ProtoDiffusion: Classifier-Free Diffusion Guidance with Prototype Learning

Gulcin Baykal[*] BAYKALG@ITU.EDU.TR
Halil Faruk Karagoz[*] HALILFARUKKARAGOZ@GMAIL.COM
Istanbul Technical University, Computer Engineering Department, Istanbul, Turkey
Taha Binhuraib TAHA.HURAIB@GMAIL.COM
Novus Technologies, Boston, USA
Gozde Unal GOZDE.UNAL@ITU.EDU.TR
Istanbul Technical University, AI and Data Engineering Department, Istanbul, Turkey

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Diffusion models are generative models that have shown significant advantages compared to other generative models in terms of higher generation quality and more stable training. However, the computational need for training diffusion models is considerably increased. In this work, we incorporate prototype learning into diffusion models to achieve high generation quality faster than the original diffusion model. Instead of randomly initialized class embeddings, we use separately learned class prototypes as the conditioning information to guide the diffusion process. We observe that our method, called ProtoDiffusion, achieves better performance in the early stages of training compared to the baseline method, signifying that using the learned prototypes shortens the training time. We demonstrate the performance of ProtoDiffusion using various datasets and experimental settings, achieving the best performance in shorter times across all settings.

Keywords: diffusion models; classifier-free diffusion guidance; prototype learning; image generation

1. Introduction

Diffusion models have gained significant popularity for various image generation tasks due to their visually pleasing and perceptually accurate results. This popularity has especially increased after the Denoising Diffusion Probabilistic Models (DDPMs) are shown to outperform the state-of-the-art GAN-based methods for image synthesis (Dhariwal and Nichol (2021)). Diffusion models are used for different tasks such as unconditional (Ho et al. (2020); Song et al. (2021)) and conditional (Nichol et al. (2022); Rombach et al. (2021)) image synthesis, super-resolution (Ho et al. (2021)), image inpainting (Lugmayr et al. (2022)), image-to-image translation (Saharia et al. (2022a)), text-to-image synthesis (Ramesh et al. (2022); Rombach et al. (2021)), and video synthesis (Ho et al. (2022)). While the diffusion models attain outstanding performance for these tasks, the reasons behind the success of

[*]Equal contribution.

the diffusion models as well as their existing problems are currently important venues of exploration.

Most of the aforementioned works utilize enriched conditioning information, such as text or images, and their high performance can mainly be attributed to the conditioning processes. While the conditioning enables high generation performance that is attained in relatively few sampling steps, handling text-based data is difficult. Therefore, diffusion models using "lighter" conditioning constraints such as class labels are progressively being developed pertaining to their theoretical and practical aspects. Class labels, which can be considered as less enriched conditioning information, are used to guide the generation process in various ways (Dhariwal and Nichol (2021); Ho and Salimans (2021)). However, obtaining stable and fast training of diffusion models using the class labels remains a significant challenge (Nichol et al. (2022)).

To mitigate the problem of fast training of the diffusion models, various solutions for decreasing the training time might be used. Performing parallelized training across multiple GPUs is one way to decrease the time, but it is computationally expensive. Additionally, novel diffusion models for increased stability using a few sampling steps might be designed as in (Salimans and Ho (2022)). In our work, we search the ways of performing fast and stable training of the diffusion models that use the class labels as the conditioning information, without forming a new training method as in (Salimans and Ho (2022)).

Classifier-free diffusion models (Ho and Salimans (2021)) use class labels as conditioning to guide the generation, and they do not need any explicit classifier for the training as in (Dhariwal and Nichol (2021)). For each class in the dataset, the embeddings which represent these classes are learned during the training of the classifier-free diffusion models. These embeddings can be viewed as prototypes, and it is important to learn them effectively such that they can guide the generations with stability and consistency. As prototype learning by itself is a challenging task, we hypothesize that learning the prototypes during the diffusion process disregarding the prototype learning concepts might have significant effects on the consistency of the generations and the speed of the diffusion training. Therefore, exploring the prototype learning methods is crucially the first step to test this hypothesis.

Prototype learning is a favoured machine learning method for learning representative low-dimensional embeddings to represent the diverse classes in the data. More than one prototype might be used to represent a single class to capture the essential characteristics of that class. Generally, they are updated based on specifically designed objectives during the training of the learning model. For the classification of the data, the distances to the learned prototypes are used to assign the data to the closest prototype's class. The effectiveness of the prototype learning is presented for various tasks (Yang et al. (2018); Zhou et al. (2022); Shu et al. (2019); Kim et al. (2021); Xu et al. (2020)), and we find out that carefully designed methods for learning the prototypes actually matter in terms of performance. Therefore, we study on how the prototype learning can be incorporated within the diffusion models to mitigate the stabilized and fast training problems.

In this work, we propose a novel combination of prototype learning and diffusion models called *ProtoDiffusion*. We learn the prototypes of the classes with a separate classifier (Yang et al. (2018)) in $\sim 2-10$ minutes, based on the dataset. Then, we start the training of the diffusion model after we initialize the class embeddings with the learned prototypes, which are used to guide the diffusion process. We demonstrate that the quality of the generated images by the diffusion model is already increased by the beginning of the training. Therefore, our model achieves the same performance ~ 2 times faster than the original diffusion model which indicates a significant amount of reduction in the training time. Hence, training a model to learn the prototypes in at most 10 minutes saves hours in the training of the diffusion model. We summarize our contributions as follows:

- We present that the representativeness of the prototypes used as the conditioning information for the diffusion models might affect the training time and the generation performance.
- We mitigate the training time problem of the diffusion models using the already learned prototypes of the classes.
- We demonstrate that the stability of the training in terms of generation quality is increased, and this stability even leads to better generation quality compared to the baseline.

The ProtoDiffusion model we present in this work outperforms the performance of the baseline methods in terms of Fréchet Inception Distance (FID) (Heusel et al. (2017)) and Inception Score (IS) (Salimans et al. (2016)) on various datasets while we obtain those results faster than the baselines.

2. Related Work

Diffusion Models Diffusion models are promising generative models that have the potential to generate high-quality images with a variety of applications. While some of the works focus on the decreased sampling time (Song et al. (2021); Salimans and Ho (2022); Zhang and Chen (2023)), some other works propose architectural improvements to increase the generation quality such as cascaded diffusion (Ho et al. (2021)). Alongside with these methods, conditioning the diffusion process with enriched information such as text and image to improve the generation quality is a highly studied approach. Generally, the text and the image information are embedded to the latent spaces using pretrained encoders such as CLIP (Radford et al. (2021)) or VQGAN (Esser et al. (2020)) to ease the training of the enormous diffusion models as in DALL-E (Ramesh et al. (2021)), DALL-E 2 (Ramesh et al. (2022)), and Imagen (Saharia et al. (2022b)). Further models utilizing the conditioning information like Latent Diffusion Models (LDMs) (Rombach et al. (2021)) perform the diffusion process over the latent representation instead of the image in order to decrease the need for computational power. Retrieval augmented diffusion models (Blattmann et al. (2022)) condition the image generation on the neighbors of the input image. The neighbor images are retrieved from a database, and they enable achieving a better performance compared to the baseline using a comparably smaller diffusion model. Even though our method employs simpler information as the class label for conditioning, it has a similar intuition to the retrieval augmented diffusion models. As we incorporate the learned prototypes of the classes for conditioning, each of these prototypes captures the various aspects of the class that it represents. Therefore, the learned prototypes have the similar advantage of using the retrieved neighbors which also capture the different aspects of the class that the input image belongs to.

Prototype Learning Prototype learning methods utilizing deep neural networks have gained interest for a wide range of tasks. Zhou et al. (2022) propose a semantic segmentation solution with a prototype view. They learn to represent different semantic classes with prototypes to improve their performance. Xu et al. (2020) employ prototype learning to zero-shot representation learning while Snell et al. (2017) propose prototypical networks for few-shot classification problem. Shu et al. (2019) propose a prototype-based network for open set recognition tasks. Kim et al. (2021) utilize prototypes for saliency feature learning for person search. Oord et al. (2017) use vector quantization to learn a discrete latent space for image reconstruction task that the discrete latent space essentially includes prototypes. Unsupervised (Wu et al. (2018)) and supervised (Guerriero et al. (2018); Mettes et al. (2019); Yang et al. (2018)) classification can be also performed using prototypes. In our work, we need to learn prototypes to effectively represent each of the classes in the dataset. As we have a supervised setting, we use the method proposed by Yang et al. (2018) which performs supervised classification by learning prototypes. After learning such prototypes, we use them to guide the diffusion training.

3. Background

3.1. Diffusion Models

We use Gaussian diffusion models as a generative method. After the diffusion models are introduced by Sohl-Dickstein et al. (2015), numerous models based on the diffusion idea are proposed. In the training phase of the diffusion models, Gaussian noise is progressively added to a sample from the real data distribution $x_0 \sim q(x)$ in T time steps:

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

where x_{t-1} is transformed to x_t by adding Gaussian noise at t^{th} time step. β_t is the variance of the added noise, and $\sqrt{1-\beta_t}$ is the scaling parameter according to a variance schedule. In order to obtain x_t at an arbitrary time step t without t steps iterations, a reparameterization trick enabled by the properties of the Gaussian distribution might be performed:

$$\begin{gathered} q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathrm{I}), \\ x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon. \end{gathered} \tag{2}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\epsilon \sim \mathcal{N}(0, \mathrm{I})$.

The training phase of the diffusion models consists of the denoising process to reverse the noising process in Equation 1. The denoising process is defined as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)). \tag{3}$$

where θ defines the parameters of the neural network that predicts the mean $\mu_\theta(x_t, t)$ and the variance $\Sigma_\theta(x_t, t)$ parameters of the Gaussian distribution. Starting from the white Gaussian noise x_T, the x_0 image is obtained by gradually reducing the noise in T time steps. As a generative model, the main objective of the diffusion models is to learn the reverse process such that a high quality image x_0 can be attained using a random noise vector x_T. In DDPM (Ho et al. (2020)), the mean $\mu_\theta(x_t, t)$ is learned while the variance $\Sigma_\theta(x_t, t)$ is set to constant.

A tractable variational lower bound exists for the optimization of the neural network in Equation 3. Ho et al. (2020) decompose the objective function, and presents that predicting the noise ϵ added in the current time step as in Equation 2 is the best way to parameterize the mean $\mu_\theta(x_t, t)$ of the model:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right). \tag{4}$$

where ϵ_θ is the function approximator to predict the ϵ value added to obtain x_t. The simplified training objective is derived as in (Ho et al. (2020)):

$$L_{simple} = \mathbb{E}_{t \sim [1,T], x_0 \sim q(x), \epsilon \sim \mathcal{N}(0,\mathrm{I})}[||\epsilon - \epsilon_\theta(x_t, t)||^2]. \tag{5}$$

While the DDPM synthesizes unconditional images, guided diffusion models are also available for conditional image generation. Dhariwal and Nichol (2021) propose classifier guidance where the class conditional parameters $\mu_\theta(x_t|y)$ and $\Sigma_\theta(x_t|y)$ are perturbed by the gradients of a classifier $p_\phi(y|x_t)$ that predicts the target class y. The perturbed mean with the guidance scale s is derived as:

$$_\theta(x_t|y) = \mu_\theta(x_t|y) + s\Sigma_\theta(x_t|y)\nabla_{x_t}\log(p_\phi(y|x_t)). \tag{6}$$

Even though the classifier guidance improves the quality of the images, there are a few problems of the classifier guidance. As the denoising process starts with highly noised input and proceeds with noisy images at most of the time steps, the classifier should be robust to noise. While obtaining such a classifier is challenging, predicting a class label does not require using the most of the information in the data. Therefore, taking the gradients of such a classifier might mislead the generation direction.

Ho and Salimans (2021) propose classifier-free guidance method that does not require a separate classifier. The conditioning information y is periodically utilized while it is dropped out at the remaining time. Therefore, a single model can be used for both unconditional and conditional generation. Ho and Salimans (2021) derive that the unconditional $\epsilon_\theta(x_t, t)$ and conditional $\epsilon_\theta(x_t, t, y)$ estimations can be used to represent the gradients of the classifier as:

$$\begin{aligned} \nabla_{x_t}\log(y|x_t) &= \nabla_{x_t}\log p(x_t|y) - \nabla_{x_t}\log p(x_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \left(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)\right). \end{aligned} \tag{7}$$

Equation 7 implies that an implicit classifier can eliminate the need for an explicit classifier, and Ho and Salimans (2021) report better results with the classifier-free guidance compared to the explicit classifier guidance.

In our work, we enhance the classifier-free guidance method with prototype learning in order to improve the performance while decreasing the training time.

3.2. Prototype Learning

Prototype learning is a classical and popular representation learning method for numerous tasks as described in Section 2. Specifically, prototype learning is a plausible choice to categorize the labeled data or to cluster the unlabeled data. For a classification task, prototypes that define different classes might be learned through a parameter optimization of feature extractor convolutional neural networks (CNNs). The extracted features by the CNN might be compared to the prototypes of the classes to predict the target class instead of using softmax. The CNN parameters and the prototypes might be jointly trained.

Given C different classes in the dataset and K different prototypes for each class, a prototype is denoted as e_{ij} where $i \in 1, 2, \ldots, C$ and $j \in 1, 2, \ldots, K$. The number of prototypes for each class K is a design choice. The input data x is given to a CNN f_ψ for feature extraction, and the features of x might be compared to the prototypes for the class assignment:

$$x \in \text{argmax}_{i=1}^{C}(-\text{min}_{j=1}^{K}||f_\psi(x) - e_{ij}||_2^2). \tag{8}$$

Via Equation 8, the input is assigned to the class which consists of the closest prototype to the input's features. In order to jointly train the feature extractor f_ψ and the codebook $\mathcal{M}$, which comprises the prototypes e's, in an end-to-end manner, carefully designed loss functions might be used. Yang et al. (2018) introduce several loss functions for the training, and compares their effects on the accuracy of the classification. The withstanding loss function is defined as:

$$p(x \in e_{ij}|x) = \frac{e^{-\gamma||f_\psi(x)-e_{ij}||_2^2}}{\sum_{k=1}^{C}\sum_{l=1}^{K} e^{-\gamma||f_\psi(x)-e_{kl}||_2^2}}, \tag{9}$$

$$\mathcal{L}(\psi, \mathcal{M}) = -\log\left(\sum_{j=1}^{K} p(x \in e_{yj}|x)\right) + \lambda||f_\psi(x) - e_{\text{closest}}||_2^2. \tag{10}$$

The first term in Equation 10 minimizes the cross entropy loss using a sample (x, y) and its probability of belonging to the prototype e_{ij} defined in Equation 9. In Equation 9, the γ parameter controls the sharpness of the assignment, and the probability of belonging to a prototype is calculated using the distance to the features extracted by f_ψ.

While minimizing the first term in the loss function decreases the sample's distance to the closest prototype compared to the distances to the remaining prototypes, it is not enough to minimize the distance between the sample and the closest prototype. Therefore, a regularization term forcing the minimum distance between the sample and the closest prototype e_{closest} is added to the optimization as in the second term in Equation 10. Training f_ψ and $\mathcal{M}$ using this loss function leads to better classification results on numerous datasets as presented in (Yang et al. (2018)).

In our work, we exploit the learned prototypes that represent the classes in the data space in order to enhance the classifier-free diffusion guidance.

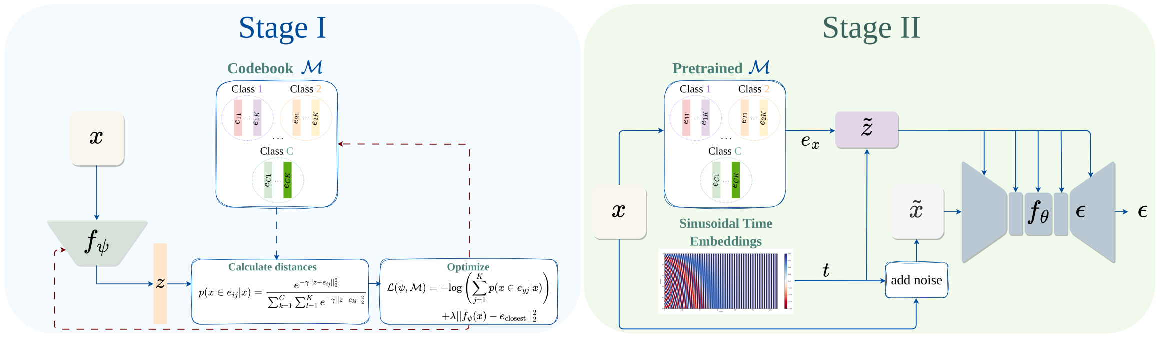

Stage I
Codebook $\mathcal{M}$
Class 1
Class 2
Class C
x
f_ψ
z
Calculate distances
$p(x \in e_{ij}|x) = \frac{e^{-\gamma||z-e_{ij}||_2^2}}{\sum_{k=1}^{C}\sum_{l=1}^{K} e^{-\gamma||z-e_{kl}||_2^2}}$
Optimize
$\mathcal{L}(\psi, \mathcal{M}) = -\log\left(\sum_{j=1}^{K} p(x \in e_{yj}|x)\right) + \lambda||f_\psi(x) - e_{\text{closest}}||_2^2$
Stage II
Pretrained $\mathcal{M}$
e_x
$\tilde{z}$
Sinusoidal Time Embeddings
t
add noise
$\tilde{x}$
f_θ
ϵ

Figure 1: Training stages of ProtoDiffusion. At the first stage, a codebook $\mathcal{M}$ consisting of the prototypes are trained. Then, the pretrained prototypes and the time embeddings are concatenated as $\tilde{z}$ to guide the diffusion process based on the class and the time information. Lastly, the noisy image $\tilde{x}$ is forwarded to a U-Net like network for the noise prediction.

4. Method

Our model ProtoDiffusion is presented in Figure 1. We conduct two staged training such that at the first stage, we obtain the class based prototypes while at the second stage, we train the diffusion model.

At Stage I, we follow the prototype learning method (Yang et al. (2018)) explained in Section 3.2. Based on the dataset, the number of the classes in the codebook $\mathcal{M}$ may vary. However, to be able to use the class embeddings in the diffusion process as in (Ho and Salimans (2021)), we need a single embedding to represent the class. Therefore, we learn a single prototype per class in Stage I which is practically allowed as shown in (Yang et al. (2018)). As illustrated in Stage I of Figure 1, we train the feature extractor f_ψ and the codebook $\mathcal{M}$ in an end-to-end manner.

At Stage II, we follow the proposed diffusion training proposed by Ho and Salimans (2021) with specific differences as follows. Instead of initializing the class embeddings randomly, we utilize the prototypes that we learned in Stage I, and initialize the class embeddings with our prototypes. Then, we concatenate the sampled sinusoidal time embedding t with the prototype e_x that represents the class label of the input x, and obtain $\tilde{z}$. As explained in Section 3.1, the denoising process is learned by a neural network, and it is optimized by predicting the added noise to the input x in t time steps. Stage II in Figure 1 consists of a U-Net like neural network f_θ to predict the noise ϵ of the noisy image $\tilde{x}$. During the training, various time steps are sampled in different iterations, and f_θ is optimized to predict the noise values based on these time steps. In order to guide the diffusion process, the class embedding and the time step are also given to the specific layers of f_θ.

In the baseline model, the randomly initialized class embeddings are optimized by the diffusion process without explicit optimizations measuring the representativeness of these embeddings. Therefore, we suspect that attaining prototypical class embeddings with this setting might be a bottleneck for the training time and the generation quality. As a result, we start the diffusion training with already representative class embeddings, and achieve a

Table 1: GPU hours comparison of the baseline model and ProtoDiffusion model across datasets.

Method	CIFAR10	STL10	Tiny ImageNet
Classifier-Free Guidance	29.33	88.89	233.33
ProtoDiffusion	**13.33**	**66.67**	**166.67**

better performance in a shorter time period. We incorporate the learned prototypes to the diffusion process in two ways.

The first method we use is the ProtoDiffusion that utilizes the class prototypes as they are, and not update them during the diffusion training. We call it ProtoDiffusion (Frozen). The reason we follow this method is that the class prototypes are already learned, and they may not need further optimization to guide the diffusion training. Therefore, the gradients are not allowed to flow to the class embeddings to update them.

The second method we use is the ProtoDiffusion (Unfrozen) where the class prototypes are also updated during the diffusion training. Comparing these two methods with the baseline method demonstrates not only the effects of the prototypes to the diffusion training, but also the effects of the diffusion training to the representativeness of the class embeddings.

5. Experiments

We conduct our experiments on CIFAR10 (Krizhevsky and Hinton (2009)), STL10 (Coates et al. (2011)), and Tiny ImageNet (Le and Yang (2015)) datasets using 4 NVIDIA A100 GPUs. We present all of the experimental settings in Appendix A.

We compare the GPU hours of the baseline diffusion model and our model ProtoDiffusion until they reach their best performance in Table 1. The results for the ProtoDiffusion models consist of the prototype learning time and the diffusion training time. We learn the prototypes for the datasets using a single NVIDIA A100 GPU. We obtain the prototypes of CIFAR10 and STL10 in ~ 2 minutes and Tiny ImageNet in ~ 13 minutes. ProtoDiffusion achieves the best performance considerably faster than the baseline model for all datasets, which supports our hypothesis for faster training enabled by the learned prototypes.

We use an existing method for prototype learning proposed by Yang et al. (2018). In order to evaluate the performance of the learned feature extractor f_ψ, we project the data of the CIFAR10 dataset to a 2D space using PCA. As Figure 2 shows, f_ψ learns to cluster the data x into classes accurately, and the learned prototypes in ProtoDiffusion are the cluster centers of these clusters.

We evaluate the performance of the baseline method and the ProtoDiffusion methods using FID (Heusel et al. (2017)) and IS (Salimans et al. (2016)) in Table 2. As the lower FID and higher IS indicate better performance, we show that ProtoDiffusion methods outperform the baseline method for all of the datasets, except for the IS measure with the Tiny ImageNet. We observe that the ProtoDiffusion (Unfrozen) method is mostly better than the ProtoDiffusion (Frozen) method. These results demonstrate that the parameter update of the class embeddings caused by the diffusion process enhances the stability and positively impacts training, indicating that the gradient updates from the diffusion process allows the

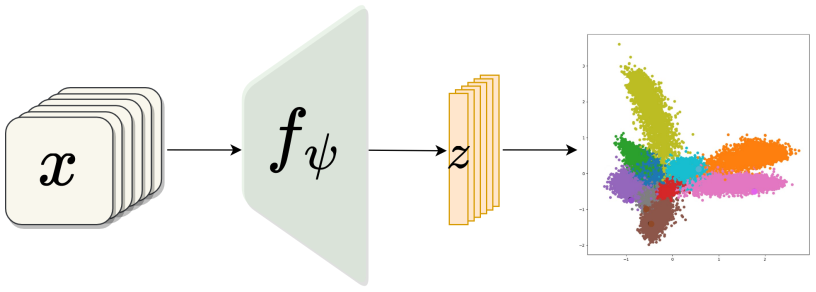


Figure 2: CIFAR10 clusters are depicted on the right after the prototype learning process.

Table 2: Comparison of the models in terms of FID (↓) and IS (↑) on different datasets.

Method	**CIFAR10**		**STL10**		**Tiny ImageNet**	
	FID(↓)	*IS(↑)*	*FID(↓)*	*IS(↑)*	*FID(↓)*	*IS(↑)*
Classifier-Free Guidance	11.82	8.85	33.86	11.33	11.49	**28.67**
ProtoDiffusion (Frozen)	9.19	8.95	23.64	11.36	**9.56**	24.82
ProtoDiffusion (Unfrozen)	**8.55**	**9.01**	**22.57**	**11.51**	9.94	23.07

embeddings to capture more of the latent space that represent the class labels. The FID, which compares the distribution of generated images with the distribution of real images in a given dataset, is a commonly preferred metric over the IS, which evaluates only the distribution of generated images. For the Tiny ImageNet dataset, although the IS scores of ProtoDiffusion models are below that of the baseline, both ProtoDiffusion models achieve lower FID scores than the baseline.

We monitor the training progress of all models so that we can decide whether to stop the training or not. Figure 3 presents the FID scores calculated during the training of the models using the CIFAR10 dataset. For each 100 epoch, we check the FID values and compare them. The first important observation that we note is the large difference between the FID values of the baseline and the ProtoDiffusion models at the beginning of the training. We observe that the learned prototypes' effect on the generation quality at the beginning is considered valuable. Therefore, we manage to obtain the best performance faster than the baseline model. The green area highlights the epoch that we decide to stop the training for the ProtoDiffusion (Unfrozen) model. We compare the FID scores at the same epoch, and obtain a better score compared to that of the baseline. Therefore, we stop the training of the ProtoDiffusion (Unfrozen) at 500^{th} epoch while we continue to train the other models. We demonstrate that the ProtoDiffusion (Frozen) does not present a stabilized training compared to the ProtoDiffusion (Unfrozen). These results justify our conclusion that the frozen prototypes may not be adapted to the diffusion training in the same time frame as the ProtoDiffusion (Unfrozen) method since they are not updated with the gradients of the diffusion process. As we attain the best performance two times faster than the baseline, this also affects the computational usage demonstrated in Table 1.

As an ablation study, we observe the effects of the prototype dimensionality on the performance in terms of generating quality. Table 3 presents the FID results on the CIFAR10

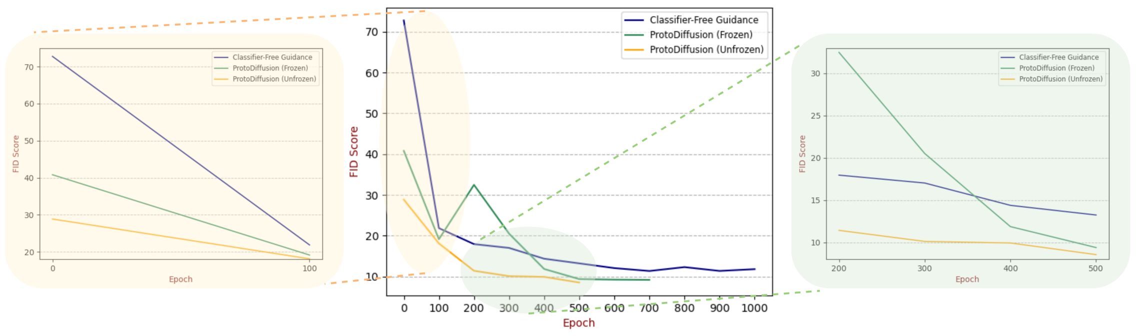


Figure 3: FID scores of all models during the training. While the yellow area highlights the benefits of using the learned prototypes to the starting, the green area highlights that the best results obtained for the dataset is achieved way faster than the baseline.

Table 3: The effects of the different prototype dimensionalities on CIFAR10 in terms of FID.

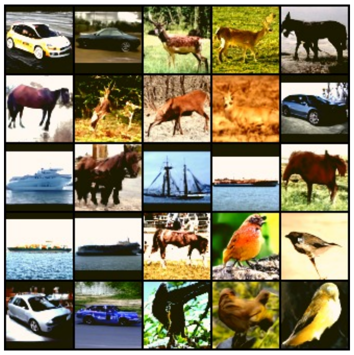
Method	**32**	**64**	**128**
Classifier-Free Guidance	12.51	9.89	11.82
ProtoDiffusion (Frozen)	9.89	12.48	9.98
ProtoDiffusion (Unfrozen)	**9.25**	**9.15**	**8.55**

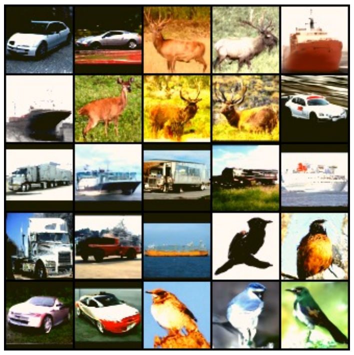
dataset using 32, 64, and 128 as the prototype dimensionalities. While ProtoDiffusion (Unfrozen) outperforms the other methods for all settings, using 128 as the prototype dimensionality gives the best performance. Since the prototypes with higher dimensionalities can accommodate extra information about the underlying data space, it is reasonable to obtain a better performance in the latter case. Therefore, we choose 128 among the other options for all experiments.

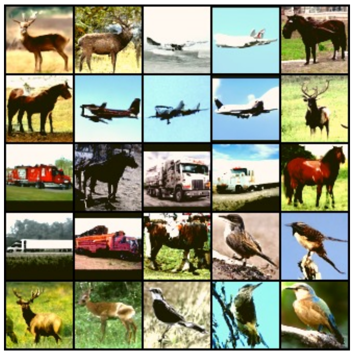
In addition to the numerical evaluations, we perform visual assessments of the methods. Figure 5 demonstrates the generated samples from the STL10 dataset using the baseline method and ProtoDiffusion methods. We observe that the generated samples by the ProtoDiffusion methods depict structural consistency compared to the baseline's samples. Additionally, the samples of almost all classes show good quality, while the generated samples from some classes such as the horse seem to appear more flawed.

We present generated samples from the Tiny ImageNet dataset using the best performing ProtoDiffusion method in Figure 5. As the results depict, the ProtoDiffusion model produces samples from the diverse classes of the dataset showing good quality.

Figure 4: Generated samples from the STL10 dataset using the baseline and ProtoDiffusion.


(*a*) Regular Classifier-Free Diffusion Guidance.


(*b*) ProtoDiffusion (Frozen).


(*c*) ProtoDiffusion (Unfrozen).

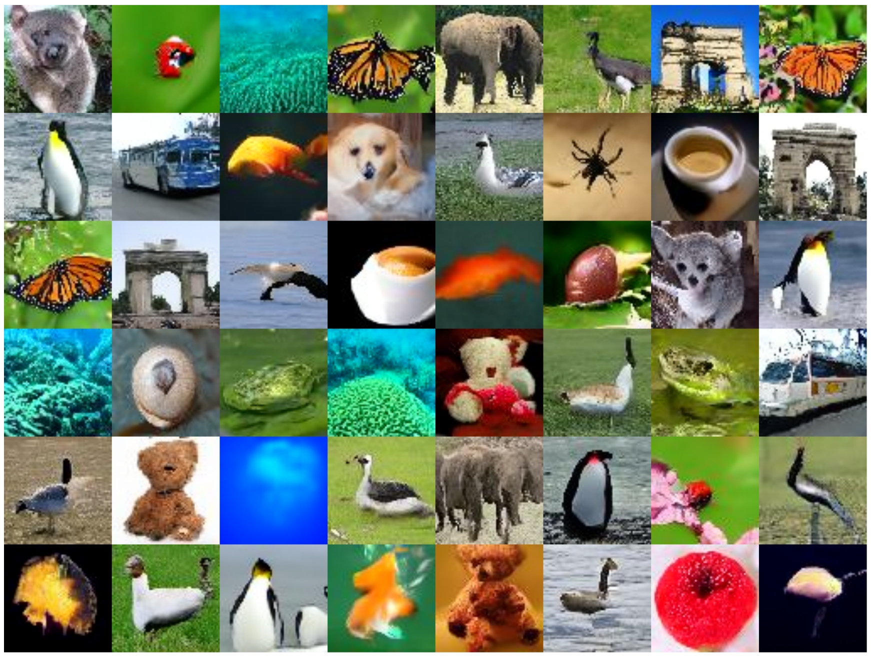


Figure 5: Generated samples from Tiny ImageNet dataset using ProtoDiffusion.

6. Conclusion

In this work, we propose a method called ProtoDiffusion that incorporates the representative class prototypes into the diffusion training as the conditioning information. We observe that the learned prototypes affect the training even at the beginning, and the improved performances in terms of generation quality are achieved faster than the baseline method. We conduct several experiments to visualize the representativeness of the prototypes, to view the effects of the prototype dimensionality, and to monitor the diffusion training progress so that we can derive conclusions about the significance of the prototype learning.

Although the ProtoDiffusion models achieve significantly better performances way faster than the baseline for all datasets, we note that there is room for improvement for the Tiny ImageNet dataset. The effects of the prototype learning for the Tiny ImageNet dataset is not as clear as compared to the other datasets in terms of the chosen metrics. One reason might be the representativeness of the prototypes itself as they are attained by the same simple neural network which may not be adequate in characterizing the complexity of the given dataset. Therefore, as future work, more complex datasets like Tiny ImageNet should be further studied to analyze the effects of the prototype learning.

Acknowledgments

In this work, Gulcin Baykal was supported by Google DeepMind Scholarship Program at ITU. Computing resources used during this research were provided by the National Center for High Performance Computing of Turkey (UHeM) under grant number 1007422020.

References

Andreas Blattmann, Robin Rombach, Kaan Oktay, and Björn Ommer. Retrieval-augmented diffusion models, 2022.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2020.

S. Guerriero, B. Caputo, and T. E. J. Mensink. Deep nearest class mean classifiers. In *International Conference on Learning Representations Workshops*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Martin Heusel, Hubertus Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Conference on Neural Information Processing Systems*, pages 6626–6637, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 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. *arXiv preprint arXiv:2106.15282*, 2021.

Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022.

H. Kim, Sunghun Joung, Ig-Jae Kim, and Kwanghoon Sohn. Prototype-guided saliency feature learning for person search. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4863–4872, 2021.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

Andreas Lugmayr, Martin Danelljan, Andrés Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11451–11461. IEEE, 2022.

Pascal Mettes, Elise van der Pol, and Cees G. M. Snoek. *Hyperspherical Prototype Networks.* 2019.

Alexander Quinn 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. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 16784–16804. PMLR, 2022.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

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. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092, 2021.

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

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021.

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. 2022a.

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, 2022b.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Yu Shu, Yemin Shi, Yaowei Wang, Tiejun Huang, and Yonghong Tian. P-odn: Prototype-based open deep network for open set recognition. *Scientific Reports*, 10, 2019.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. NIPS'17, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for zero-shot learning. In *NeurIPS*, 2020.

Hong-Ming Yang, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Robust classification with convolutional prototype learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3474–3482. Computer Vision Foundation / IEEE Computer Society, 2018.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023.

Tianfei Zhou, Wenguan Wang, Ender Konukoglu, and Luc Van Gool. Rethinking semantic segmentation: A prototype view. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 2572–2583. IEEE, 2022.

Appendix A. Experimental Settings

A.1. Prototype Learning

In the prototype learning, we use a basic ResNet-18 He et al. (2016) architecture as the feature extractor. We train the models 20 epochs for all datasets as we achieve the highest accuracy within the first 20 epochs. As presented in Table 3, we train different networks to learn prototypes with various dimensionalities. We use 512 batch size and Adam optimizer with 0.0001 learning rate.

A.2. Diffusion Training

In the diffusion training process, the U-Net architecture is utilized to effectively predict the noise present in Denoising Diffusion Probabilistic Models (DDPMs). The initial channel size of the model is set to 64. For the Encoder part, which is responsible for down-sampling the input, there are 4 blocks, each consisting of 2 ResNet layers. These layers are equipped with a 0.1 Dropout layer. The channel size of each block progressively increases using multiplication factors of 2, 2, 4, and 4. This hierarchical representation enables the capturing of both fine-grained and high-level features. The Decoder part, responsible for up-sampling, mirrors the Encoder by utilizing Upsample blocks with the same multiplication factors. However, the channel sizes of the Decoder's blocks gradually decrease, ultimately resulting in an RGB image with 3 channels.

For all experiments, we adopt the AdamW optimizer with a fixed learning rate of 0.0002. The batch size of 128, which corresponds to one GPU, is used for CIFAR10 experiments. For the STL10 and Tiny ImageNet datasets, a batch size of 64 (one GPU) is employed.

In order to decrease the computational and the structural complexity during our experiments, we resize the image size of the STL10 dataset from 96 to 64. The original image sizes of the CIFAR10 and Tiny ImageNet datasets remain unchanged at 32 and 64, respectively.