NVIDIA A100 GPUs with 40 GB RAM.

Hyperparameter	Value
Epochs	25
Effective Batch Size	8
Batches per Epoch	32
Inference Steps per Image	30
LORA Matrix Rank	4
LORA α	4
Optimizer	AdamW
Learning Rate	0.0015
AdamW β_1	0.9
AdamW β_2	0.99
AdamW Weight decay	1e-4
AdamW ϵ	1e-8

Table 8: DDPO Hyperparameters

Hyperparameter	Value
Epochs	100
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Adam β_1	0.9
Adam β_2	0.99
Adam Weight decay	0.0
Adam ϵ	1e-8
Noise Dim	100
Wasserstein λ	10
Leaky ReLU negative slope	0.2
Convolutional Kernel	4
Convolutional Stride	2
Transpose Convolutional Kernel	4
Transpose Convolutional Stride	2

Table 9: CAN Hyperparameters

D.1 Architecture

For diffusion model training, the text encoder, autoencoder and unet were all loaded from `https://huggingface.co/stabilityai/stable-diffusion-2-base`. These model components were all frozen, but we added trainable LoRA weights to the cross-attention layers of the Unet. Parameter counts are shown in table 10. The diffusion model components used the same amount of parameters regardless of image size, but the generator and discriminator had more parameters as image size increased.

Model Component	Total Parameters	Trainable Parameters	Percent Trainable
Text Encoder	34,0387,840	0	0%
Autoencoder	83,653,863	0	0%
UNet	866,740,676	829,952	0.1%
Generator (Image Dim 512)	48,014,784	48,014,784	100%
Discriminator (Image Dim 512)	20,115,932	20,115,932	100%

Table 10: Parameter Counts

We used the convolutional neural network (Dumoulin & Visin, 2018) architecture described in Elgammal et al. (2017) for the CAN but had to use more/less layers to produce higher/lower dimension images. The generator takes a 1×100 gaussian noise vector $\in \mathbb{R}^{100} \sim \mathcal{N}(0, I)$ and maps it to a $4 \times 4 \times 2048$ latent space, via a convolutional transpose layer with kernel size = 4 and stride =1, followed by 6 transpose convolutional layers each upscaling the height and width dimensions by two, and halving the channel dimension (for example one of these transpose convolutional layers would map $\mathbb{R}^{4\times4\times2048} \rightarrow \mathbb{R}^{8\times8\times1024}$) followed by batch normalization (Ioffe & Szegedy, 2015) and Leaky ReLU (Maas et al., 2013), and then one final convolutional transpose layer with output channels = 3 and tanh (Dubey et al., 2022) activation function. Diagrams of the generator is shown in the figure 7.

For the discriminator, we first applied a convolution layer to downscale the input image height width dimensions by 2 and mapped the 3 input channel dimensions to 32 ($\mathbb{R}^{512\times512\times3} \rightarrow \mathbb{R}^{256\times256\times32}$) with Leaky ReLU activation. Then we had 5 convolutional layers each downscaling the height and width dimensions by 2 and doubling the channel dimension (for example, one of these convolutional layers would map $\mathbb{R}^{256\times256\times32} \rightarrow \mathbb{R}^{128\times128\times64}$) with batch normalization and Leaky ReLU activation. Then we had two more convolutional layers, each downscaling the height and width dimensions but keeping the channel dimensions constant (using the prior layer's channel dimensions), with batch normalization and Leaky ReLU activation. The output of the convolutional layers was then flattened. The discriminator had two heads- one for style classification (determining which style a real image belongs to) and one for binary classification (determining whether an image was real or fake). The binary classification head consisted of one linear layer with one output neuron. The style classification layer consisted of 2 linear layers with LeakyReLU activation and Dropout, with output 1024 output neurons and 512 output neurons, respectively, followed by a linear layer with 27 output neurons for the 27 artistic style classes. Diagrams of the discriminator is shown in 8.

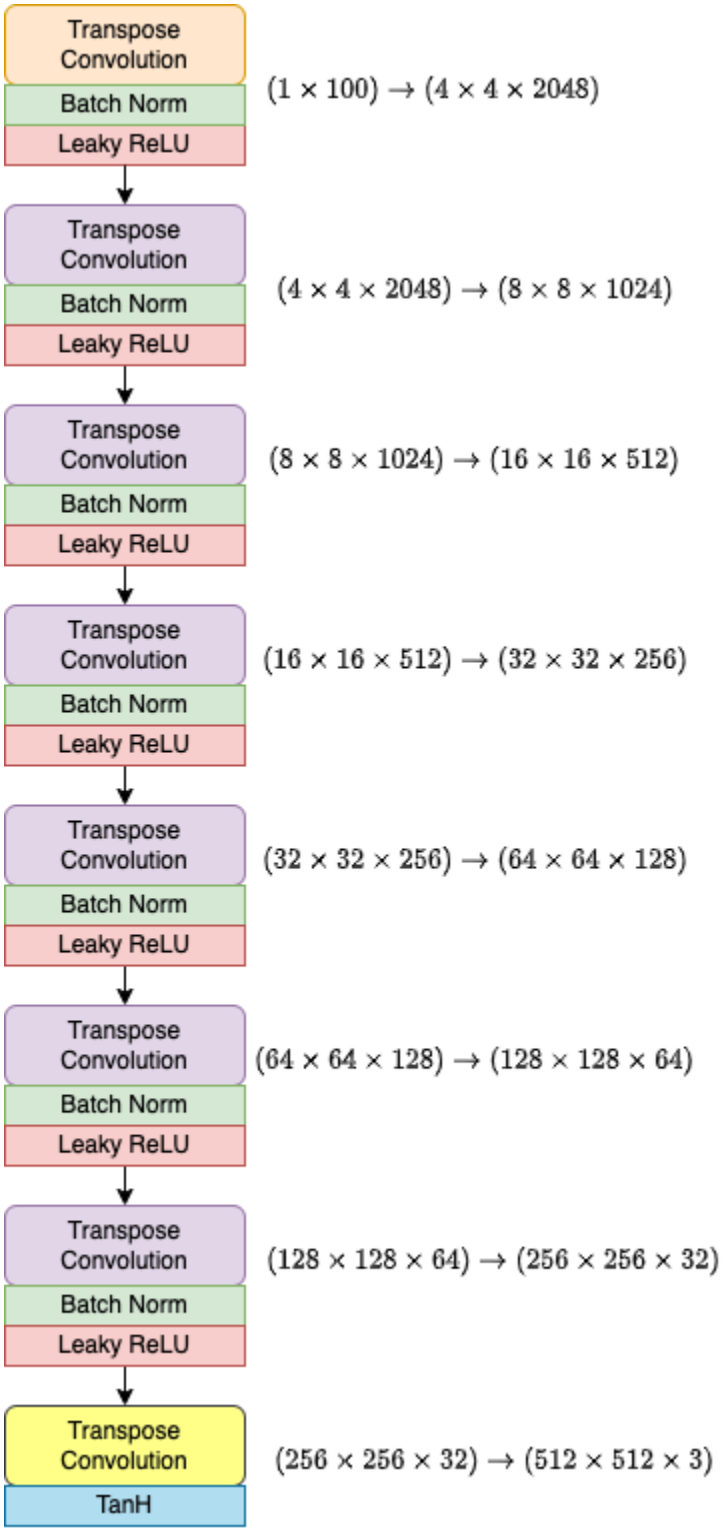

Figure 7: Generator Architecture (Image Dim 512)

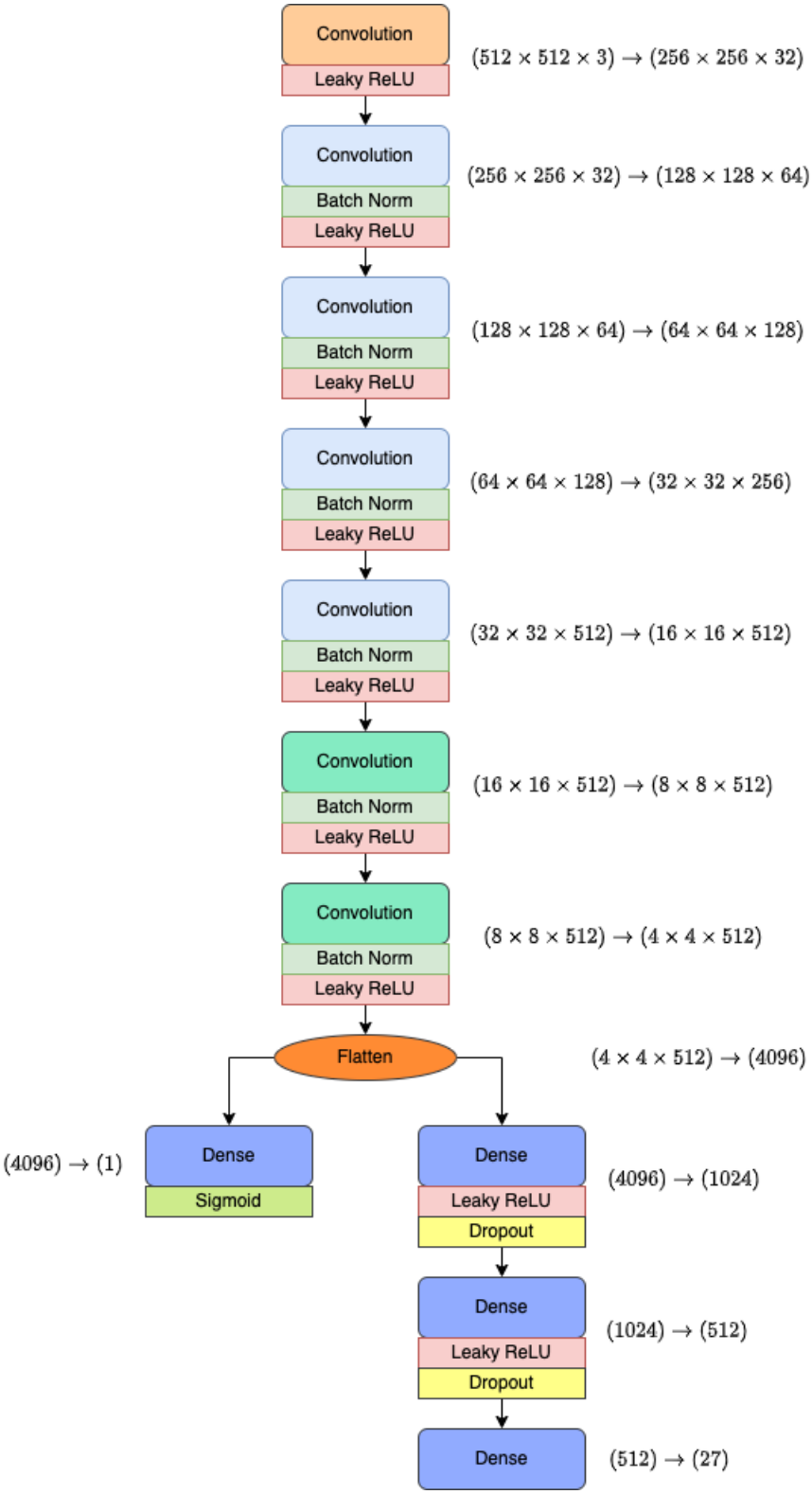

Figure 8: Discriminator Architecture