Class-Prototype Conditional Diffusion Model with Gradient Projection for Continual Learning

Khanh Doan¹, Quyen Tran¹, Tung Lam Tran¹, Tuan Nguyen², Dinh Phung², and Trung Le²

¹ VinAI Research {v.khanhdn10, v.quyentt15, v.lamtt12}@vinai.io
² Monash University {tuan.ng, dinh.phung, trunglm}@monash.edu

Abstract. Mitigating catastrophic forgetting is a key hurdle in continual learning. Deep Generative Replay (GR) provides techniques focused on generating samples from prior tasks to enhance the model's memory capabilities using generative AI models ranging from Generative Adversarial Networks (GANs) to the more recent Diffusion Models (DMs). A major issue is the deterioration in the quality of generated data compared to the original, as the generator continuously self-learns from its outputs. This degradation can lead to the potential risk of catastrophic forgetting (CF) occurring in the classifier. To address this, we propose the Gradient Projection Class-Prototype Conditional Diffusion Model (GPPDM), a GR-based approach for continual learning that enhances image quality in generators and thus reduces the CF in classifiers. The cornerstone of GP-PDM is a learnable class prototype that captures the core characteristics of images in a given class. This prototype, integrated into the diffusion model's denoising process, ensures the generation of high-quality images of the old tasks, hence reducing the risk of CF in classifiers. Moreover, to further mitigate the CF of diffusion models, we propose a gradient projection technique tailored for the cross-attention layer of diffusion models to maximally maintain and preserve the representations of old task data in the current task as close as possible to their representations when they first arrived. Our empirical studies on diverse datasets demonstrate that our proposed method significantly outperforms existing state-of-the-art models, highlighting its satisfactory ability to preserve image quality and enhance the model's memory retention.

Keywords: Continual Learning \cdot Generative replay \cdot Diffusion model

1 Introduction

Continual Learning (CL) is a process where neural networks progressively acquire and accumulate knowledge from sequentially arriving tasks over time. Mimicking human learning, CL aims to develop models capable of swiftly adapting to evolving real-world environments. The primary challenge in CL is addressing Catastrophic Forgetting (CF) [11], which involves the model's need to learn new

tasks without losing previously acquired knowledge, particularly when access to old data is not possible.

Continual learning strategies primarily focus on overcoming CF by preserving knowledge from previous tasks. This is particularly crucial in situations where privacy concerns prevent the retention of old raw data and the task ID remains unidentified during testing. In such cases, methods that employ generative models, specifically Generative Replay (GR), are notably effective for learning the distribution of past data. GR approaches [12,19,35] usually adopt a two-part system comprising a generator and a classifier. The generator's role is to generate samples from prior tasks, which are then used to train the classifier. The classifier is designed to not only classify samples from the current task but also effectively categorize the generated samples, thereby helping to mitigate catastrophic forgetting.

In the context of GR approaches, initial efforts utilized GAN and Variational Auto Encoder (VAE) architectures, which operate in a single-directional manner (from generator to classifier) and recycle generated samples, leading to a gradual decline in image quality. The recent DDGR method [12] leverages the benefits of Diffusion Models (DMs) to more effectively prevent forgetting in generative models. DDGR employs a bi-directional interaction between the generator and the classifier. In the generator-to-classifier direction, the generator creates high-quality samples by denoising the noisy images with class information from previous tasks, which are then utilized to train the classifier. Conversely, in the classifier-to-generator direction, the classifier which is pretrained on previous tasks guides the generator in synthesizing high-quality samples from these tasks. However, our experimental studies revealed a noticeable decrease in the quality of generated images when the model was trained on a lengthy sequence of tasks, adversely affecting its capacity to retain acquired knowledge from earlier tasks.

To address the challenge of catastrophic forgetting in Continual Learning more efficiently, this paper introduces the Gradient Projection Class-Prototype conditional Diffusion Model (GPPDM). Specifically, considering the bi-directional relationship operated under GR strategies as described above, in our model, the classifier aids the diffusion model in learning additional suitable prototypes. These prototypes will help preserve abstract information of old data, thus reducing the disadvantage of DDGR in gradually forgetting old knowledge. Consequently, when generating replay data in new tasks for training the classifier, the diffusion model then makes use of the information from both the labels and the corresponding learned prototypes to get a better quality of generated images. Moreover, after training a class y, this label and its prototype, which are inputs of the DM, will become invariant components. Therefore, to exploit learned prototypes more efficiently and further limit CF, we desire to preserve the corresponding knowledge on DM more effectively to maintain the representations of old task data in the current task as close as possible to their representations when they first arrived. Fortunately, this desideratum aligns with the strategy of GPM [33], which inspires us to design a novel approach to inject GPM into DM, specifically Cross-Attention layers. This approach is not only streamlined but also effective

in avoiding forgetting effectively. In summary, our primary contributions can be outlined as follows:

- We propose an efficient training technique for the diffusion model by learning class prototypes, which capture the most representative examples of classes from previous tasks. This approach facilitates the generation of high-quality images through class-prototype conditional denoising.
- To further reduce forgetting, we suggest a novel design of GPM for DM. This approach creates an overall harmony when combined with learned class prototypes in preserving old knowledge effectively, under a simple implementation.
- The comprehensive results from benchmark datasets show that our proposed method significantly outperforms the current state-of-the-art model, achieving higher average accuracy and lower average forgetting rate.

2 Related Work

2.1 Continual Learning

In general, CL techniques combating catastrophic forgetting are divided into three main groups: regularization-based, parameter isolation, and memory-based approaches.

Regularization-based approaches incorporate constraints to parameter updates [4,41] or enforce consistency with previous model outputs [24,43].

Parameter isolation approaches aim to tackle the problem by assigning each task to a dedicated portion of the network, preventing the overwriting of weights and the loss of information. When no architectural size constraints apply, it is possible to grow new branches for new tasks while freezing previous task parameters [27, 34, 40].

Memory-based approaches exhibit superior performance across diverse settings, particularly excelling in scenarios where task boundaries during training and task IDs during testing are unspecified, providing increased flexibility. Particularly, these approaches utilize episodic memory to store past data [5,6,31,33].

2.2 Generative Replay Continual Learning

Generative replay (GR) leverages a generative model [13,16,20] to create a replay memory for old tasks, mitigating catastrophic forgetting. Classical GR methods typically employ VAE [20] or GAN [13] as the generator for Continual Learning (CL) [1,28,29,39,42]. For instance, DGR [35] uses GAN to generate previous samples for data replaying. MeRGANs [7] addresses forgetting in GANs and excels in CL. Additionally, the mnemonics training framework proposed by [25] generates optimizable exemplars. Building on the success of diffusion models [16,18], [12] suggests using diffusion models as a generative replay for Class Incremental

4 Doan et al.

Learning (CIL). Despite achieving success in mitigating the catastrophic forgetting of classifiers in CL, existing generative replay approaches using VAE, GAN, and diffusion models still face catastrophic forgetting themselves, resulting in a reduction in the quality of generated images across tasks. This work proposes an efficient workaround to alleviate the catastrophic forgetting of a diffusion model-based generative replay. Key components of our approach include class-conditioned prototypes that summarize class information to guide the diffusion model and diversity exploration with neighbor mix style to enhance the diversity of generated images.

3 Generative Replay CL with Diffusion Models

In what follows, we present the problem setting of task incremental CL and the technicality of generative replay CL with diffusion models.

Problem Setting of Class Incremental CL: In Class Incremental Learning (CIL), a seamless and continuous sequence of tasks, denoted as $\mathcal{D}^t = (\mathbf{x}_i^t, y_i^t)_{i=1}^{N^t}$ for $t = 1, \ldots, T$, arrives in the system. Here, the image $\mathbf{x}_i^t \in \mathbb{R}^{H \times W \times C}$ and the label $y_i^t \in \mathcal{Y}^t$ (the set of labels for task t). The goal is to train a deep neural network f_{ϕ} initialized as f_{ϕ}^0 . Upon arrival of each task \mathcal{D}^t , the model needs to adapt from f_{ϕ}^{t-1} to f_{ϕ}^t , mainly relying on the current data in \mathcal{D}^t without revisiting the data from previous tasks $\mathcal{D}^{1:t-1}$. One of the most significant challenges in TIL is catastrophic forgetting, where the current model f_{ϕ}^t tends to easily forget information from the oldest tasks, hence leading to poor predictive performances on these tasks.

Generative Replay with Diffusion Models: To address catastrophic forgetting, generative models like GAN [13] and VAE [20] are commonly employed to generate replay memory for old tasks [19,35]. However, GANs often encounter the mode collapsing problem, and VAEs may generate blurry images [14]. Recently, diffusion models [17,36] have emerged as state-of-the-art generative models capable of producing high-quality and diverse images. To overcome the inherent issues of GANs/VAEs and capitalize on the advantages of diffusion models, [12] proposes using diffusion models as a generative replay for TIL. Specifically, at the end of task t, it trains a diffusion model $\epsilon_{\theta}^{t}(\mathbf{x}_{k}, k)$ on the current data \mathcal{D}^{t} and the replay memory of the previous tasks $\mathcal{M}^{1:t-1}$. This trained diffusion model $\epsilon_{\theta}^{t}(\mathbf{x}_{k}, k)$ is then employed to generate the replay memory $\mathcal{M}^{1:t}$ for the next task.

4 Our Proposed Approach

4.1 Motivations

Catastrophic Forgetting (CF) in classifiers poses a significant challenge in CIL. While applying generative models to replay old data can alleviate CF in classifiers, the generative models themselves may face CF within the context of CL. This occurs because, at the end of each task t, the generative model is trained based on

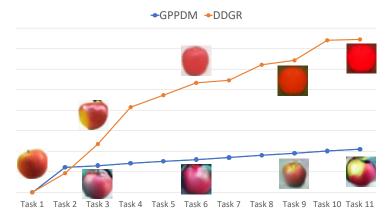


Fig. 1: FID scores across tasks for generated images corresponding to the first task dataset of the baselines DDGR [12] and our proposed GPPDM. The illustration belongs to the label class "Apple", whereas the leftmost one is the real photo. As can be observed, for DDGR, the quality of generated apples deteriorates significantly across tasks. Consequently, the generated apples of DDGR in some last tasks become hardly recognizable, while the generated apples of our proposed GPPDM maintain their quality more efficiently. Finally, FID scores across tasks for our GPPDM are significantly better than DDGR, especially for later tasks when the gap becomes more pronounced.

the new task data \mathcal{D}^t and the replay memory $\mathcal{M}^{1:t-1}$. The latter is generated from the generative model trained at the end of task t-1, using \mathcal{D}^{t-1} and the previous replay memory $\mathcal{D}^{1:t-2}$. Notably, the quality of old data in the generative replay memory (e.g., \mathcal{M}^1) diminishes task by task, making it increasingly challenging for current and future diffusion models to maintain high-quality representations on this data.

While it has been demonstrated in [12] that Diffusion Models as generative replay can assist in mitigating the catastrophic forgetting of generative models in CL, our observations indicate that challenges still persist. As illustrated in Figure 1, the quality of generated images for task 1's data significantly diminishes across tasks. Consequently, the generated images for some of the last tasks exhibit very low quality, making it difficult for the current model to recall old data and exacerbating the issue of catastrophic forgetting.

Our primary approach to alleviate the CF of diffusion models involves learning class prototypes that succinctly summarize data within a given class. Subsequently, during both training and inference of new data, the diffusion models are conditioned on these class prototypes. It is worth noting that these class prototypes are learnable variables, which serve as a reserved information channel to effectively remind diffusion models of class concepts, contributing to reducing the CF across tasks. Moreover, to further mitigate the CF of diffusion models, we propose a gradient projection technique [33] tailored for the cross-attention layer of diffusion models to maximally maintain and preserve the representations of old task data in the current task as close as possible to their representations when they first arrived.

4.2 Class-Prototype Conditional Diffusion Probabilistic Model

When data $(\mathbf{x}, y) \in \mathcal{D}^t$ of task t is arriving, thanks to the high-quality images in \mathcal{D}^t , we can train a diffusion model (DM) parameterized by θ to generate good images for task t. However, while learning a later task t' > t, to avoid forgetting old knowledge, we need to recall a small replay memory \mathcal{M}^t of each task t < t', which is generated from the DM being trained. It is worth noting that the quality of \mathcal{M}^t gradually degrades, leading to the cumulative forgetting of data of class y of task t when trained and recalled in a future task t'. To alleviate this CF and improve the quality of generated samples, we propose using additional learnable prototypes as an essential condition to preserve the most important and general information of old data. In particular, our diffusion model p_{θ} produces an image \mathbf{x} based on not only label y, but also the corresponding prototype $\mathbf{c}(y)$. For each label y, $\mathbf{c}(y)$ is an important factor that can capture and maintain abstract properties corresponding to data of this class. In our implementation, we treat it as a learnable factor and optimize it concurrently with θ during training. In what follows, we provide details of our class-prototype diffusion model.

Forward process. Given a predefined diffusion process of L discrete timesteps, let $\mathbf{x}_0 \sim q(\mathbf{x}_0|y)$ be clean data samples, where $q(\mathbf{x}_0|y)$ represents the class-conditional data distribution over \mathbb{R}^d . Conditioned on \mathbf{x}_0 and y, the joint distribution of the sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L$ can be factorized using the chain rule of probability and the Markov property as follows:

$$q(\mathbf{x}_{1:L}|\mathbf{x}_0, y) = \Pi_{l=1}^L q(\mathbf{x}_l|\mathbf{x}_{l-1}, y). \tag{1}$$

Reverse process. We begin the reverse process with a prior distribution $p(\mathbf{x}_L|y,\mathbf{c}(y)) = \mathcal{N}(\mathbf{x}_L;\mathbf{0},\mathbf{I})$, and then progressively denoise the noisy image \mathbf{x}_L to generate new data that belongs to class y. The process is encapsulated in the following equation:

$$p_{\theta}(\mathbf{x}_{0:L}|y,\mathbf{c}(y)) = p(\mathbf{x}_{L}|y,\mathbf{c}(y))\Pi_{l=1}^{L}p_{\theta}(\mathbf{x}_{l-1}|\mathbf{x}_{l},y,\mathbf{c}(y)). \tag{2}$$

Training objective. Minimize the cross-entropy (CE) between $q(\mathbf{x}_0|y)$ and $p_{\theta}(\mathbf{x}_0|y, c(y))$ to learn $p_{\theta}(\mathbf{x}_{l-1}|\mathbf{x}_l, y, \mathbf{c}(y))$:

$$CE\left(q\left(\mathbf{x}_{0}|y\right), p_{\theta}\left(\mathbf{x}_{0}|y, \mathbf{c}(y)\right)\right) = -\mathbb{E}_{q\left(\mathbf{x}_{0}|y\right)}\left[\log p_{\theta}\left(\mathbf{x}_{0}|y, \mathbf{c}(y)\right)\right]$$

$$\leq \mathbb{E}_{q\left(\mathbf{x}_{0:L}|y\right)}\left[\log \frac{q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right)}{p_{\theta}\left(\mathbf{x}_{0:L}|y, \mathbf{c}(y)\right)}\right].$$
(3)

This leads to the optimization problem:

$$\min_{\theta, \mathbf{c}(y)} \mathbb{E}_{q(\mathbf{x}_{0:L}|y)} \left[\log \frac{q(\mathbf{x}_{0:L} \mid \mathbf{x}_{0}, y)}{p_{\theta}(\mathbf{x}_{0:L}|y, \mathbf{c}(y))} \right]. \tag{4}$$

Using the ϵ -network $\epsilon_{\theta}(\mathbf{x}_{l}, l, y, \mathbf{c}(y))$ to predict the noise ϵ , we reach the following optimization problem for diffusion training:

$$\min_{\boldsymbol{\theta}, \mathbf{c}(y)} \mathcal{L}_d := \mathbb{E}_{\boldsymbol{\epsilon}, \mathbf{x}_0, l, y} \left[\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}} \left(\mathbf{x}_l, l, y, \mathbf{c}(y) \right) \|_2^2 \right]. \tag{5}$$

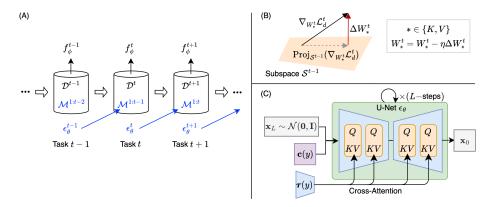


Fig. 2: (A) GPPDM framework for CL with generative replay: During task t, the classifier f_{ϕ}^{t} trains on data from the current task \mathcal{D}^{t} and on data from the previous task $\mathcal{M}^{1:t-1}$ generated by the diffusion model to reduce catastrophic forgetting. (B) Diffusion training with Gradient Projection: To maintain performance on images generated from previous tasks, the gradients of the diffusion loss \mathcal{L}_{d}^{t} w.r.t. W_{K}^{t} and W_{V}^{t} are projected onto the subspace \mathcal{S}^{t-1} and the orthogonal components, ΔW_{K}^{t} and ΔW_{V}^{t} , are used to update W_{K}^{t} and W_{V}^{t} respectively. (C) Diffusion sampling process: The diffusion model begins by denoising the initial noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Each denoising step incorporates class information from CLIP embeddings $\boldsymbol{\tau}(y)$ and the learned class prototype $\mathbf{c}(y)$. The class prototype is designed to capture the most distinctive features of a given class and assists the generator in synthesizing high-quality images.

Here we note that the class-prototype $\mathbf{c}(y)$ with $y \in \mathcal{Y}^t$ is trained during the task t and kept fixed in the future tasks.

4.3 Gradient Projection for CL Diffusion Model

Considering the CF phenomenon discussed above, it is desirable to further preserve and maintain the intermediate representations of data from task t with respect to the latent diffusion model $\epsilon_{\theta^{t'}}(\mathbf{x}_l, l, y, \mathbf{c}(y))$ in future task t', closely resembling those with respect to the diffusion model $\epsilon_{\theta^t}(\mathbf{x}_l, l, y, \mathbf{c}(y))$ in task t.

This desideratum aligns with the gradient projection memory [33] in the setting of Continual Learning (CL), where at each immediate layer, we need to compute a subspace which is the linear span of representations of the data from tasks so far, and then the model will be updated in the direction orthogonal to these subspaces. However, this approach is impractical in diffusion models due to (i) the large number of layers in the ϵ -network diffusion model and (ii) the requirement to compute subspaces with respect to all $\mathbf{x}_l, l = 1, \ldots, L$, given $\mathbf{x}_0 = \mathbf{x}$. In the following, we present how to achieve the above desideratum through a novel gradient projection technique tailored for the diffusion model in the context of image generation tasks.

For diffusion models [16,32], the cross-attention layers where the label embeddings $\tau_y \in \mathbb{R}^{d_\tau \times N}$ are injected to the model are crucially important. In this

Algorithm 1 Pseudocode for training our GPPDM.

Input: Task t, dataset \mathcal{D}^t with the label set \mathcal{Y}^t , classifier f_{ϕ} , diffusion model ϵ_{θ} , set of class-prototypes \mathcal{C} , pretrained CLIP model.

```
Output: Optimal \phi^*
 1: for t = 1, ..., T do
         if t \ge 2 then
 2:
             \mathcal{M}^{1:t-1} = DiffusionSampler(\mathcal{Y}^{1:t-1}, \mathcal{C}^{1:t-1}, \tilde{\epsilon}_{\theta})
 3:
             (\mathbf{x}, y) \sim \mathcal{D}^t \cup \mathcal{M}^{1:t-1}
 4:
 5:
 6:
             (\mathbf{x}, y) \sim \mathcal{D}^t
 7:
         end if
         Update \phi according to (7).
 8:
 9:
         Initialize \mathbf{c}(y) according to (16).
          Update \theta, \mathbf{c} \in \mathcal{C}^t according to (5) using the gradient projection technique.
10:
11: end for
```

work, we tailor the cross-attention layers for our gradient projection technique. Specifically, the computation of a cross-attention layer is as follows:

$$Q^{t} = W_{O}^{t} \varphi\left(\mathbf{x}\right), K^{t} = W_{K}^{t} \boldsymbol{\tau}_{y}, V^{t} = W_{V}^{t} \boldsymbol{\tau}_{y}, \tag{6}$$

where \mathbf{x} is a data example of task $t, \varphi(\mathbf{x}) \in \mathbb{R}^{d_z \times M}$ is the latent representation inputted to the cross-attention layer, $W_Q^t \in \mathbb{R}^{d \times d_z}$, $W_K^t \in \mathbb{R}^{d \times d_\tau}$, and $W_V^t \in \mathbb{R}^{d \times d_\tau}$. Here, we use the superscript (e.g., t) to represent the model parameters and representations at a task (e.g., t). Eventually, the cross-attention is computed as $Attention(Q, K, V) = softmax\left(\frac{Q^TK}{\sqrt{d}}\right)V^T$.

Since the diffusion model performs best with the data of task t, at a future task t' > t, we aim to ensure that $V^{t'} \approx V^t$, $K^{t'} \approx K^t$, and $Q^{t'} \approx Q^t$ to maintain performance on the data of task t when task t' arrives. Enforcing the constraint $Q^{t'} \approx Q^t$ is particularly challenging due to the large cardinality of $\varphi(\mathbf{x})$, where \mathbf{x} can be any element in the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_L$ from the forward process of the diffusion model with $(\mathbf{x}_0, y) \in \mathcal{D}^t$. This makes it impossible to construct a subspace for $\varphi(\mathbf{x})$. Therefore, to enforce this constraint, we simply update the parameter $W_Q^{t'}$ and other parameters to minimize the objective function in (5) with respect to the replay memory \mathcal{M}^t .

To ensure $K^{t'} \approx K^t$, and $Q^{t'} \approx Q^t$, we can develop a gradient projection technique for updating W_K^t and W_V^t . First, when task t arrives, we compute the subspace \mathcal{S}^t as the linear span of the column vectors of $\boldsymbol{\tau}(y)$, where $y \in \mathcal{Y}^t$. We then combine this subspace \mathcal{S}^t with the current set of subspaces \mathbf{S}^{t-1} to yield the set of subspaces up to task t: \mathbf{S}^t (i.e., $\mathbf{S}^t = \mathbf{S}^{t-1} \cup \mathcal{S}^t$). To gain W_K^t and W_V^t , we start from $W_K^t = W_K^{t-1}$ and $W_V^t = W_V^{t-1}$ and perform many update steps. Inspired by [33], at an update step, we denote $W_K^t = W_K^t - \eta \Delta W_K^t$ and $W_V^t = W_V^t - \eta \Delta W_V^t$ with the learning rate η , and require the row vectors in ΔW_K^t and ΔW_V^t to be orthogonal to \mathbf{S}^{t-1} to preserve the performance of old classes before task t. Additionally, these orthogonal constraints can be conveniently

Algorithm 2 DiffusionSampler

Input: Label set $\mathcal{Y}^{1:t-1}$, class-prototypes from previous tasks $\mathcal{C}^{1:t-1}$, classifier-free diffusion model $\tilde{\epsilon}_{\theta}$, diffusion steps L, number of generated samples R.

```
Output: \mathcal{M}^t
  1: \mathcal{M}^t = \emptyset
  2: for y \in \mathcal{Y}^{1:t-1} do
                 r = 0
  3:
                 while r \leq R do
   4:
                        Get \mathbf{c}(y) from C^{1:t-1}
   5:
   6:
                        \mathbf{x}_L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
   7:
                        for l = L, ..., 1 do
                              \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
  8:
                             \tilde{\boldsymbol{\mu}}_{\theta} = \frac{1}{\sqrt{\alpha_{l}}} \left( \mathbf{x}_{l} - \frac{1 - \alpha_{l}}{\sqrt{1 - \bar{\alpha}_{l}}} \tilde{\boldsymbol{\epsilon}}_{\theta} \left( \mathbf{x}_{l}, l, \mathbf{c}(y), \boldsymbol{\tau}(y) \right) \right)
\mathbf{x}_{l-1} = \tilde{\boldsymbol{\mu}}_{\theta} + \sigma_{l} \mathbf{z}
  9:
10:
11:
                        \mathcal{M}^t = \mathcal{M}^t \cup \{(\mathbf{x}_0, y)\}
12:
13:
                         r = r + 1
                  end while
14:
15: end for
```

implemented using the gradient projection technique. Specifically, let \mathcal{L}_d^t be the diffusion loss in (5) at task t. We compute $\Delta W_*^t = \nabla_{W_*^t} \mathcal{L}_d^t - \operatorname{Proj}_{\mathcal{S}^{t-1}} \left(\nabla_{W_*^t} \mathcal{L}_d^t \right)$ with $* \in \{K, V\}$ and use them to update W_K^t and W_V^t (see Figure 2).

4.4 Class-Prototype Conditional DM with Gradient Projection

Training Methodology. For each task t, we train the classifier f_{ϕ}^{t} by minimizing CE loss w.r.t. ϕ :

$$\min_{\phi} \mathcal{L}_{\mathcal{C}} := \mathbb{E}_{(\mathbf{x}, y)} \left[CE(f_{\phi}^{t}(\mathbf{x}), y) \right], \tag{7}$$

where $(\mathbf{x}, y) \sim \mathcal{D}^t \cup \mathcal{M}^{1:t-1}$. Subsequently, the diffusion model $\boldsymbol{\epsilon}_{\theta}^t (\mathbf{x}_k, k, \mathbf{c}(y), y)$ learns data from \mathcal{D}^t and replay memory $\mathcal{M}^{1:t-1}$ to generate samples for $\mathcal{M}^{1:t}$ that serves for the next task (as illustrated in Figure 2).

Class-prototype Initialization and Update. At task t, for each label $y_i^t \in \mathcal{Y}^t$, we initialize $\mathbf{c}(y_i^t)$ with sample belonging to this label that our model f_{ϕ}^t has the highest confidence on prediction:

$$\mathbf{c}(y_i^t) \coloneqq \operatorname{argmax}_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} p(y_i^t | \mathbf{x}^t) = \operatorname{argmin}_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} \left[CE(f_{\phi}^t(\mathbf{x}^t), y_i^t) \right], \tag{8}$$

where
$$\mathcal{X}^{y_i^t} \coloneqq \{\mathbf{x}^t | (\mathbf{x}^t, y^t) \in \mathcal{D}^t \text{ and } y^t = y_i^t\}.$$

These class prototypes are then updated along with the diffusion model by minimizing the objective function in (5). Note that we apply the gradient projection technique as discussed in Section 4.3 to update W_K^t and W_V^t of the cross-attention layers. The detailed algorithms are outlined in Algorithms 1 and 2.

5 Experiments

We assess the effectiveness of our proposed GPPDM in the commonly encountered Class Incremental (CI) scenario [37,38] and Class Incremental with Repeatition (CIR) scenario [8] in CL.

5.1 Setup

To ensure a balanced comparison, we adhere to the four experimental settings outlined in [12], applied to the CIFAR-100 and ImageNet [10] datasets for CI, and CORe50 [26] for CIR. Besides, one more dataset is used, CUB-200 [3], to have a comprehensive view of the performance of GPPDM in CI.

Baselines. We compare our approach with the baselines including Finetuning, SI [41], MAS [4], EWC [21], IMM [23], DGR [35], MeRGAN [7], PASS [43], and DDGR [12]. SI (Synaptic Intelligence), MAS (Memory Aware Synapses), EWC (Elastic Weight Consolidation), and IMM (Incremental Moment Matching) are methods that primarily focus on estimating the prior distribution of model parameters when assimilating new data. Generally, these approaches evaluate the significance of each parameter within a neural network, operating under the assumption of parameter independence for practicality. On the other hand, DGR (Deep Generative Replay) and MeRGAN leverage Generative Adversarial Networks (GANs) to create prior samples, facilitating data replay. Meanwhile, PASS represents a straightforward approach that does not rely on exemplars.

Datasets. For CI, we use two scenarios for each dataset CIFAR-100, ImageNet, and CUB-200. Half of the classes belong to the first task, the rest is divided equally among the remaining tasks. Additionally, CORe50 already has specific settings adapted to CIR. Specific details are as follows:

- CIFAR-100: initial task consists of 50 classes with two cases of incremental tasks: 5 incremental tasks (i.e., 10 classes per task) and 10 incremental tasks (i.e., 5 classes per task).
- ImageNet: initial task consists of 500 classes with two cases of incremental tasks: 5 incremental tasks (i.e., 100 classes per task) and 10 incremental tasks (50 classes per task).
- CUB-200: initial task consists of 100 classes with two cases of incremental tasks: 5 incremental tasks (i.e., 20 classes per task) and 10 incremental tasks (10 classes per task).
- CORe50: 79 batches as tasks are designed following [26] from 50 classes (10 coarse labels, 5 subclasses for each label) in which 10 classes for the initial task and 5 for each subsequent one.

Models. Following previous works [9, 25, 30], three model architectures are used for the classifier: AlexNet [22], 32-layer ResNet [15], and Vision Transformer ViT-B/16 [2]. Similarly, following [17], popular UNet architecture is used for the diffusion model.

Table 1: Average accuracy and forgetting (%) of methods across different architectures and numbers of classes in incremental tasks on CIFAR-100 and ImageNet. Underlined scores represent the second-best results.

Average Accuracy $A_T(\uparrow)$							Average Forgetting $F_T(\downarrow)$									
		CIFA	R-100		ImageNet					CIFAR-100 ImageNet						
Method	Alex	Net	Res	Net	Alex	Net	Res	Net	Alex	κNet	Res	Net	Alex	kNet	Res	Net
	NC=5	10	5	10	50	100	50	100	5	10	5	10	50	100	50	100
Finetuning	6.11	5.12	18.08	17.50	5.33	3.24	12.95	10.28	60.45	59.87	61.65	62.79	56.55	57.83	58.58	59.71
SI	16.96	13.57	26.45	23.15	19.38	14.38	28.88	24.38	48.58	50.18	52.27	56.65	41.18	45.94	41.93	44.56
EWC	15.29	9.71	25.49	18.82	15.22	13.03	23.51	22.03	50.38	54.27	52.83	60.47	45.65	47.01	46.98	46.96
MAS	20.13	18.94	29.94	28.28	16.35	14.51	31.25	25.51	45.75	45.37	49.38	51.31	44.85	45.70	39.55	44.34
IMM	11.26	9.87	21.02	19.79	13.68	11.13	23.19	19.73	54.60	54.57	58.12	59.89	46.65	49.31	47.70	49.62
$\overline{\mathrm{DGR}}$	42.49	38.16	52.96	48.94	43.94	38.81	53.32	47.56	24.08	26.52	26.36	31.14	17.31	22.52	17.96	21.84
MeRGAN	46.03	43.23	57.19	55.69	_	_	_	_	36.95	26.49	20.12	22.55	_	_	_	_
PASS	53.21	48.65	62.30	60.63	_	_	_	_	27.35	19.43	16.97	21.21	_	_	_	_
DDGR	<u>59.20</u>	52.22	63.40	60.04	<u>53.86</u>	<u>52.21</u>	<u>64.83</u>	61.26	23.00	16.86	15.34	<u>19.25</u>	6.98	7.82	5.65	7.73
GPPDM	76.35	62.76	77.79	68.07	58.49	52.71	72.88	65.98	8.46	9.63	6.47	6.21	5.94	4.40	4.30	4.75

Evaluations. For CI evaluations, two primary metrics are employed: Average Accuracy and Average Forgetting, with their specifics available in the supplementary materials. In the context of CIR, the test set consists of a comprehensive dataset encompassing all classes, evaluated through sequential tasks. Here, the performance of the current model on this dataset is measured in terms of Average Accuracy.

5.2 Experimental Results

Results in CI scenario. Table 1 displays the outperformance of our GPPDM compared to baseline models on the CIFAR-100 and ImageNet datasets, under different architectures and number of classes per subsequent task (NC). On CIFAR-100, GPPDM consistently beats the second best method by a large margin in terms of average accuracy and average forgetting, especially on a longer sequence of tasks. For example, with AlexNet, GPPDM improves the average accuracy of DDGR by around 17% and 10% when NC=5 and 10, respectively. On a more challenging ImageNet dataset, a similar pattern can be observed. In the case of AlexNet with NC=100, despite a slight improvement in regard to average accuracy, our method shows its ability to clearly reduce average forgetting from 7.82% to 4.40%. Moreover, Table 4 shows that our GPPDM also surpasses the main baseline DDGR on CUB-200.

Results in CIR scenario. A test dataset is maintained throughout the training process, therefore it is reasonable to consider maximum average accuracy instead of the forgetting term. As presented in [12], DDGR performs well in this scenario, the most obvious example is sample quality visualized, which is as good as real data. In experiments with AlexNet architecture, our results are the same as

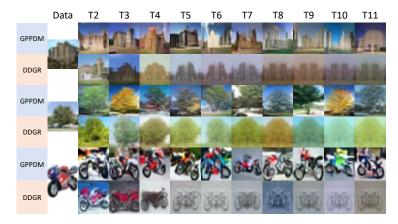


Fig. 3: Comparison of generated images from GPPDM and DDGR on CIFAR-100

DDGR. But with ResNet, we reach outperformance: 8% for A_T and 11% for $\max_{1 \le i \le T} (A_T)$, see Table 2.

Table 2: Comparison in Average accuracy (%) between our GPPDM and DDGR for CIR scenario with CORe50 dataset.

	A_T	(†)	$\max_{1 \leq i \leq T} (A_i) (\uparrow)$			
Method	AlexNet	ResNet	AlexNet	ResNet		
DDGR	31.00	40.66	31.00	42.00		
GPPDM	31.05	48.69	34.30	53.31		

5.3 Ablation Study

Forgetting comparison in DMs. Although DDGR demonstrates an effective ability to preserve sample quality across numerous tasks in CIR settings, it cannot maintain this performance in CI settings, where each class's real data appears only once across tasks. Figure 3 clearly shows that DDGR faces severe generation catastrophic forgetting, as seen in later tasks where its generated images become increasingly blurry and difficult to recognize. In contrast, our GPPDM effectively maintains key object features, demonstrating a more efficient approach to mitigating generational catastrophic forgetting. This is evident in the clarity and recognizability of the images generated by GPPDM, even in successive tasks. Figure 4 provides a more detailed look at the gaps across tasks in some metrics regarding image quality as well as the performance differences they bring. The gaps are computed as the differences in the results of DDGR and two variants of our GPPDM. Data belonging to task 1 and task 2 are selected to evaluate sample quality (FID between synthetic and real data, and IS of synthetic data). Dashed lines and triangles are used to describe for data task 1, dotted lines and circles are used for data task 2.

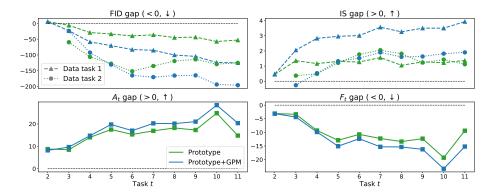


Fig. 4: Outperformance of our methods compare to DDGR during sequential tasks in settings of CIFAR-100 (NC = 5) with AlexNet is presented by gaps in Fréchet inception distance (FID), Inception Score (IS), Average Accuracy (A_t), and Average Forgetting (F_t).

Learning Class-Prototypes. Figure 5 presents the class-prototypes achieved after training. Notably, these prototypes tend to emphasize and focus more intensely on the objects present in the initial images. Such a concentration on object-specific information is clearly advantageous, guiding our GPPDM to preserve class-concept information effectively in subsequent tasks.

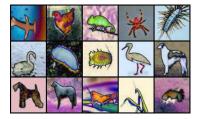


Fig. 5: Visualization of prototypes.

Contribution of each proposed compo-

nent. As shown in Table 3, when solely using either class-prototype guidance or tailored gradient projection technique, the forgetting of DDGR is significantly mitigated, resulting in higher average accuracy and lower average forgetting. Gradient projection seems to be better at maintaining stability of the model than prototype, but when the two are combined, the average forgetting is further reduced by around 1% across most settings.

Table 3: Contribution of each proposed component on CIFAR-100.

		$A_T($	1)	$F_T(\downarrow)$				
Method	Alex	Net	ResNet		AlexNet		Res	Net
	NC = 5	10	5	10	5	10	5	10
Default (DDGR)	59.20	52.22	63.40	60.04	23.00	16.86	15.34	19.25
Prototype	70.89	61.01	76.56	66.52	13.97	11.71	10.79	8.73
GPM	75.81	61.98	77.64	67.68	9.04	10.78	7.53	7.41
$\underline{ \text{Prototype} + \text{GPM} }$	76.35	62.76	77.79	68.07	8.46	9.63	6.47	6.21

The memory cost of GPPDM. In comparison with DDGR, GPPDM requires saving a prototype per class and, which is equivalent to saving only one image. Nonetheless, as previously shown, these prototypes can notably improve DDGR. Here, in Table 4, we present a fairer comparison by equipping DDGR a buffer with one real image per old class to train the diffusion model and the classifier. Generally, replaying one sample is helpful but can not enhance DDGR's performance as much as ours. This is because our learned prototypes can capture class concepts better than the saved images, and is also due to the effectiveness of our cross-attention gradient projection.

Table 4: A fair comparison to DDGR in which a real image for each class is chosen to put in the generative replay memory (i.e., DDGR + buffer).

			CIFAR-100						CUB-200			
Metric	Metric Method		AlexNet		ResNet		ViT		AlexNet		iΤ	
		5	10	5	10	5	10	10	20	10	20	
	DDGR	59.20	52.22	63.40	60.04	91.77	88.51	23.42	20.43	30.39	29.38	
$A_T(\uparrow)$	${\rm DDGR}{+}{\rm buffer}$	62.82	55.22	65.68	61.73	93.27	90.69	24.19	22.64	35.35	29.09	
	GPPDM	76.35	62.76	77.79	68.07	93.66	92.29	27.52	30.10	39.92	34.69	
	DDGR	23.00	16.86	15.34	19.25	6.17	7.36	17.20	9.19	17.46	10.00	
$F_T(\downarrow)$	${\rm DDGR}{+}{\rm buffer}$	21.29	16.85	19.03	9.94	4.80	5.49	8.11	6.00	10.91	9.73	
	GPPDM	8.46	9.63	6.47	6.21	4.49	4.29	1.94	4.52	8.36	7.33	

6 Conclusion

Catastrophic forgetting poses a critical challenge in the realm of continual learning. Generative replay, a method employing a generative model to recreate a replay memory for old tasks, aims to reinforce the classifier's understanding of past concepts. However, generative models themselves may encounter catastrophic forgetting, impeding the production of high-quality old data across tasks. Recent solutions, such as DDGR, integrating diffusion models, aim to reduce this issue in generation, yet they still face significant challenges. To address this, in this paper, we propose the Class-Prototype conditional Diffusion Model with Gradient Projection (GPPDM) to notably enhance the mitigation of generation catastrophic forgetting. The core idea involves the acquisition of a class-prototype that succinctly encapsulates the characteristics of each class. These class-prototypes then serve as guiding cues for diffusion models, acting as a mechanism to prompt their memory when generating images for earlier tasks. Through empirical experiments on real-world datasets, our GPPDM demonstrates its superiority, significantly outperforming current leading methods.

References

- Achille, A., Eccles, T., Matthey, L., Burgess, C.P., Watters, N., Lerchner, A., Higgins, I.: Life-long disentangled representation learning with cross-domain latent homologies. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. p. 9895–9905. NIPS'18 (2018)
- 2. Dosovitskiy et al., A.: An image is worth 16x16 words: Transformers for image recognition at scale
- 3. Wah et al., C.: The caltech-ucsd birds-200-2011 dataset
- 4. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European conference on computer vision (ECCV). pp. 139–154 (2018)
- Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-GEM. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=Hkf2_sC5FX
- Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: Continual learning with tiny episodic memories (2019)
- Chenshen, W., Herranz, L., Xialei, L., et al.: Memory replay gans: Learning to generate images from new categories without forgetting [c]. In: The 32nd International Conference on Neural Information Processing Systems, Montréal, Canada. pp. 5966–5976 (2018)
- Cossu, A., Graffieti, G., Pellegrini, L., Maltoni, D., Bacciu, D., Carta, A., Lomonaco,
 V.: Is class-incremental enough for continual learning? (2021)
- De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence 44(7), 3366– 3385 (2021)
- Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
- 11. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences $\mathbf{3}(4)$, 128-135 (1999)
- Gao, R., Liu, W.: DDGR: Continual learning with deep diffusion-based generative replay. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 10744–10763 (23–29 Jul 2023)
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014)
- 14. Goodfellow, I.J.: Nips 2016 tutorial: Generative adversarial networks. ArXiv abs/1701.00160 (2016)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
- Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 6840–6851. Curran Associates, Inc. (2020)

- 17. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
- 18. Ho, J., Salimans, T.: Classifier-free diffusion guidance. In: NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications (2021), https://openreview.net/forum?id=qw8AKxfYbI
- Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=SJ1Xmf-Rb
- Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014)
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences 114(13), 3521–3526 (2017)
- 22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 25 (2012)
- Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming catastrophic forgetting by incremental moment matching. Advances in neural information processing systems 30 (2017)
- Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence 40(12), 2935–2947 (2017)
- Liu, Y., Su, Y., Liu, A.A., Schiele, B., Sun, Q.: Mnemonics training: Multi-class incremental learning without forgetting. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 12245–12254 (2020)
- 26. Lomonaco, V., Maltoni, D.: Core50: a new dataset and benchmark for continuous object recognition (2017)
- 27. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7765–7773 (2018). https://doi.org/10.1109/CVPR.2018.00810
- 28. Mundt, M., Pliushch, I., Majumder, S., Hong, Y., Ramesh, V.: Unified probabilistic deep continual learning through generative replay and open set recognition. Journal of Imaging 8(4), 93 (2022)
- 29. Ramapuram, J., Gregorova, M., Kalousis, A.: Lifelong generative modeling. Neuro-computing **404**, 381–400 (2020)
- 30. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
- 31. Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science **7**(2), 123–146 (1995)
- 32. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022)
- Saha, G., Garg, I., Roy, K.: Gradient projection memory for continual learning. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=3A0j0RCNC2
- 34. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4548–4557. PMLR (10–15 Jul 2018)

- Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay.
 In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan,
 S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30.
 Curran Associates, Inc. (2017)
- 36. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning. pp. 2256–2265. PMLR (2015)
- 37. Van de Ven, G.M., Tolias, A.S.: Generative replay with feedback connections as a general strategy for continual learning. arXiv preprint arXiv:1809.10635 (2018)
- 38. Van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv preprint arXiv:1904.07734 (2019)
- Wu, C., Herranz, L., Liu, X., Wang, Y., Weijer, J.v.d., Raducanu, B.: Memory replay gans: Learning to generate images from new categories without forgetting. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. p. 5966–5976. NIPS'18 (2018)
- Xu, J., Zhu, Z.: Reinforced continual learning. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. p. 907–916 (2018)
- 41. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International conference on machine learning. pp. 3987–3995. PMLR (2017)
- 42. Zhai, M., Chen, L., Tung, F., He, J., Nawhal, M., Mori, G.: Lifelong gan: Continual learning for conditional image generation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 2759–2768 (2019)
- 43. Zhu, F., Zhang, X.Y., Wang, C., Yin, F., Liu, C.L.: Prototype augmentation and self-supervision for incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5871–5880 (2021)

Supplementary Material for "Class-Prototype Conditional Diffusion Model with Gradient Projection for Continual Learning"

No Author Given

No Institute Given

1 Training objective of Conditional Diffusion Probabilistic Model

To learn $p_{\theta}(\mathbf{x}_{l-1}|\mathbf{x}_{l}, y, \mathbf{c}(y))$, we apply the variational approach, which involves minimizing the cross-entropy divergence between $q(\mathbf{x}_{0}|y)$ and $p_{\theta}(\mathbf{x}_{0}|y, \mathbf{c}(y))$, represented by:

$$CE\left(q\left(\mathbf{x}_{0}|y\right), p_{\theta}\left(\mathbf{x}_{0}|y, \mathbf{c}(y)\right)\right)$$
 (1)

$$= -\mathbb{E}_{q(\mathbf{x}_0|y)} \left[\log p_{\theta} \left(\mathbf{x}_0 | y, \mathbf{c}(y) \right) \right] \tag{2}$$

$$= -\mathbb{E}_{q(\mathbf{x}_0|y)} \left[\int q(\mathbf{x}_{1:L} \mid \mathbf{x}_0, y) \log p_{\theta}(\mathbf{x}_0|y, \mathbf{c}(y)) d\mathbf{x}_{1:L} \right]$$
(3)

$$= -\mathbb{E}_{q(\mathbf{x}_0|y)} \left[\int q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_0, y\right) \log \frac{p_{\theta}\left(\mathbf{x}_{0:L} \mid y, \mathbf{c}(y)\right)}{p_{\theta}\left(\mathbf{x}_{1:L} \mid \mathbf{x}_0, y, \mathbf{c}(y)\right)} d\mathbf{x}_{1:L} \right]$$
(4)

$$= -\mathbb{E}_{q(\mathbf{x}_{0}|y)} \left[\int q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right) \log \frac{p_{\theta}\left(\mathbf{x}_{0:L} \mid y, \mathbf{c}(y)\right)}{q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right)} \frac{q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right)}{p_{\theta}\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y, \mathbf{c}(y)\right)} d\mathbf{x}_{1:L} \right]$$
(5)

$$= -\mathbb{E}_{q(\mathbf{x}_0 \mid y)} \Big[\int q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_0, y\right) \log \frac{p_{\theta}\left(\mathbf{x}_{0:L} \mid y, \mathbf{c}(y)\right)}{q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_0, y\right)} d\mathbf{x}_{1:L}$$

+
$$D_{KL}\left(q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right) \| p_{\theta}\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y, \mathbf{c}(y)\right)\right)\right]$$
 (6)

$$\leq -\mathbb{E}_{q(\mathbf{x}_{0}|y)} \left[\int q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right) \log \frac{p_{\theta}\left(\mathbf{x}_{0:L} \mid y, \mathbf{c}(y)\right)}{q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right)} d\mathbf{x}_{1:L} \right]$$
(7)

$$= -\mathbb{E}_{q(\mathbf{x}_{0:L}|y)} \left[\log \frac{p_{\theta}\left(\mathbf{x}_{0:L}|y,\mathbf{c}(y)\right)}{q\left(\mathbf{x}_{1:L}\mid\mathbf{x}_{0},y\right)} \right]$$
(8)

$$= \mathbb{E}_{q(\mathbf{x}_{0:L}|y)} \left[\log \frac{q(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y)}{p_{\theta}(\mathbf{x}_{0:L}|y, \mathbf{c}(y))} \right]. \tag{9}$$

The optimization problem then becomes:

$$\min_{\theta} \mathcal{L} := \mathbb{E}_{q(\mathbf{x}_{0:L}|y)} \left[\log \frac{q(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y)}{p_{\theta}(\mathbf{x}_{0:L}|y, \mathbf{c}(y))} \right]. \tag{10}$$

We further derive the objective function of interest as:

$$\log \frac{q\left(\mathbf{x}_{1:L} \mid \mathbf{x}_{0}, y\right)}{p_{\theta}\left(\mathbf{x}_{0:L} \mid y, \mathbf{c}(y)\right)} = \log \frac{q\left(\mathbf{x}_{L} \mid \mathbf{x}_{0}, y\right)}{p\left(\mathbf{x}_{L}, y, \mathbf{c}(y)\right)} \times \prod_{l=2}^{L} \frac{q\left(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, \mathbf{x}_{0}, y\right)}{p_{\theta}\left(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, y, \mathbf{c}(y)\right)}$$

$$\times \frac{1}{p_{\theta}\left(\mathbf{x}_{0} \mid \mathbf{x}_{1}, y, \mathbf{c}(y)\right)}$$

$$= \log \frac{q\left(\mathbf{x}_{L} \mid \mathbf{x}_{0}, y\right)}{p\left(\mathbf{x}_{L}, y, \mathbf{c}(y)\right)} + \sum_{l=2}^{L} \log \frac{q\left(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, \mathbf{x}_{0}, y\right)}{p_{\theta}\left(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, y, \mathbf{c}(y)\right)}$$

$$- \log p_{\theta}\left(\mathbf{x}_{0} \mid \mathbf{x}_{1}, y, \mathbf{c}(y)\right),$$

$$(12)$$

leading us to the following conclusion:

$$\mathcal{L} = \mathbb{E}_{q} \left[\log \frac{q(\mathbf{x}_{L} \mid \mathbf{x}_{0}, y)}{p(\mathbf{x}_{L} \mid y, \mathbf{c}(y))} + \sum_{l=2}^{L} \log \frac{q(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, \mathbf{x}_{0}, y)}{p_{\theta}(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, y, \mathbf{c}(y))} - \log p_{\theta}(\mathbf{x}_{0} \mid \mathbf{x}_{1}, y, \mathbf{c}(y)) \right]$$

$$= \mathbb{E}_{q} \left[\underbrace{D_{KL} \left(q(\mathbf{x}_{L} \mid \mathbf{x}_{0}, y) \parallel p(\mathbf{x}_{L} \mid y, \mathbf{c}(y)) \right)}_{\mathcal{L}_{L}} + \sum_{l>1} \underbrace{D_{KL} \left(q(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, \mathbf{x}_{0}, y) \parallel p_{\theta}(\mathbf{x}_{l-1} \mid \mathbf{x}_{l}, y, \mathbf{c}(y)) \right)}_{\mathcal{L}_{l-1}} - \underbrace{\log p_{\theta}(\mathbf{x}_{0} \mid \mathbf{x}_{1}, y, \mathbf{c}(y))}_{\mathcal{L}_{0}} \right]$$

$$(14)$$

2 Implementation Specification and Additional Experimental Results

2.1 Additional experiments

Analysis of Class-Prototype Initialization. In the CIFAR-100 (NC=5) experiment using the AlexNet model, we examine various initializations of class-prototypes. In addition to the initialization used in the main paper, given $y_i^t \in \mathcal{Y}^t$, three alternatives are also investigated as follows:

- Most confident initialization (as presented in Section 4.4)

$$\mathbf{c}(y_i^t) \coloneqq \arg\max_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} p(y_i^t | \mathbf{x}^t) = \arg\min_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} \left[CE(f_{\phi}^t(\mathbf{x}^t), y_i^t) \right]. \tag{15}$$

Least confident initialization

$$\mathbf{c}(y_i^t) := \arg\min_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} p(y_i^t | \mathbf{x}^t) = \arg\max_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} \left[CE(f_{\phi}^t(\mathbf{x}^t), y_i^t) \right]. \tag{16}$$

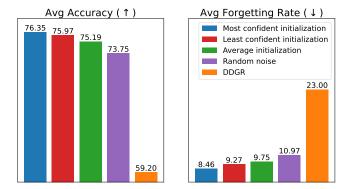


Fig. 1: Comparison of final average accuracy $A_T(\uparrow)$ and final average forgetting $F_T(\downarrow)$ in the setting of CIFAR-100 (NC=5), AlexNet with different class-prototype initialization strategies relative to DDGR.

- Average initialization in the same class

$$\mathbf{c}(y_i^t) \coloneqq \frac{1}{|\mathcal{X}^{y_i^t}|} \sum_{\mathbf{x}^t \in \mathcal{X}^{y_i^t}} \mathbf{x}^t. \tag{17}$$

Random initialization

$$\mathbf{c}(y) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$
 (18)

Figure 1 illustrates the implicit relationship between the generator and the classifier in how to initialize the class prototypes, revealing that the most confident initialization method yields the best performance. Averaging samples in pixel space appears to be less effective, leading to worse outcomes compared to even the least confident initialization approach. Additionally, while the random initialization method results in slightly poorer performance, it significantly outperforms the baseline (DDGR).

Experiment without a large first task. Table 1 presents the Average Accuracy after each task on CIFAR-100 where all tasks have the same number of classes (20) with AlexNet as classifier architecture. Our method yields better results compared to DDGR and PASS across all tasks. However the gap of the final Average Accuracy (6.31) is not as much as in other cases which have a larger first task ($NC=5:17.15,\ NC=10:10.54$). There are two reasons: fewer tasks are considered, and the capacity of the first task is smaller. Hence, it becomes simpler for the classifier to remember old tasks.

The effect of diffusion steps. Increasing the number of diffusion timesteps (K) in training (from 1000 to 2000, 3000, 4000) while keeping the number of training steps and the number of inference timesteps the same (100) makes the generator converge more slowly and makes it more difficult to form an image, thereby making the results worse. Figure 2 shows final Average Accuracy and final Average

Table 1: Average Accuracy $A_i(\uparrow)$ across tasks on CIFAR-100 5 tasks, 20 classes / task, AlexNet model.

	Average Accuracy								
Method	Task 1 (A_1)	Task 2 (A_2)	Task 3 (A_3)	Task 4 (A_4)	Task 5 (A_5)				
PASS	70.51	52.17	50.41	47.39	44.32				
DDGR	70.65	54.33	53.35	51.16	49.03				
GPPDM	70.90	66.30	61.10	56.79	55.34				



Fig. 2: Results on CIFAR-100, AlexNet model, while changing the number of diffusion timesteps.

Forgetting of AlexNet model for CIFAR-100 dataset in two cases NC = 5 and NC = 10.

2.2 Architecture / Hyperparameters

Data preparation and preprocessing. In our implementation, which is grounded on the DDGR framework as delineated in [12], a standard preprocessing protocol is employed across three distinct datasets: CIFAR-100, ImageNet, CUB-200, and CORe50 [26]. Central to this protocol is the resizing of image samples to a uniform dimension of 64×64 . However, some exceptions are made: Vision Transformers (ViT-B/16) only get inputs with size of 224×224 ; the rest case is for ImageNet and CUB-200, wherein the 64×64 resized samples undergo an additional resizing step, being scaled up to 256×256 before being fed to the classifier (AlexNet and ResNet32). This resizing process, applied consistently across all datasets, employs a bilinear technique, ensuring uniformity in sample processing.

Architecture. We utilize UNet architecture in our implementation. Specifically, we acquire a noisy sample denoted as \mathbf{x}_l at time step l, which mirrors the dimensions

of real data, measuring $3 \times 64 \times 64$. Simultaneously, our class-prototype, $\mathbf{c}(y)$, shares this spatial dimensionality. To facilitate seamless integration into our subsequent processes, we concatenate \mathbf{x}_l and $\mathbf{c}(y)$ in the channel dimension, resulting in a composite tensor measuring $6 \times 64 \times 64$. This composite tensor is then seamlessly integrated into the UNet model for further processing and analysis. Label text embedding, output of CLIP model, having dimensions of 1×768 , is used as key and value of Cross Attention operation in some specific blocks. Figure 3 presents the pipeline of UNet model.

Label text embeddings. In this section, we detail the method of using text embedding for labels using CLIP.

 CI scenario: Datasets used in this scenario are CIFAR-100, CUB-200, and ImageNet, which have a relatively clear distinction between the classes.
 Creating label text embedding is as simple as follows:

$$y \mapsto \tau(y) = \mathbf{CLIP}(\text{``a photo of a } [\mathbf{classname}(y)]\text{''})$$
 (19)

 CIR scenario: The CORe50 dataset has a little difference in the meaning of the classes compared to the above two sets. This dataset has a hierarchical labels set, in which there are 10 coarse labels, each includes 5 fine labels. Therefore, label text embedding follows the following formula:

$$y \mapsto \tau(y) = \mathbf{CLIP}(\text{``a photo of a } [\mathbf{classname}(y)]]$$

in category [classfineindex(y)]") (20)

where $classname(\cdot)$ and $classfineindex(\cdot)$ are presented in the Table 2

Hyperparameters. Details of hyperparameters for each experiment are shown in Table 3.

 ${\bf Table~2:~Details~CORe 50~dataset~labels.}$

Coarse label	Label index (y)	Fine label	Class name	Class fi
	0	plug_adapter1	plug adapter	
plug adapter	1	$plug_adapter2$	plug adapter	
1 -01	2	plug_adapter3	plug adapter	
	3	$plug_adapter4$	plug adapter	
	4	$plug_adapter5$	plug adapter	
	5	${\bf mobile_phone1}$	mobile phone	
mobile phone	6	$mobile_phone2$	mobile phone	
moone_prione	7	$mobile_phone3$	mobile phone	
	8	$mobile_phone 4$	mobile phone	
	9	$mobile_phone 5$	mobile phone	
	10	scissor1	scissor	
scissor	11	scissor2	scissor	
3013301	12	scissor3	scissor	
	13	scissor4	scissor	
	14	scissor5	scissor	
	15	light_bulb1	light bulb	
1: mb. 4 1 11	16	light_bulb2	light bulb	
light_bulb	17	light_bulb3	light bulb	
	18	light_bulb4	light bulb	
	19	light bulb5	light bulb	
	20	can1	can	
	21	can2	can	
can	22	can3	can	
	23	can4	can	
	24	can5	can	
	25	glass1	glass	
	26	glass2	glass	
glass	27	glass3	glass	
	28	glass4	glass	
	29	glass5	glass	
	30	ball1	ball	
	31	ball2	ball	
ball	32	ball3	ball	
	33	ball4	ball	
	34	ball5	ball	
	35	marker1	marker	
		marker1 marker2		
marker	36	marker2 marker3	marker	
	37			
	38	marker4	marker	
	39	marker5	marker	
	40	cup1	cup	
cup	41	cup2	cup	
	42	cup3	cup	
	43	cup4	cup	
	44	cup5	cup	
	45	remote_control1		
$remote_control$		remote_control2		
	47	remote_control3		
	48		remote control	

 Table 3: Details of hyperparameters.

	Dataset		CI						
			CIFA	.R-100	CUE	3-200	Imag	CORe50	
	Experiment		NC = 5	NC = 10	$NC = 10 \ NC = 20$		NC = 50 NC = 10		
	No. task	s	11	6	11	6	11	6	79
Statistic	No. classes/task	Initital task	50	50	100	100	500	500	10
Deadlistic	rior classes, tasir	Rest tasks	5	10	10	20	50	100	5
	No. training samples/task	Initital task	25000	25000	3000	3000	650000	650000	3000
		Rest tasks	2500	5000	300	600	65000	130000	1500
	No. training epo	ochs/task	100	100	150	150	15	15	100
	Batch size	AlexNet, ResNet32	256	256	64	64	64	64	256
Classifier	Batter Sille	ViT-B/16	64	64	64	64	-	-	-
	Optimiz	er	SGD	SGD	SGD	SGD	SGD	SGD	SGD
	Learning 1	0.001	0.001	0.0001	0.0001	0.0001	0.0001	0.0001	
	Weight de	cay	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
	No. training steps/task	Initial task	15000	15000	15000	15000	150000	150000	15000
	rtor training stops, task	Rest tasks	15000	15000	15000	15000	30000	60000	15000
	Batch si	ze	64	64	64	64	64	64	64
	Mixed prec	fp16	fp16	fp16	fp16	fp16	fp16	fp16	
	Drop label rate (Classif	0.2	0.2	0.2	0.2	0.2	0.2	0.2	
Diffusion	Optimiz	er	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
	Learning	ate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Weight de	cay	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	Gradient cli	pping	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Training sch	eduler	DDPM	DDPM	DDPM	DDPM	DDPM	DDPM	DDPM
	No. training ti	mesteps	1000	1000	1000	1000	1000	1000	1000
	Inference sch	eduler	DDIM	DDIM	DDIM	DDIM	DDIM	DDIM	DDIM
	No. inference t	imesteps	100	100	100	100	100	100	100
	Classifier-free guidance sa	impling weight (w)	4.0	4.0	4.0	4.0	4.0	4.0	4.0
	No. generated sar	nples/class	20	20	20	20	20	20	20
Class-protype	Learning 1	ate	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Jaco protype	Weight de	cay	0.01	0.01	0.01	0.01	0.01	0.01	0.01

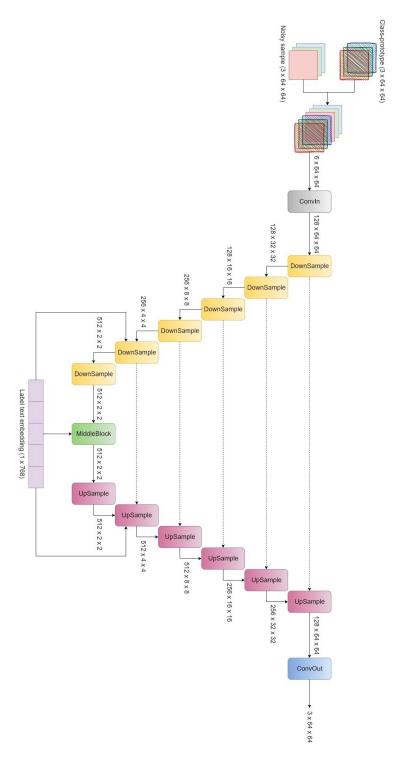


Fig. 3: UNet diffusion model.