UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

REINFORCEMENT LEARNING ENHANCED GENERATIVE REPLAY FOR CONTINUAL LEARNING IN DIFFUSION MODELS: A COMPREHENSIVE STUDY OF EXISTING AND NOVEL METHODOLOGIES

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE DATOS,

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

ROBERTO JOAQUÍN RODOLFO BARCELÓ SOLTMANN

PROFESOR GUÍA:
FELIPE TOBAR HENRÍQUEZ

PROFESOR CO-GUÍA:
JORGE SILVA SANCHEZ

COMISIÓN:
JOAQUÍN FONTBONA TORRES

Este trabajo ha sido financiado por Fondecyt Regular N°1210606

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE DATOS
Y MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: ROBERTO JOAQUÍN RODOLFO BARCELÓ SOLTMANN
FECHA: 2024
PROF. GUÍA: FELIPE TOBAR HENRIQUEZ
PROF. CO-GUÍA: JORGE SILVA S.

REPETICIÓN GENERATIVA MEJORADA CON APRENDIZAJE POR REFUERZO PARA EL APRENDIZAJE CONTINUO EN MODELOS DE DIFUSIÓN: UN ESTUDIO INTEGRAL DE METODOLOGÍAS EXISTENTES Y NUEVAS

Los modelos generativos de difusión han demostrado capacidades notables en la generación de datos artificiales de alta calidad en diversas áreas, tales como imágenes, proteínas y materiales. Sin embargo, estos modelos enfrentan desafíos significativos en escenarios de Aprendizaje Continuo, donde deben aprender de forma continuada distribuciones de datos en evolución mientras preservan la información aprendida en etapas anteriores.

Esta tesis investiga la mejora de los modelos de difusión en Aprendizaje Continuo a través de estrategias de repetición generativa. Los objetivos principales incluyen una evaluación exhaustiva de las estrategias existentes de Aprendizaje Continuo y el desarrollo de metodologías novedosas aprovechando el Aprendizaje Reforzado, específicamente utilizando *Denoising Diffusion Policy Optimization* (DDPO). Aprovechando esta metodología, proponemos dos marcos de entrenamiento: un enfoque directo y un marco de profesor-estudiante orientado a mejorar la estabilidad del entrenamiento en Aprendizaje Continuo de estos modelos de difusión.

A través de variados experimentos, replicamos con éxito metodologías existentes y establecimos puntos de referencia claros con los cuales comparar las metodologías. Nuestras nuevas propuestas, en particular la repetición generativa mejorada con DDPO, demostraron mejoras significativas en la calidad de las muestras sintéticas generadas y la retención de tareas previamente aprendidas en Aprendizaje Continuo. El marco profesor-estudiante mejoró aún más el rendimiento al separar efectivamente las tareas de entrenamiento, enfocándose en aprender mejores representaciones de datos en el modelo profesor y evitando la sobreparametrización en el modelo estudiante.

Nuestros hallazgos indican que el Aprendizaje Reforzado puede mejorar significativamente las capacidades de Aprendizaje Continuo de los modelos de difusión. Sin embargo, el éxito de estos métodos depende en gran medida de la calidad de las funciones de recompensas utilizadas. En el futuro se debería explorar recompensa más sofisticadas y estrategias alternativas de Aprendizaje Reforzado para mejorar aún más el rendimiento y la diversidad de los modelos generativos.

Esta investigación contribuye al avance de los modelos generativos, extendiendo su aplicabilidad y eficacia en entornos que requieren aprendizaje continuo.

ABSTRACT OF THE REPORT TO QUALIFY TO THE DEGREE OF MASTER OF DATA SCIENCE
AND TO THE DEGREE OF CIVIL ELECTRICAL ENGINEERING
BY: ROBERTO JOAQUÍN RODOLFO BARCELÓ SOLTMANN
DATE: 2024
PROF. GUÍA: FELIPE TOBAR HENRIQUEZ
PROF. CO-GUÍA: JORGE SILVA S.

REINFORCEMENT LEARNING ENHANCED GENERATIVE REPLAY FOR CONTINUAL LEARNING IN DIFFUSION MODELS: A COMPREHENSIVE STUDY OF EXISTING AND NOVEL METHODOLOGIES

Generative diffusion models have shown remarkable capabilities in synthesizing high quality data across various domains, such as images, proteins, and materials. However, these models face significant challenges in continual learning scenarios, where they must continuously learn from evolving data distributions while preserving previously learned information.

This thesis investigates the enhancement of generative diffusion models through generative replay in continual learning. The main objectives include a comprehensive evaluation of existing continual learning strategies and the development of novel methodologies leveraging Reinforcement Learning, specifically Denoising Diffusion Policy Optimization (DDPO). Leveraging this methodology, we propose two frameworks for training: a direct approach and a teacher-student framework aimed at improving the stability of training continually these generative diffusion models.

Through extensive experiments, we successfully replicated existing methodologies and established clear benchmarks. Our novel approaches, particularly the DDPO enhanced generative replay, demonstrated significant improvements in sample quality and retention of previously learned tasks. The teacher-student framework further enhanced performance by effectively separating training tasks, focusing on learning better data representations in the teacher model and avoiding overparameterization in the student model.

Our findings indicate that Reinforcement Learning can significantly improve the continual learning capabilities of diffusion models. However, the success of these methods heavily depends on the quality of the reward model. Future work should explore more sophisticated reward functions and alternative reinforcement learning strategies to further enhance the performance and diversity of generative models.

This research contributes to the advancement of generative models, extending their applicability and efficacy in settings that require continual learning.

Acknowledgements

Agradecimiento a mis profesoras y profesores por su valiosa formación y dedicación. A mi familia y amigos que me apoyaron en este proceso de estudio y esfuerzo. En especial a mis padres, que gracias a ellos pude disfrutar de la oportunidad de estudiar y profundizar en mis áreas de interés.

Table of Content

Index of Tables

Table of Figures

Chapter 1

Introduction

1.1. Research Problem

Generative models, particularly diffusion models [1] [2], have demonstrated remarkable capabilities in synthesizing high-quality data across various domains, such as multimedia content (images, video, audio, etc), novel materials, molecules and text. A significant challenge for diffusion models arises in dynamic environments, where they must continuously learn from evolving data distributions [3] [4]. These models often struggle to adapt to new data while preserving previously learned information, a dilemma known as catastrophic forgetting. This issue is particularly critical in scenarios that demand continual or lifelong learning, where the ability to integrate new information seamlessly without undermining the previously acquired knowledge base is essential.

Currently, the problem of catastrophic forgetting in diffusion models is tackled through various methods that include regularization techniques, finetuning, and replay methods during training [5] [6] [7] [8] [9] [10]. Among replay strategies, generative replay, an approach where the model regenerates its training data, presents an interesting alternative to buffer-based replay systems. However, existing implementations of generative replay often fall short in performance, struggling to effectively balance the retention of old knowledge with the acquisition of new information. [3]

This thesis investigates the challenge of enhancing the performance of generative diffusion models, through generative replay in continual learning. The research primarily focuses on two aspects: firstly, a thorough examination of the effectiveness of current continual learning strategies within the framework of generative diffusion models, and secondly, the proposal and evaluation of novel methodologies that leverage Reinforcement Learning to improve the generative process. In particular, this study leverages the Denoising Diffusion Policy Optimization (DDPO) algorithm, hypothesized to significantly enhance the quality and stability of generated samples, especially in scenarios where leveraging previous data is crucial for continual learning. The integration of this algorithm enables the generative model to self-generate its training data, thus retaining proficiency in previously learned tasks while acquiring new ones.

By addressing these challenges, this research aims to contribute to the advancement of generative models, extending their applicability and efficacy in settings that require continual learning.

1.2. Hypothesis

This thesis proposes two innovative methodologies, grounded in the concept of generative replay, to enhance the performance of generative diffusion models in continual learning environments using the Denoising Diffusion Policy Optimization (DDPO) algorithm.

The first hypothesis predicts that applying DDPO directly to a generative diffusion model, with custom-designed reward functions, will significantly improve the quality of the model's self-generated data. This enhancement is expected to exceed the capabilities of traditional generative replay by producing samples that more closely align with the original data, thereby reducing the need for retraining on a buffer of the original data.

The second hypothesis suggests that a "teacher-student" model framework could be beneficial. In this setup, a "teacher" model, a version of the original model optimized with DDPO, generates refined training samples. These samples are then used to train the "student" model, which does not undergo direct optimization with DDPO. This strategy prevents over-optimization of the student model by utilizing higher-quality, task-specific data from the teacher model.

Both approaches are hypothesized to expand the capabilities of generative diffusion models in continual learning settings. They are expected to not only reinforce the models existing knowledge but also to adapt to evolving or diverse tasks, thereby enhancing the overall scope and effectiveness of generative replay in these contexts.

1.3. Objectives

1.3.1. General Objectives

The general objective of this research is to enhance the capabilities of generative diffusion models in continual learning environments. This involves developing and assessing methodologies that enable these models to effectively adapt to evolving data distributions while retaining previously learned information. The aim is to address the challenges of catastrophic forgetting and to ensure robust, high-quality data generation in dynamic learning scenarios.

1.3.2. Specific Objectives

1. To conduct a comprehensive implementation and evaluation of existing continual learning strategies within the context of generative diffusion models, identifying their strengths and limitations in dynamic data environments.

2. To develop and implement a novel approach using Denoising Diffusion Policy Optimization (DDPO) to directly improve the sample generation quality of generative diffusion models, focusing on the adaptation and retention of learned information.

3. To create and evaluate a "teacher-student" model framework, where the "teacher" model, optimized with DDPO, generates training samples for the "student" model. This ap-

proach aims to leverage high-quality, task-specific data generation without directly involving the student model in the complexities of RL optimization.

4. To assess the effectiveness of the proposed methodologies in mitigating catastrophic forgetting and enhancing the model's adaptability to new tasks and data, while comparing their performance against traditional generative replay techniques.

5. To explore and define optimal reward function configurations in the DDPO algorithm that align with specific continual learning objectives in generative diffusion models.

6. To conduct empirical experiments in various continual learning scenarios to evaluate the stability, adaptability, and output quality of the enhanced generative diffusion models, using appropriate metrics and benchmarks.

Chapter 2

Background and Theory

2.1. Generative Models

Generative models, as discussed in [11], encompass a broad class of machine learning algorithms designed with the principal goal of learning to represent and generate data that is similar to the input data they have been trained on. These models are adept at understanding and capturing the underlying probability distributions of data domains to produce new instances that could plausibly come from the same distributions.

The mechanics of generative models revolve around the concept of modeling the joint distribution of data samples, which can be either unconditional or conditioned on additional variables. The distinction between deep generative models (DGMs) and probabilistic graphical models (PGMs) is prominent, with DGMs relying on neural networks to transform latent representations into complex data, and PGMs often using simpler, sometimes linear relationships between variables. Both types, along with their hybrids, contribute to the diversity and flexibility of generative models.

The application of generative models is multi-faceted, ranging from synthetic data generation and augmentation to complex decision-making frameworks. The utility of these models in various domains showcases their importance in advancing both theoretical and practical aspects of machine learning.

We will explore a specific class of generative models known as diffusion models [1]. These models have received attention for their proficiency in generating high-quality samples and offer a novel perspective on the generative process.

2.2. Difussion Models

Diffusion models have recently garnered significant interest for their capability to generate diverse, high-quality samples and their relatively straightforward training methodology. This enables scaling up to very large model sizes.

The core concept of diffusion models leverages the ease of transitioning from structured data to noise, a process that is more challenging to perform in reverse. The forward process, also known as the diffusion process, incrementally transforms observed data x_0 into a noisier version x_T via T iterative applications of a stochastic encoder $q(x_t|x_{t-1})$. After sufficient iterations, the data converges to a noise distribution, typically Gaussian $\mathcal{N}(0, I)$.

Conversely, a learned reverse process or decoder $p_\theta(x_{t-1}|x_t)$ aims to recover the original data from this noisy state. This bidirectional process outlines the training and generation phases of diffusion models, encapsulated in a high-level framework illustrated in the Figure 2.1.

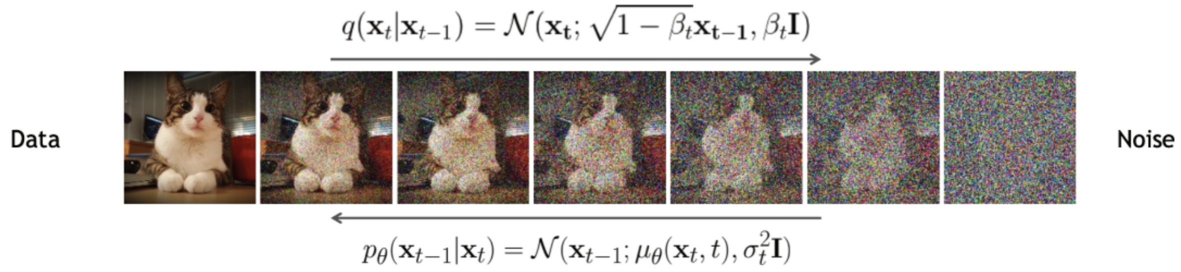

$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$
Data
Noise
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

Figure 2.1: The denoising diffusion model with the forward process $q(x_t|x_{t-1})$ adding Gaussian noise, and the learned reverse process $p_\theta(x_{t-1}|x_t)$ reconstructing the data. Adapted from [11].

2.2.1. Denoising Difussion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPMs) [1, 11] are a class of generative models that learn to generate data through the reversal of a diffusion process. The diffusion process incrementally adds noise to the data, transforming it into a completely noisy state. Conversely, the reverse process aims to convert the noise back into meaningful data.

Inspired by principles from non-equilibrium thermodynamics, DDPMs employ a series of latent variables that correspond to various levels of noise. The model is tasked with the denoising of data at each stage of the diffusion process, effectively learning to negate the added noise.

2.2.1.1. Encoder (Forward Diffusion)

The encoding process for forward diffusion is characterized by a simple linear Gaussian model:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t \mid \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}\right). \tag{2.1}$$

Here, the parameters β_t are selected from the interval $(0, 1)$ and are specified by a noise schedule (discussed further). The joint distribution over all latent stages, with respect to the input, is formulated as:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}). \tag{2.2}$$

As this represents a linear Gaussian Markov chain, it is possible to compute its marginals in closed form. In particular:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t \mid \sqrt{\alpha_t}x_0, (1-\alpha_t)\mathbf{I}\right). \tag{2.3}$$

where α_t is defined as:

$$\alpha_t \triangleq 1 - \beta_t, \quad \alpha_{1:t} = \prod_{s=1}^{t} \alpha_s. \tag{2.4}$$

The noise schedule is chosen so that $\alpha_T \approx 0$, resulting in $q(x_T \mid x_0) \approx \mathcal{N}(0, \mathbf{I})$.

The distribution $q(x_t \mid x_0)$, known as the diffusion kernel, when applied to the input data distribution followed by the calculation of unconditional marginals, is equivalent to Gaussian convolution:

$$q(x_t) = \int q_0(x_0) q(x_t \mid x_0) dx_0. \tag{2.5}$$

As t increases, the marginals become increasingly simpler, a process visualized in Figure 25.2. Within the realm of image processing, this leads to an initial removal of high-frequency content, such as detail and texture, followed by the diminishing of low-frequency content, which includes the more significant "semantic" information, as demonstrated in Figure 25.1.

2.2.1.2. Decoder (Reverse Diffusion)

In the reverse diffusion phase of the decoder, the objective is to invert the forward diffusion steps. If the original input x_0 is known, we can reverse a single forward step with:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(x_{t-1} \mid \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \mathbf{I}\right). \tag{2.6}$$

The mean $\tilde{\mu}_t$ is a function of x_t and x_0, given by:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha_{t-1}\beta_t}}{1 - \alpha_t} x_0 + \frac{\sqrt{\alpha_t (1 - \alpha_{t-1})}}{1 - \alpha_t} x_t. \tag{2.7}$$

The adjusted noise scale $\tilde{\beta}_t$ is expressed as:

$$\tilde{\beta}_t = \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \beta_t. \tag{2.8}$$

To generate new data points where x_0 is not known, the generative model, trained to approximate the distribution over x_0, is described by:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1} \mid \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right). \tag{2.9}$$

Typically, $\Sigma_\theta(x_t, t)$ is chosen as $\sigma_t^2 \mathbf{I}$. The choices $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \tilde{\beta}_t$ are natural, corresponding to the bounds on the reverse process's entropy.

The generative model's full joint distribution is given by:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t). \tag{2.10}$$

with the initial distribution $p(x_T) = \mathcal{N}(0, \mathbf{I})$. Sampling from this model can be executed as per the steps detailed in Algorithm 25.2.

2.2.1.3. Model Fitting

The fitting of diffusion models involves maximizing the evidence lower bound (ELBO) [1]. This is accomplished for each data example x_0 by computing the log likelihood as an expectation over the distribution $q(x_1 : T|x_0)$ and the model's distribution $p_\theta(x_0 : T)$, leading to the ELBO as:

$$\log p_\theta(x_0) \geq \mathbb{E}_q \left[\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] = \mathcal{L}(x_0). \tag{2.11}$$

We exploit the Markov property of DDPMs and Bayes rule to express the ELBO in terms of the model's parameters and the variational posterior [1]. The negative ELBO, which serves as a variational upper bound, can then be decomposed into time-dependent Kullback-Leibler (KL) divergences between the variational posterior and the model's predictive distributions. These terms can be computed analytically when all involved distributions are Gaussian.

Optimizing the Model—In practice, the optimization of the diffusion model aims at predicting the noise in the data. The model is trained to estimate the noise, which is then used to calculate the mean of the denoised version of x_{t-1} from its noisy input x_t. This leads to a loss function that corresponds to the maximum likelihood estimation and can be simplified further by setting certain time-dependent weights to one, yielding a simpler loss that often results in better sample quality:

$$L_{simple} = \mathbb{E}_{x_0 \sim q_0(x_0), \epsilon \sim \mathcal{N}(0,I), t \sim \mathrm{Unif}(1,T)} \left[\| \epsilon - \epsilon_\theta(\sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon, t) \|^2 \right]. \tag{2.12}$$

This summarizes the essential process of fitting diffusion models by maximizing the ELBO, with the goal of improving the generated samples quality.

2.3. Reinforcement Learning for Difussion Models

2.3.1. Markov Decision Processes and Reinforcement Learning

2.3.1.1. General Definition

Reinforcement Learning (RL) is defined through its fundamental problem: learning from interactions to achieve specific goals. This problem, which spans a wide range of applications, characterizes any method capable of solving it as an RL method. The objective here is to broadly outline the RL problem, emphasizing its potential applications and introducing its mathematical structure.

2.3.1.2. The Agent-Environment Interface

As defined by Sutton and Barto [12], the reinforcement learning problem centers around the interaction between an agent and its environment. The agent, as the decision-maker and learner, selects actions based on the current state of the environment. Each action taken by the agent leads the environment to present a new state and provide rewards. These rewards are numerical values that the agent seeks to maximize, representing the core objective of any reinforcement learning task. A complete specification of the environment, therefore, defines the learning task and the challenges the agent must overcome to achieve its goal.

This continuous loop of action, state update, and reward is fundamental to the learning process in reinforcement learning and is depicted in the conceptual model in Figure2.2 that is described in Sutton and Barto's text. The model demonstrates how the agent-environment interaction underpins all decision-making and learning activities in reinforcement learning systems.

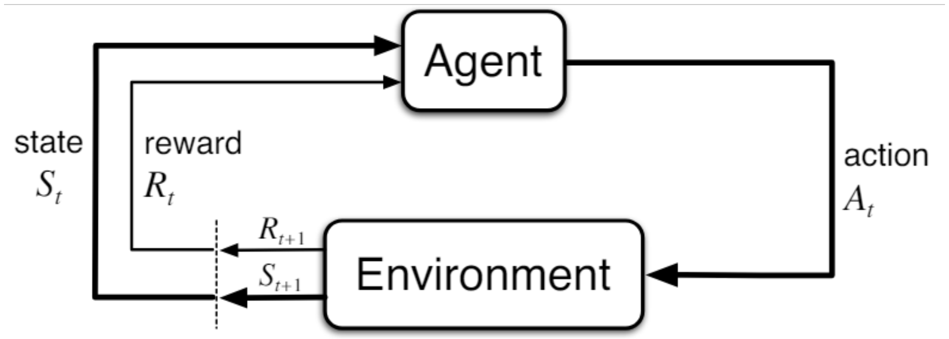


Figure 2.2: The agent-environment interaction in reinforcement learning.

Here, the environment is described using a formal model consisting of state, action, and reward spaces. Each of these elements plays a crucial role in defining the learning process:

State Space

The state space S encompasses all potential configurations of the environment or agent, with each state $s_t \in S$ offering essential information for decision-making. This space can provide complete details in fully observable environments or limited data in partially observable settings, affecting the learning and decision-making complexity.

Action Space

The action space A includes all actions available to the agent in any state, where each action $A_t \in A$ influences state transitions and outcomes. This space can range from a limited set of discrete choices to a continuous range, depending on the application.

Reward Space

The reward space defines a function $R_t : S_t \times A_t \to \mathbb{R}$, assigning a numerical reward to each state-action pair. This function measures the immediate value of actions, guiding the agent to maximize cumulative rewards over time, while balancing short-term gains against long-term outcomes.

Objective

Combining these spaces, the agent's objective in reinforcement learning is to discover a policy $\pi : S_t \to A_t$ that maximizes the expected cumulative reward. This involves evaluating how good it is to be in a certain state or to take a specific action in a state, which is formalized through the concepts of the value function and Q-function. The learning process adjusts the policy based on the observed outcomes to improve the expected rewards over time.

The interaction model outlined in Figure 2.2 visually summarizes these concepts, showing how actions taken by the agent based on the current state lead to new states and rewards, thus closing the loop of decision-making in reinforcement learning.

2.3.2. Denoising Diffusion Policy Optimization (DDPO)

In the work of Black et al. [13], a novel approach to training diffusion models using reinforcement learning is proposed.

The foundation of this approach is a pre-existing diffusion model that could be pre-trained or randomly initialized. When a fixed sampler is in place, this model dictates a sample distribution $p_\theta(x_0|c)$. The reinforcement learning (RL) objective within the denoising diffusion context aims to maximize a reward signal r based on the generated samples and their respective contexts. Thus, the denoising diffusion RL objective function $J_{DDRL}(\theta)$ is expressed as:

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

. This sets the framework for applying RL to diffusion models, with the goal of optimizing for the maximum expected reward.

Denoising Diffusion Policy Optimization (DDPO) reinterprets the denoising process of diffusion models as a multi-step Markov Decision Process (MDP). This perspective allows the utilization of policy gradient methods to directly optimize the diffusion model parameters with respect to a reward function $r(x_0, c)$, which assesses the quality and relevance of generated samples in a specific context.

In DDPO, the denoising process is structured as a sequence of decisions within this MDP framework:

$$
\begin{array}{lll}
s_t \equiv (c, t, x_t) & \pi(a_t|s_t) \equiv p_\theta(x_{t-1}|x_t, c) & P(s_{t+1}|s_t, a_t) \equiv (\delta_c, \delta_{t-1}, \delta_{x_{t-1}}) \\
a_t \equiv x_{t-1} & \rho_0(s_0) \equiv (p(c), \delta_T, N(0, I)) & R(s_t, a_t) \equiv \begin{cases} r(x_0, c) & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases}
\end{array}
$$

In the MDP framework, transitions span T timesteps culminating in a terminal state. The aggregate reward for a trajectory is given by $r(x_0, c)$, thus optimizing $J_{DDRL}(\theta)$ aligns with the RL objective $J_{RL}(\pi)$. Employing a standard sampler, π takes the form of an isotropic Gaussian, simplifying the model from the complex distribution $p_\theta(x_0|c)$. This allows for straightforward evaluation of log-likelihoods and their gradients with respect to model parameters.

Given the capacity to compute likelihoods and their gradients, direct Monte Carlo estimation of the gradients of $J_{DDRL}(\theta)$ becomes feasible. DDPO leverages this capability to collect denoising trajectories $\{x_T, x_{T-1}, \ldots, x_0\}$ and perform parameter updates via gradient descent.

The first variant, DDPO with Score Function (DDPOSF), employs the score function policy gradient estimator, also known as REINFORCE:

$$
\nabla_\theta J_{DDRL}(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log p_\theta(x_{t-1}|x_t, c) r(x_0, c)\right]. \tag{2.13}
$$

This expectation is over the denoising trajectories produced by the current parameters θ. While this estimator is straightforward, it permits only a single optimization step for each round of data collection.

To facilitate multiple optimization steps, an importance sampling estimator is used (DDPOIS). This allows the optimization to leverage data collected from a previous policy θ_{old}:

$$
\nabla_\theta J_{DDRL}(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \frac{p_\theta(x_{t-1}|x_t, c)}{p_{\theta_{old}}(x_{t-1}|x_t, c)} \nabla_\theta \log p_\theta(x_{t-1}|x_t, c) r(x_0, c)\right]. \tag{2.14}
$$

Inaccuracy can arise if p_θ diverges significantly from $p_{\theta_{old}}$. To manage this, trust regions are applied to constrain the magnitude of parameter updates. Practically, this is often achieved using a clipping mechanism as in Proximal Policy Optimization (PPO).

The success of DDPOSF and DDPOIS hinges on the reward function design, $r(x_0, c)$. An effective reward function is central to guiding the model towards desired outcomes, making its design an important component in the application of these reinforcement learning algorithms to diffusion models.

2.4. Continual Learning

The following sections on Continual Learning are based on the book [[11]].

2.4.1. General Definition

In this section, we explore the concept of *continual learning*, also known as *life-long learning*. This learning paradigm involves a system acquiring knowledge from a series of varying data distributions, denoted as $p_1, p_2, \ldots$. A key characteristic of continual learning is the sequential nature of the learning process, where at each time step t, the model is presented with a batch of labeled data:

$$D_t = \{(x_n, y_n) \sim p_t(x, y) : n = 1 : N_t\}. \tag{2.15}$$

Here, $p_t(x, y)$ represents the unknown data distribution, formulated as $p_t(x)p(y|f_t(x))$, with $f_t : X_t \to Y_t$ being the unknown prediction function. Each distinct data distribution encapsulates a unique task.

The learner's objective is to continually update its understanding of the underlying distribution, employing this knowledge to make predictions on an independent test set:

$$D^t_{\text{test}} = \{(x_n, y_n) \sim p^t_{\text{test}}(x, y) : n = 1 : N^t_{\text{test}}\}. \tag{2.16}$$

In class incremental learning, we focus on scenarios where the model encounters new class labels over time, affecting the evolution of the distribution $p_t(x, y)$. At any time t, the test set may include samples from the current class labels Y_t, as well as all previously introduced labels. This scenario emphasizes the need for the model to prevent *forgetting* previous classes while adapting to new ones.

Contrasting this, we also consider situations where the test distribution only includes the current class labels. Here, the model's adaptability is tested, requiring efficient learning and integration of new classes without the need to retain detailed information about past classes. This approach aligns with online learning principles but poses unique challenges in balancing new learning with retention of relevant past knowledge.

2.4.2. Class Incremental Learning

Class incremental learning, a widely studied form of continual learning, focuses on scenarios where new class labels are introduced over time. The model assumes a true static prediction function $f : X \to Y$, but at each step t, it only encounters samples from (X, Y_t) where $Y_t \subset Y$. For example, in digit classification from images, Y_1 might include 0, 1, while Y_2 might include 2, ..., 9, demonstrating the expanding label set.

This form of learning has been explored under various assumptions. If no well-defined task boundaries exist, we encounter continuous task-agnostic learning. With well-defined boundaries, we differentiate two sub-cases: discrete task-agnostic learning when boundaries are unknown during training, and task-aware learning when boundaries are known.

In task-aware scenarios, a common approach is to modify the MNIST and FashionMNIST datasets, either by permuting pixels across all 10 classes (permuted MNIST) or presenting a subset of 2 classes at each step (split MNIST and split FashionMNIST). The task-aware setting can vary in whether the task identity is known during testing, impacting the learning approach.

Class incremental learning can thus be seen as a problem with hierarchical output space, representing a cross-product of task ID and class label. The challenge lies in efficiently updating the model to accommodate new classes while retaining knowledge of previous ones, balancing between catastrophic forgetting and the flexibility to adapt to new information.

2.4.3. Typical Phenomena in Continual Learning

In the realm of continual learning, specifically in scenarios of class incremental learning, several phenomena are observed that significantly influence the learning dynamics [11]. This section illustrates these phenomena, each represented through conceptual figures, highlighting their implications on the learning process.

2.4.3.1. Catastrophic Forgetting

Catastrophic forgetting occurs when a model, upon learning new tasks, loses the ability to perform well on previously learned tasks. This phenomenon is particularly evident in class incremental learning, where previously learned classes can become less recognizable as new classes are introduced. Figure (2.3) illustrates how learning new tasks incrementally may degrade performance on earlier tasks if not managed properly. Here, the dotted line represents the expected performance threshold.

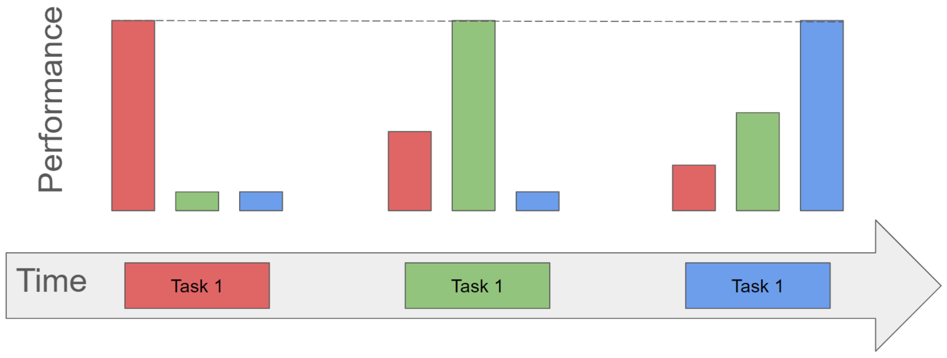


Figure 2.3: Catastrophic forgetting

2.4.3.2. Forward Transfer

Forward transfer refers to the beneficial effect where learning previous tasks improves the performance on future tasks. This phenomenon suggests that earlier learning can establish foundational knowledge that aids in quicker or more effective learning of subsequent tasks. In Figure 2.4 its depicted how training on past tasks can enhance the model's ability to learn new tasks beyond what would be achievable if it had started learning from scratch.

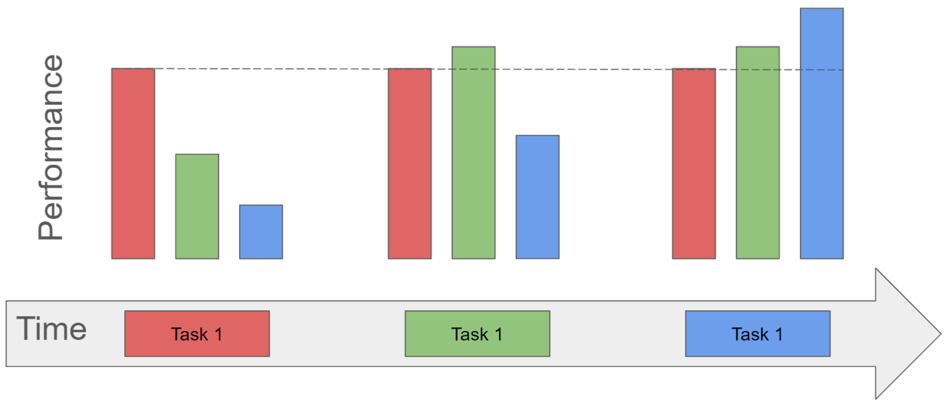


Figure 2.4: Forward transfer

2.4.3.3. Backward Transfer

Backward transfer involves the situation where learning new tasks improves performance on previously learned tasks. This can occur when newer information helps refine or correct the model's understanding of earlier tasks. The following visual representations (See Figure 2.5) shows how training on future tasks can enhance the performance on past tasks, sometimes correcting past inaccuracies or solidifying the knowledge base.

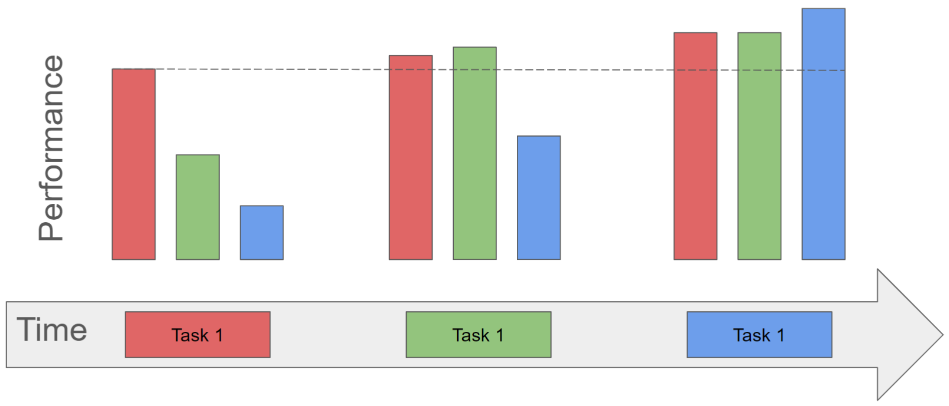


Figure 2.5: Backward Transfer

As we progress through the exploration of diffusion models within continual learning scenarios, we will observe how these phenomena—catastrophic forgetting, forward transfer, and backward transfer—manifest in practice. The impact of these phenomena will be examined in detail as we implement different continual learning strategies with our diffusion models.

2.5. Evaluation of Generative Models

Evaluating the effectiveness of generative models, particularly in the domain of image generation, involves several metrics that capture different aspects of model performance. This section details the methodologies and metrics implemented to assess the quality and diversity of the generated samples.

2.5.1. Fréchet Inception Distance (FID)

The Fréchet Inception Distance (FID) [14, 15] is a widely used metric for evaluating the quality of images generated by models [15]. It measures the distance between two distributions: the generated images and the real images from the training set. Specifically, the FID calculates the Fréchet distance between two Gaussian distributions, one fitted to feature vectors of the real images and the other to the generated images. These feature vectors are typically extracted using a layer of the Inception network, hence the name.

$$\mathrm{FID} = \|\mu_m - \mu_d\|^2 + \mathrm{tr}(\Sigma_d + \Sigma_m - 2(\Sigma_d \Sigma_m)^{1/2}). \tag{2.17}$$

Here, μ_m and Σ_m are the mean and covariance of the model samples, and μ_d and Σ_d are those of the real data samples. A lower FID score indicates better model performance, as it suggests that the two distributions are closer, implying that the generated images more closely resemble the real images.

For the evaluation in this research, the FID score is calculated using the ‘torchmetrics‘ library, which utilizes a pretrained Inception v3 model to extract the features from both the generated and real images. While it is possible to train a custom model for feature extraction, using a well-established, pretrained model like Inception v3 is preferred in this context. This approach leverages the model advanced image recognition capabilities to ensure that the features used for calculating the FID are robust and representative of all sets of images. Using a pretrained model allows us to focus more on the core aspects of our work rather than on the intricacies of model training for evaluation purposes.

2.5.2. Accuracy Over Specific Classifiers

Accuracy over classifiers is a key metric used to evaluate the effectiveness of generative models, focusing on the discriminative accuracy of generated images. This metric quantifies how well a model's outputs are classified by a specific, pre-trained classifier and is indicative of the realism and correctness of the generated images relative to known labels.

Formally, the accuracy is defined as:

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y_i)}{N}. \tag{2.18}$$

where N is the total number of images tested, $\hat{y}_i$ is the predicted class label for the i-th image, y_i is the true class label, and $\mathbf{1}$ is the indicator function that is 1 if $\hat{y}_i = y_i$ and 0 otherwise.

This metric is derived by feeding generated images to a classifier that has been trained on the original dataset. A high accuracy score suggests that the generated images align closely with the class attributes defined in the training set. Thus, this metric underscores the model capacity to produce images that are correctly categorized.

Chapter 3

Proposed Methodology

This chapter outlines the proposed methodology designed to enhance continual learning in generative models. After reviewing classical approaches and their limitations, we present improved strategies and novel alternatives. Accompanying these descriptions are illustrative diagrams depicting the training regimes across three separate tasks, each comprising two classes. These visual representations serve to clarify the different methodologies and illustrate how they address the challenges associated with continual learning.

3.1. Classical Approaches to Continual Learning

3.1.1. Fine-tuning

Fine-tuning is a foundational technique in the field of continual learning for generative models. It is predicated on the principle of incremental learning, where a model pre-trained on a specific task is subsequently adjusted to perform additional tasks. As depicted in Figure 3.1, the model initially trained to generate classes 0 and 1 (Task 1) is sequentially updated to handle classes 2 and 3 (Task 2), and later classes 4 and 5 (Task 3).

In each stage of fine-tuning, the model parameters are optimized to minimize the loss on the new tasks data. However, without mechanisms to retain previously acquired knowledge, the model performance on prior tasks tends to degrade, a phenomenon known as catastrophic forgetting [5, 7]. This issue will be further explored and demonstrated in subsequent chapters through corresponding experiments.

Despite the simplicity and directness of the fine-tuning approach, its susceptibility to forgetting makes it less ideal for scenarios that require the model to maintain performance across all learned tasks. Figure 3.1 shows the sequential nature of training in fine-tuning, which, while straightforward, necessitates the development of additional strategies to mitigate forgetting and sustain model robustness throughout the learning process.

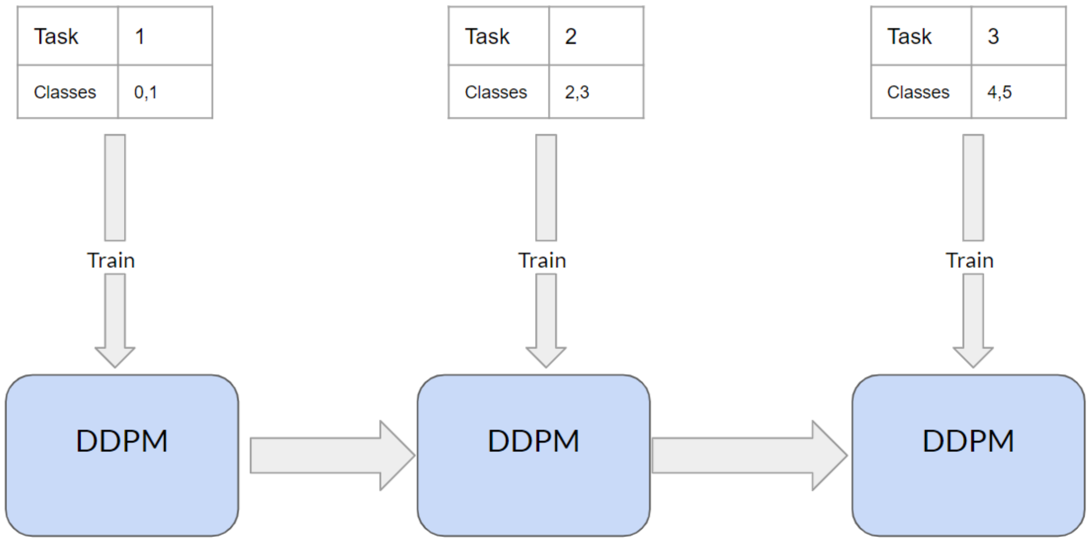


Figure 3.1: Illustration of the fine-tuning process across three sequential tasks. The DDPM model is first trained on Task 1 and subsequently fine-tuned for Tasks 2 and 3, with each task introducing new classes.

3.1.2. Experience Replay

Experience Replay addresses catastrophic forgetting by incorporating the rehearsals of prior knowledge into the continual learning process [3]. As visualized in Figure 3.2, this method involves a continual learning model that retains a memory buffer, wich are samples from past data. When the model progresses to a new task, for example moving from learning classes 0,1 to classes 2,3, it doesn't only focus on the new classes, instead it reuses the stored samples from the buffer along side the new samples. This process is repeated as the model encounters new tasks, with the buffer accumulating a diverse set of samples to ensure comprehensive rehearsal.

In practical terms, Experience Replay can be implemented in generative models through mechanisms such as a fixed-size memory buffer that captures the essence of previous data or through more dynamic approaches that adapt the stored samples based on their significance. The end goal is to maintain the generative model's performance across all learned distributions, ensuring it generates high-quality and diverse outputs as it continues to learn new tasks.

One significant drawback of Experience Replay is the requirement for additional memory to store data from previous tasks. This can be problematic when dealing with large datasets or when the number of tasks is high, leading to scalability issues.

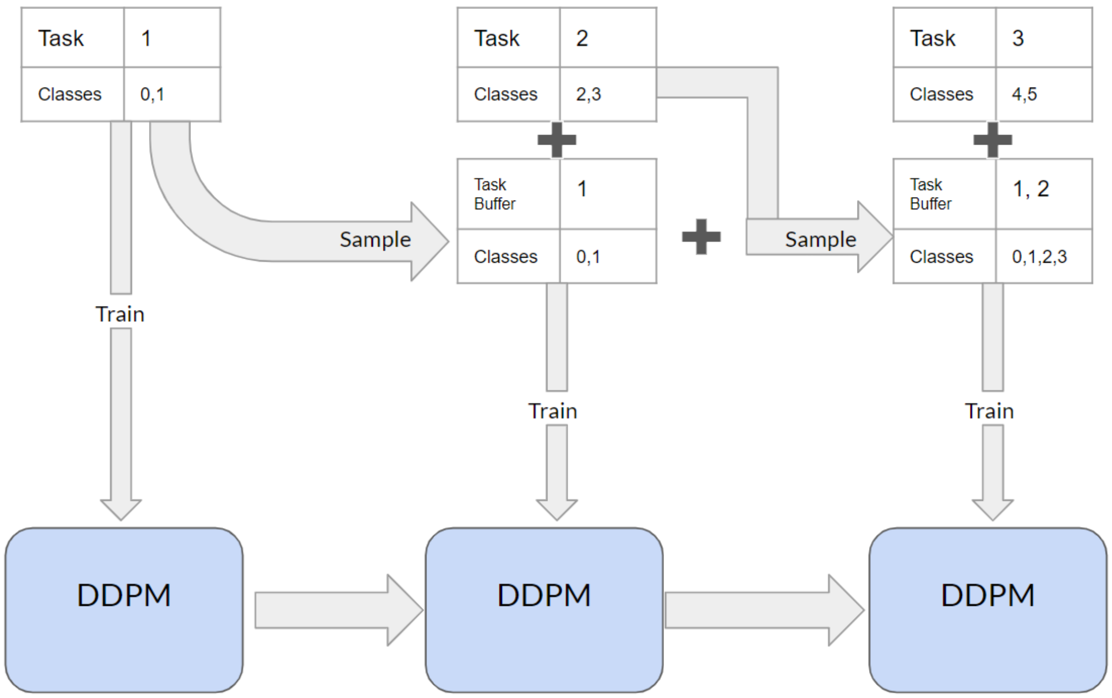


Figure 3.2: The process of Experience Replay in continual learning for generative models, showing the training sequence over three tasks and the integration of old and new data in a buffer.

3.1.3. Generative Replay

Generative Replay, an evolution of Experience Replay, uses a generative model to recreate the data of previous tasks [3], addressing the problem of catastrophic forgetting with greater efficiency. As illustrated in Figure 3.3, after the primary model is trained on Task 1 (classes 0,1), the generative model samples examples from this learned distribution to retain the knowledge. When the model progresses to Task 2 (classes 2,3), it not only learns from the new data but also from the synthetic samples of Task 1, preserving previous knowledge without the need to store actual past data. This process is cyclically repeated for the next tasks, such as Task 3 (classes 4,5), each time leveraging synthetic samples from all previous tasks to ensure learning continually.

This strategic replay of generated samples is not only memory efficient, but also promotes a more nuanced retention of knowledge, as the generative model can be fine-tuned to produce samples that are most beneficial for the current learning phase. Consequently, Generative Replay could be a tool used in continual learning frameworks for generative models, ensuring that the synthesis of new data does not erode the fidelity of previously acquired knowledge.

A notable drawback of Generative Replay is the degradation of synthetic data quality over time [3, 16]. As the generative model reproduces data for earlier tasks, slight deviations can accumulate, leading to a drift from the original data distribution and characteristics. This issue is known as 'data drift' and can cause the model to lose its knowledge of previous tasks.

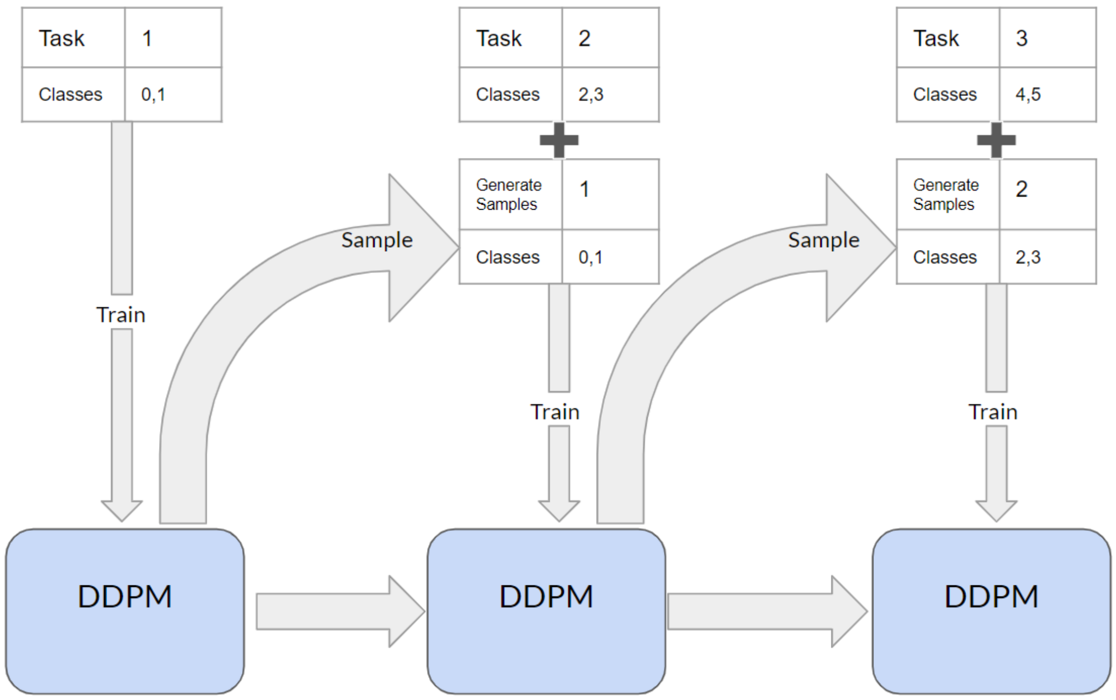


Figure 3.3: The Generative Replay process in continual learning, depicting the training flow over three tasks with synthetic sample generation for knowledge retention.

3.2. Proposed Improved Approaches

This chapter introduces two novel approaches that apply reinforcement learning to adjust diffusion model outputs. The core hypothesis is that a model can be steered to generate more representative samples of a dataset by employing a reward function that measures sample quality. By doing so, we aim to align the model closely with the data distribution, enhancing the utility of generative replay in continual learning scenarios.

3.2.1. Aligning Diffusion Model Outputs in Continual Generative Replay

Our approach, depicted in the schematic below, is the integration of reinforcement learning with generative replay, referred to as CL-RL (Continual Learning with Reinforcement Learning). Our hypothesis is that by enhancing the sample quality through reinforcement learning, specifically with Denoising Diffusion Probabilistic Models (DDPMs) optimized using the Denoising Diffusion Policy Optimization (DDPO) algorithm, we can maintain closer alignment with the original data distribution.

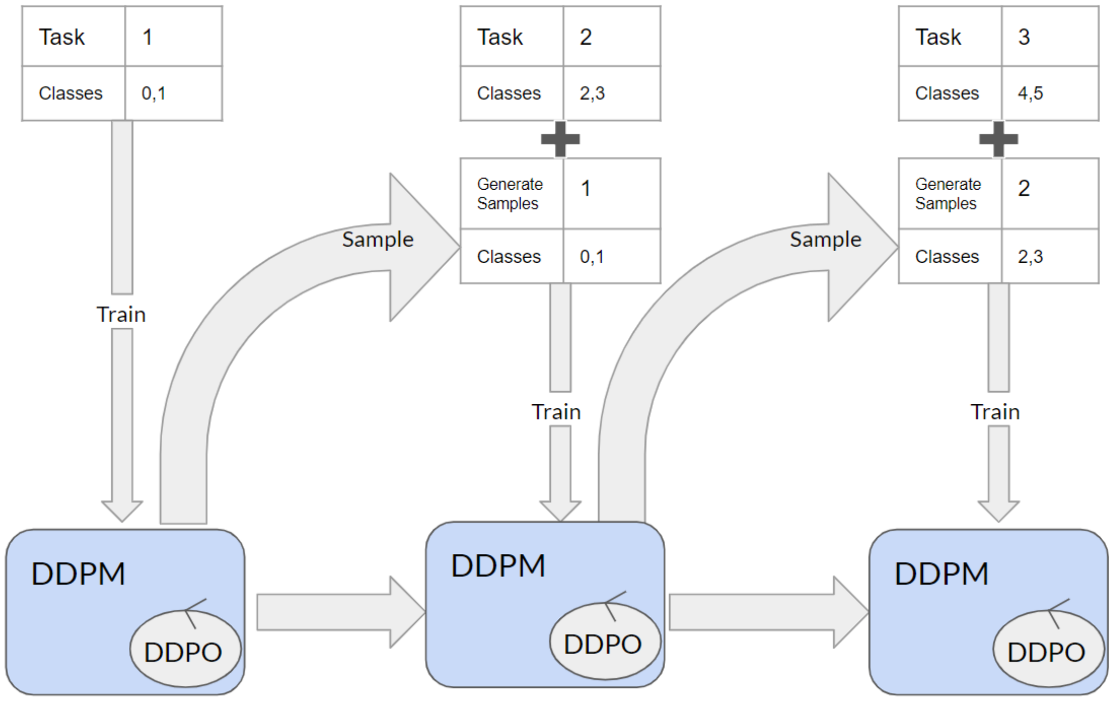


Figure 3.4: Improving Sample Quality Directly on The Diffusion Model

The method involves initially training a diffusion model on the original dataset, referred to as Task 1. Subsequently, but prior to commencing training on Task 2, a reward model is employed to enhance the quality of the samples generated by the diffusion model. This iterative process of training and refinement via Reinforcement Learning is repeated throughout the successive stages of continual learning.

As a result, the model is anticipated to maintain a more consistent performance through a series of tasks, addressing catastrophic forgetting by progressively preserving its capacity to regenerate the core attributes of the data.

3.2.2. Teacher-Student Diffusion Models in Continual Generative Replay

Experimentation shows that Reinforcement Learning, while improving diffusion model sample quality, can cause overparametrization, limiting the model's adaptability to new tasks. To address this, we introduce a teacher-student method in continual generative replay.

This method uses two versions of the same base model: the Student remains unchanged after initial training to retain learned information, and the Teacher, actively trained with Reinforcement Learning, refines sample generation. Guided by a reward function, the Teacher model produces enhanced samples for retraining the Student alongside new data, mitigating overfitting to previous tasks.

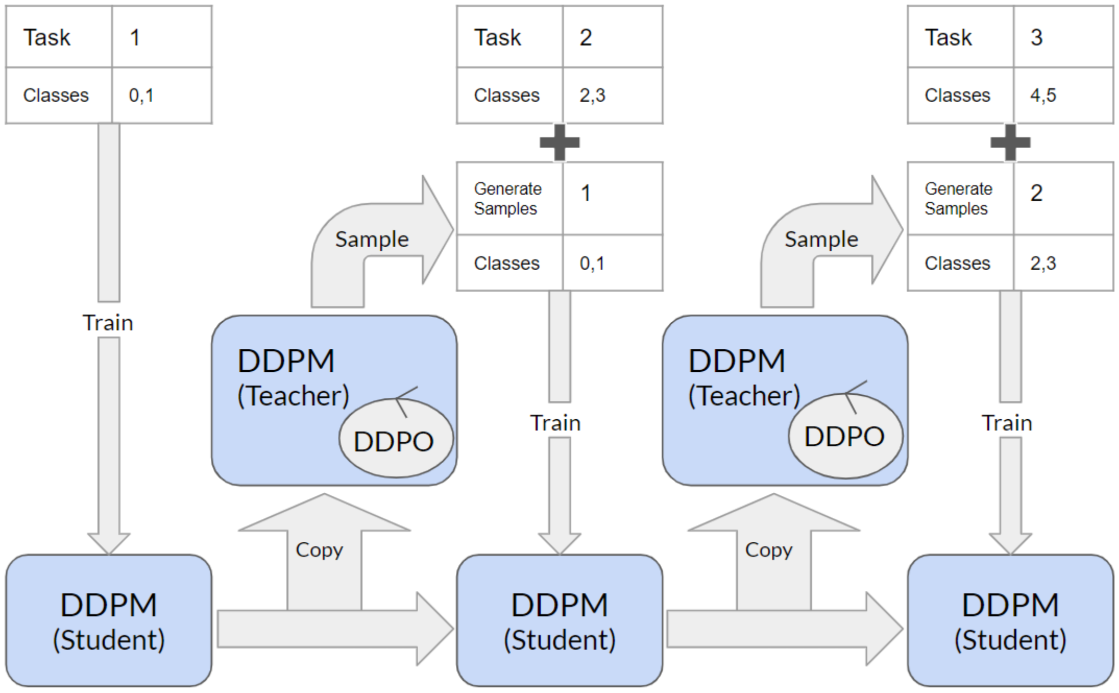


Figure 3.5: Improving Sample Quality on a Teacher Diffusion Model

This dual strategy ensures the student model continually benefits from refined generative replay, preventing catastrophic forgetting up to a point, and promoting better performance throughout successive learning tasks. The model progresses by alternating between receiving enhanced samples from the Teacher and training on subsequent tasks, thus maintaining a more balance aproach between learning new information and retaining previous knowledge.

3.3. Reward Functions Design

The training regimes with Reinforcement Learning heavily rely on the design of its reward functions. These functions are needed for guiding the learning process and setting the model's goals. Specifically in our case, the reward functions must be dynamic and robust, adapting to evolving data distributions to consistently direct the model toward producing high-quality outputs.

In this study, we utilize a basic classification model as the reward function. This approach rests on the premise that a classifier can differentiate between correct and incorrect samples, without bias towards their specific characteristics. This ensures that the diffusion model produces samples that accurately reflect the targeted classes. The inherent diversity of the model's generative capabilities is assumed to maintain variation in the output.

3.3.1. Classifier Alignment

This approach employs the output of a classifier as the reward mechanism within our reinforcement learning framework. The procedure begins by prompting the diffusion model to generate a sample that ideally belongs to a specified class of interest. Once the sample is generated, it is fed into a classifier, which then assesses the likelihood that the sample belongs to the intended class (See Figure 3.6). This probability serves as a measure of alignment and is used to compute the reward for the diffusion model.

The reward is directly proportional to the classifiers confidence that the generated sample accurately represents the desired class. This method not only reinforces the generation of accurate samples but also integrates with the diffusion model capabilities to explore a diverse set of outputs.

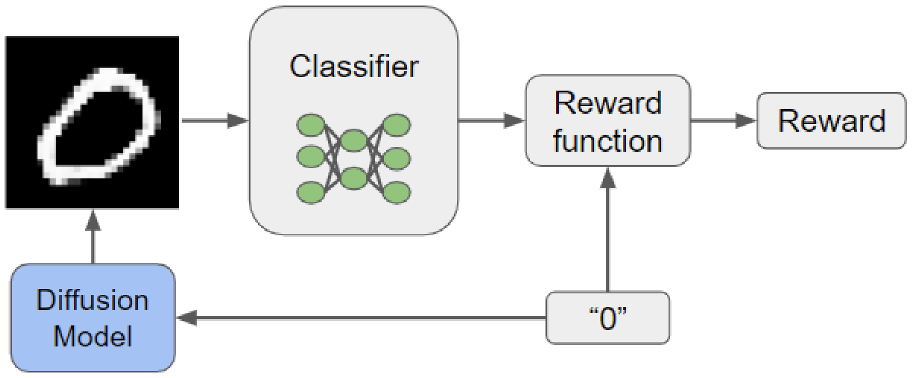


Figure 3.6: Classifier Output Rewards

3.3.2. Mixed Classifier and Cosine Distance Alignment

In this method, we extend the previous classifier based reward mechanism by incorporating cosine similarity to further refine the reward system. Initially, the process is similar to that of the Classifier Alignment, the diffusion model is tasked with generating a sample aimed to represent a specific class. Simultaneously, a subset of data is sampled from the original model, before further refinement with Reinforcement Learning, to serve as a reference set.

Once the sample is generated, its probability of belonging to the targeted class is evaluated by the classifier. In addition, the average cosine similarity between the generated sample and the reference set is computed. These two metrics, the classifier probability and the cosine similarity, are then combined to create a mixed reward metric (See Figure 3.7).

This mixed metric approach serves as an implicit regularization technique, ensuring that the reward does not cause the model to deviate excessively from its original sample generation capabilities. By integrating both class alignment and similarity to the original generated data distribution, this method prevents the model from overly concentrating on certain modes of generation, thus better preserving generative diversity and maintaining high sample quality.

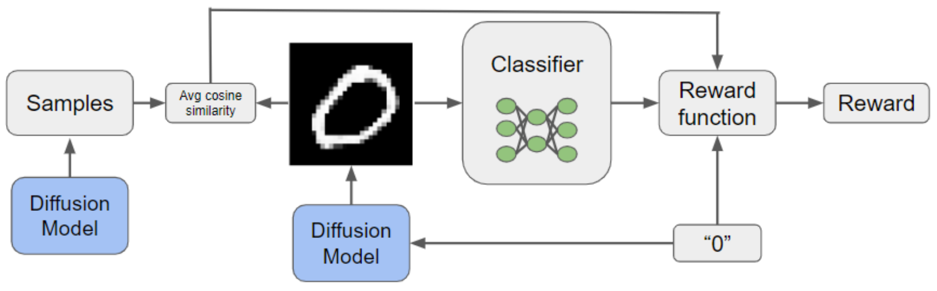


Figure 3.7: Classifier and Cosine Distance Reward

Chapter 4

Experiments

4.1. Dataset

For our experiments, we employed two datasets: MNIST[1] and FashionMNIST[2]. These datasets are standard benchmarks in machine learning for image recognition. MNIST, which contains handwritten digits, is ideal for simple classification and generative tasks. On the contrary, FashionMNIST offers a more diverse set of classes with various clothing items, posing a more complex challenge.

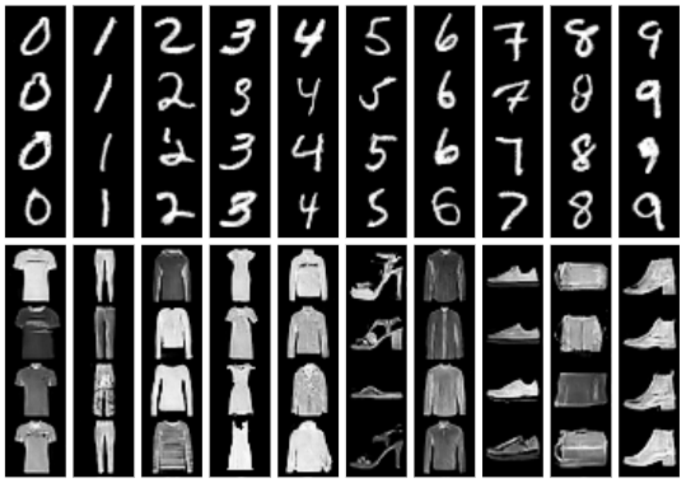


Figure 4.1: Sample images from the MNIST (top) and FashionMNIST (bottom) datasets. Image obtained from [17]

In the context of our study, both datasets were adapted to suit continual learning scenarios with diffusion models. We splited these datasets into five separate tasks, composed of two classes each. For example, in MNIST, the first task correspond to the classes ’0’ and ’1’, while in FashionMNIST the first task correspond to the classes ’T-shirt’ and ’Trouser’. Table 4.1 contains the dataset divisions in different tasks.

[1] MNIST dataset: http://yann.lecun.com/exdb/mnist/

[2] FashionMNIST dataset: https://github.com/zalandoresearch/fashion-mnist

Table 4.1: Tasks and their corresponding classes for MNIST and Fashion-MNIST datasets.

Task	**MNIST Classes**	**FashionMNIST Classes**
1	0, 1	T-shirt/top, Trouser
2	2, 3	Pullover, Dress
3	4, 5	Coat, Sandal
4	6, 7	Shirt, Sneaker
5	8, 9	Bag, Ankle boot

All the diffusion models used in our experiments were trained sequentially on these tasks. Each model is tasked with generating images that the classifier, trained on the complete dataset, evaluates. The classifier provides rewards based on the likelihood that the generated images belong to the correct classes. This reward system supports the diffusion models in refining their generative abilities across successive tasks, embodying a continual learning framework where the model adapts to new data while retaining effectiveness on previously learned tasks.

4.2. Diffusion Model Design an DDPO Training

4.2.1. Architecture of the diffusion model

Our diffusion model utilizes the U-Net architecture with class conditioning, wich is commonly used in training these smaller models. At its core, it features an embedding layer that infuses class information into the feature space, enhancing class-specific generation. Attention mechanisms are embedded within the architecture, vital for preserving spatial details through both the contraction and expansion paths of the network.

Table 4.2: Key enhancements to the U-Net architecture

Feature	**Description**
Class Embedding	An embedding layer to encode class information and guide the generative process.
Attention Modules	Placed in downsampling and upsampling stages to retain fine details.
Network Depth	Additional ResNet layers increase model capacity.
Channel Width	More channels per layer to capture complex features.
Resolution	Operates on 28x28 images, aligning with dataset standards.
Output	Single-channel output tailored for grayscale images.

The architecture's design is specifically tuned for 28x28 images and is configured to generate single-channel outputs, catering the MNIST and FashionMNIST data structure. By integrating class-conditioning with a fortified U-Net structure, the model is well-equipped for continual learning, adept at producing class specific images. For the technical specifics, see the complete `ClassConditionedUnet` class definition in Annex A.

4.2.2. DDPO Training Algorithm

We are going to outline the DDPO algorithm used for further refining the sample quality of the generated images. The algorithm is built to iteratively improve the model by adjusting to the rewards computed based on classifier outputs. Below we outline the main components of this training algorithm.

Algorithm Overview

The DDPO training algorithm enhances the diffusion model by leveraging a classifier model to provide policy updates. The procedure involves generating samples, computing rewards, and adjusting model weights based on the reward signals.

Initialization

At the beginning of training:

- Model weights for both the diffusion model and classifier are loaded from predefined paths.
- The models are transferred to the appropriate computational device (e.g., GPU).
- An AdamW optimizer is initialized with specific learning parameters and weight decay to manage the model updates.

Advantage Calculation

A custom advantage tracker is used to store and update advantages based on the rewards received and the predictions made. This tracker helps in normalizing the rewards and scaling the policy updates effectively.

Sampling and Reward Computation

For each epoch:

- The model samples a set number of images, divided into batches according to class labels.
- Each sample undergoes evaluation where a reward is computed using a classifier model. The reward reflects how well the sample aligns with the desired output characteristics.

Policy Update

After collecting samples and rewards, for each inner epoch:

- Rewards and other metrics are consolidated across all batches.
- The model enters a training loop where the optimizer updates the model weights based on the computed loss, which integrates the rewards (via advantages) and log probabilities.
- Gradient clipping is applied during optimization to stabilize training.

Optimization and Logging

Throughout the training process, the optimization steps are carefully monitored, and performance metrics such as loss are logged for each batch. This iterative process helps in fine-tuning the model by focusing on enhancing the generation of high-quality, class-specific images.

Output

At the end of training, the updated diffusion model is returned along with a log of all rewards collected during the training, providing insights into the model's performance and the effectiveness of the training regimen.

This training algorithm is central to our approach, enabling the diffusion model to adapt continually and effectively to the task requirements while ensuring the generative quality remains high. The full algoritm can be found below.

Algorithm 1 Train diffusion model using DDPO

```
Require: diffusion_model, classifier_model, diffusion_model_path, classifier_model_path
Require: num_epochs, num_inner_epochs, num_samples_per_epoch, batch_size
Require: lr, eta, clip_advantages, clip_ratio, device, class_labels
 1: Load state dicts for diffusion_model and classifier_model from paths
 2: Transfer models to computational device
 3: Initialize optimizer (AdamW) with learning rate and weight decay
 4: Initialize custom advantage tracker
 5: for epoch in 1 to num_epochs do
 6:     Initialize lists for predictions, log probabilities, advantages, labels, rewards
 7:     Calculate number of batches from num_samples_per_epoch and batch_size
 8:     Divide class_labels into batches
 9:     for batch in 1 to num_batches do
10:         Sample images with diffusion_model using current batch labels
11:         Compute rewards using classifier_model
12:         Update advantage tracker with rewards
13:         Store intermediate states, log probabilities, rewards
14:     end for
15:     Concatenate all batch data
16:     for inner_epoch in 1 to num_inner_epochs do
17:         for sample in batched data do
18:             Perform gradient descent to update diffusion_model
19:             Apply gradient clipping
20:             Log training loss
21:         end for
22:         Clear computational cache
23:     end for
24: end for
25: return trained diffusion_model and all rewards
```

A key feature of this algorithm is its division into two learning phases: exploration and exploitation. The exploration phase, defined by the *num_epochs* parameter, involves generating new images for evaluation. This phase aims to explore a variety of strategies. The exploitation phase, defined by the *num_inner_epochs*, uses these images for multiple learning rounds to refine the model performance and strengthen successful strategies. A more detailed analysis of both phases can be found in the Annex B, where the performance of the Reinforcement Learning algorithm is further studied.

4.3. Detailed Overview of the Classifiers for Continual Learning with Reinforcement Learning

This section explores the architectures and performances of classifiers integral to our continual learning framework. These classifiers are crucial for enhancing the diffusion models, ensuring the output samples are both diverse and class-accurate. Their ability to accurately distinguish between correct and incorrect samples significantly impacts the effectiveness and reliability of the learning process. Thus, developing robust classifiers is essential for improving model performance in continual learning scenarios.

In the following subsections, we detail the specific architectures employed for the MNIST and FashionMNIST classifiers, along with performance metrics evaluated on their respective testing datasets. Additionally, observations on the classifiers performance will be discussed, providing insights into their efficacy within our continual learning setup.

4.3.1. MNIST architecture and performance

The MNIST classifier utilizes a fully connected neural network architecture. The network consists of three fully connected layers:

- **Input Layer**: Flattens 28×28 images into 784 units and feeds into the first hidden layer.
- **Hidden Layer 1**: Converts 784 units to 256 hidden units.
- **Hidden Layer 2**: Reduces 256 hidden units to 64 hidden units.
- **Output Layer**: Maps 64 hidden units to 10 output classes for the digits.

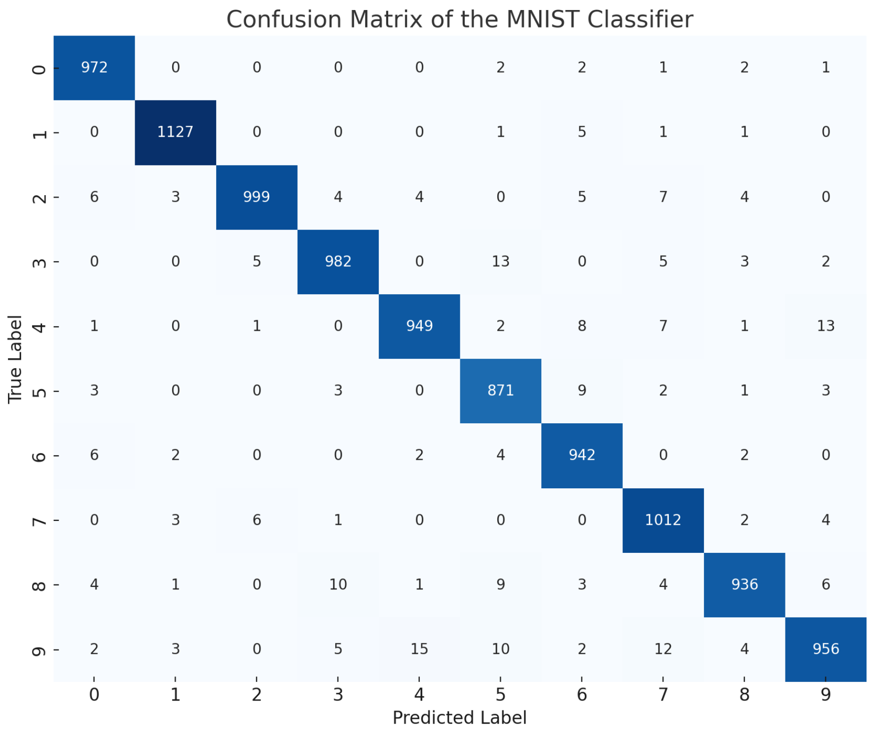
This architecture employs ReLU activations after each of the first two layers and does not use any convolutional or pooling layers, making it a purely feedforward model. The model was trained on the MNIST dataset for 10 epochs with a learning rate of 0.001 using the Adam optimizer and Cross-Entropy Loss as the criterion.

4.3.1.1. Performance Metrics

The performance of the MNIST classifier on the test dataset is summarized as follows:

Table 4.3: Performance metrics of the MNIST classifier.

Metric	**Value**
Accuracy	0.9746
Precision	0.9743
Recall	0.9744
F1 Score	0.9743



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	972	0	0	0	0	2	2	1	2	1
1	0	1127	0	0	0	1	5	1	1	0
2	6	3	999	4	4	0	5	7	4	0
3	0	0	5	982	0	13	0	5	3	2
4	1	0	1	0	949	2	8	7	1	13
5	3	0	0	3	0	871	9	2	1	3
6	6	2	0	0	2	4	942	0	2	0
7	0	3	6	1	0	0	0	1012	2	4
8	4	1	0	10	1	9	3	4	936	6
9	2	3	0	5	15	10	2	12	4	956

Figure 4.2: Confusion Matrix of the MNIST Classifier (Over 10 classes)

The classifiers high performance is essential for our reinforcement learning algorithm success. Its role is critical in accurately predicting whether samples from the diffusion model represent their respective classes, directly impacting the feedback mechanism.

High accuracy ensures that the reinforcement learning algorithm receives reliable signals to fine-tune the generative process. In contrast, poor classifier performance can lead to inaccurate feedback, quickly degrading the quality of the samples. The significance of this relationship will be further explored in our experiments with the FashionMNIST dataset, where we find that less than perfect performance can directly affect the model's generation capabilities, rendering the reinforcement learning in continual learning methodologies unstable.

4.3.2. FashionMNIST Architecture and Performance

The FashionMNIST classifier leverages a simple yet effective neural architecture designed for the FashionMNIST dataset, incorporating artificially corrupted image data to enhance training dynamics. The network structure includes:

- **Input Layer**: Converts flattened 28×28 images into a 784-unit vector.
- **First Hidden Layer**: Maps 784 input units to 256 hidden units.
- **Second Hidden Layer**: Reduces from 256 to 64 hidden units.
- **Output Layer**: Extends to 11 output units, 10 for the standard FashionMNIST classes plus one additional class for corrupted images.

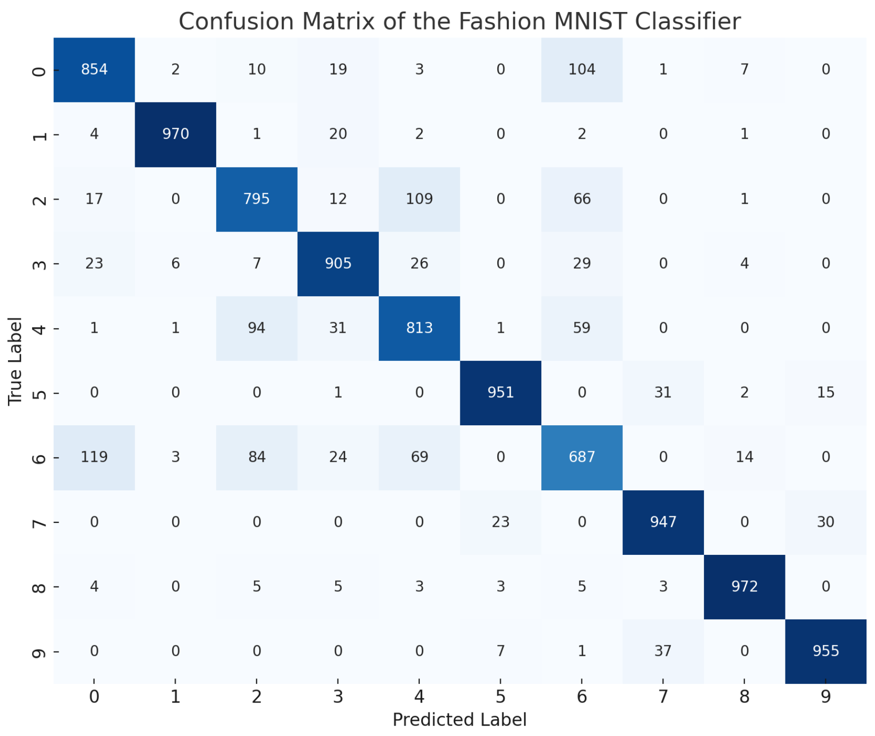
This model was trained over 10 epochs using the Adam optimizer with a learning rate of 0.001 and Cross-Entropy Loss. The inclusion of an artificial class aimed to simulate and identify corrupted images, facilitating the model's capability to handle low quality or altered images, similar to those generated by diffusion models.

4.3.2.1. Performance Metrics

Performance on the testing dataset is summarized below:

Table 4.4: Performance metrics of the FashionMNIST classifier.

Metric	**Value**
Accuracy	0.8849
Precision	0.8846
Recall	0.8849
F1 Score	0.8847


Confusion Matrix of the Fashion MNIST Classifier
True Label
Predicted Label

True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	854	2	10	19	3	0	104	1	7	0
1	4	970	1	20	2	0	2	0	1	0
2	17	0	795	12	109	0	66	0	1	0
3	23	6	7	905	26	0	29	0	4	0
4	1	1	94	31	813	1	59	0	0	0
5	0	0	0	1	0	951	0	31	2	15
6	119	3	84	24	69	0	687	0	14	0
7	0	0	0	0	0	23	0	947	0	30
8	4	0	5	5	3	3	5	3	972	0
9	0	0	0	0	0	7	1	37	0	955

Figure 4.3: Confusion Matrix of the Fashion MNIST Classifier (Over 10 classes)

4.3.2.2. Training with Corrupted Images

Training utilized the *FashionMNISTWithArtifacts* (See Annex C) dataset, which adds a class for corrupted images to mimic potential quality issues encountered in diffusion models. While this addition improved the model performance, by helping it distinguish between authentic and corrupted images, it has not achieved a level of performance that accurately simulates real preferences.

Consequently, this has sometimes led to the incorrect assignment of rewards, adversely impacting the reinforcement learning algorithm. This highlights the critical need for well calibrated reward functions in the learning dynamics of machine learning models, when they are subjected to Reinforcement Learning enviroments.

Chapter 5

Results

5.1. Results Overview

In this section, we present tables that include previously discussed metrics essential for interpreting and properly characterizing the success and failure of our implementations.

We will showcase our results through both quantitative metrics and visual examples. This dual approach will allow us to discuss the relevance of each evaluation method and their implications.

The tables provide the average values for each task performed on training the diffusion model in the MNIST and FashionMNIST datasets. While these averages offer a summary of performance, they do not fully capture the whole behavior of each individual task. For a more detailed analysis, the complete table of results is available in Annex D.

5.1.1. Results for Model Trained on MNIST

5.1.1.1. Quantitative Metrics

Table 5.1: MNIST Data FID Score Evolution (Lower is better)

Task Average Value	Finetuning	Gen Replay	Exp Replay	CL-RL		CL-RL-ts	
				Basic	Mixed	Basic	Mixed
Task 1	180.89	183.76	179.86	155.13	157.85	165.60	181.53
Task 2	224.98	205.78	153.43	196.75	204.73	214.09	203.40
Task 3	241.61	224.35	140.19	211.94	203.47	207.90	203.96
Task 4	267.50	240.82	141.75	239.08	235.65	225.27	229.21
Task 5	289.28	268.55	**140.90**	239.16	236.83	**217.69**	219.69
Average	240.85	224.65	151.22	208.41	207.7	206.10	207.55

Table 5.1 illustrates the FID score evolution across all five tasks, with lower scores indicating better performance. **Experience Replay** achieves the lowest FID scores, aligning well with the expected behavior and demonstrating its effectiveness in preserving sample quality across tasks. Although **Generative Replay** performs better than **Finetuning**, it still falls short compared to **Experience Replay**. All of our proposed methodologies outperform the baseline Generative Replay, with the best-performing one being the **Teacher-Student CL-RL with a Basic Reward**. While this approach does not fully bridge the gap between Experience Replay and Generative Replay, it shows promising results on this metrics.

Table 5.2: MNIST Data Accuracy Evolution

Task Average Value	Finetuning	Gen Replay	Exp Replay	CL-RL		CL-RL-ts	
				Basic	Mixed	Basic	Mixed
Task 1	0.845	0.795	0.830	0.913	0.898	0.840	0.735
Task 2	0.380	0.546	0.873	0.457	0.433	0.464	0.470
Task 3	0.320	0.497	0.695	0.405	0.424	0.478	0.453
Task 4	0.266	0.360	0.835	0.385	0.399	0.470	0.393
Task 5	0.187	0.289	**0.759**	0.367	0.343	**0.378**	0.377
Average	0.399	0.497	0.798	0.505	0.499	0.526	0.485

Table 5.2 presents the mean accuracy evolution across all five tasks, where accuracy is measured by the classifier used during training. Higher accuracy indicates better performance. Consistent with the results from the FID table, **Experience Replay** demonstrates the best performance, achieving the highest accuracy scores across all tasks. Our methodology, utilizing **Teacher-Student CL-RL with a Basic Reward**, obtains the second-best results, clearly outperforming the baseline implementations of **Generative Replay** and **Finetuning**. Although this metric favors our approach, it is important to note that our proposed reinforcement learning methodology is, in some sense, directly optimized for this metric. Consequently, while the accuracy scores offers insight into the class preservation capabilities of our methodology, they also reflect an expected outcome that aligns with our optimization objectives.

In the following figures we note the temporal evolution of the previously mentioned metrics, this enables a more comprehensive understanding of the learning dynamics through different tasks. We will further discuss this behaviour in the next chapter.

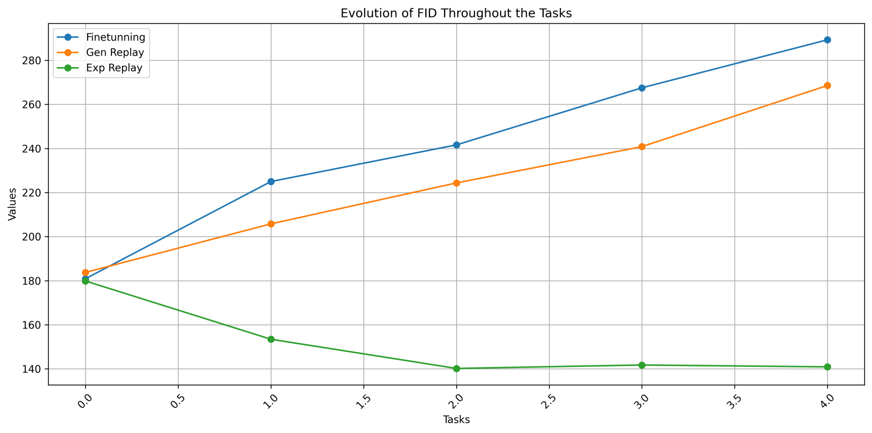


Figure 5.1: FID Evolution of Baselines: Evolution of baselines over the MNIST dataset

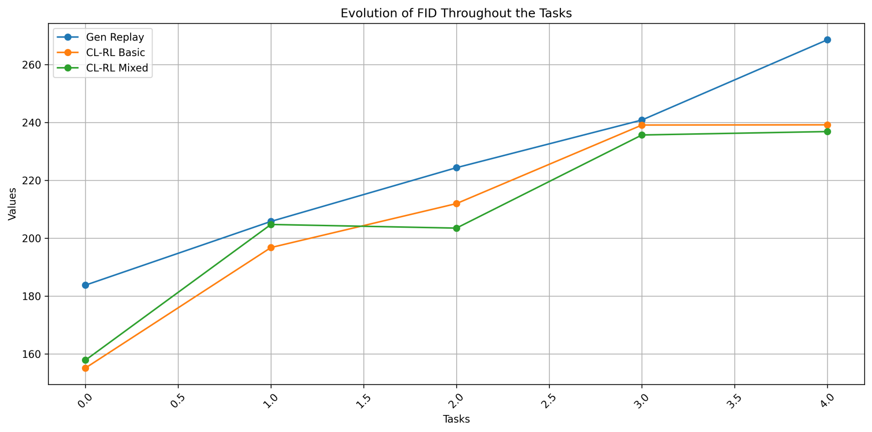


Figure 5.2: FID Evolution Base Methodology: Evolution for both reward configurations

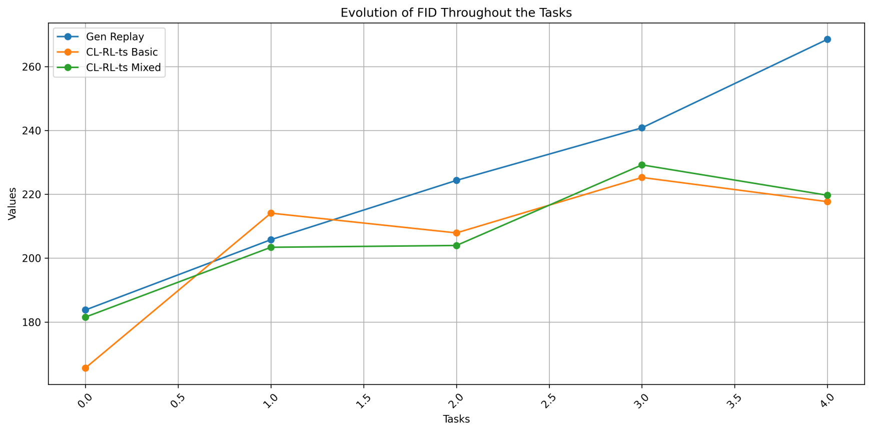


Figure 5.3: FID Evolution Teacher-Student Methodology: Evolution for both reward configurations

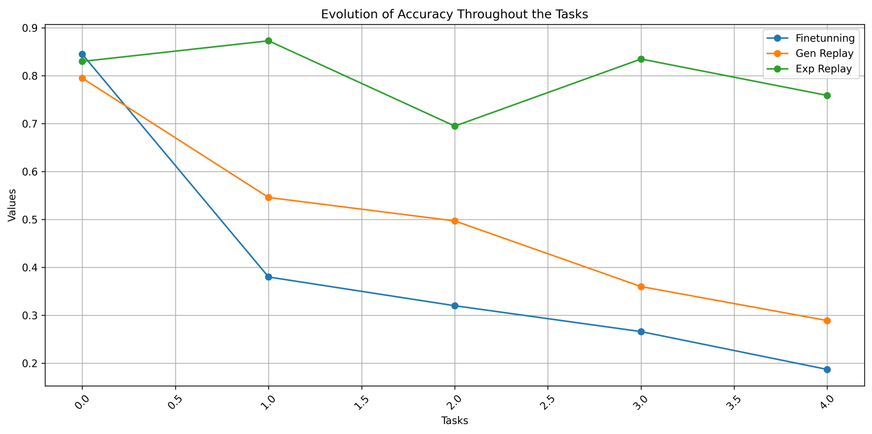


Figure 5.4: Accuracy Evolution of Baselines: Evolution of baselines over the MNIST dataset

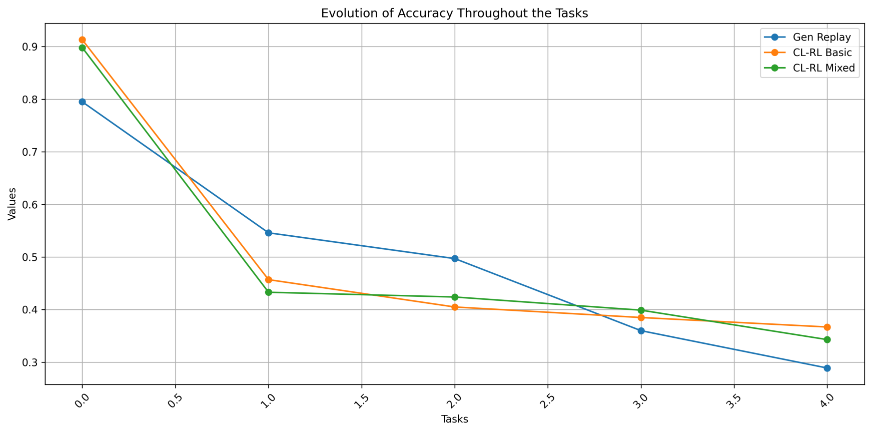


Figure 5.5: Accuracy Evolution Base Methodology: Evolution for both reward configurations

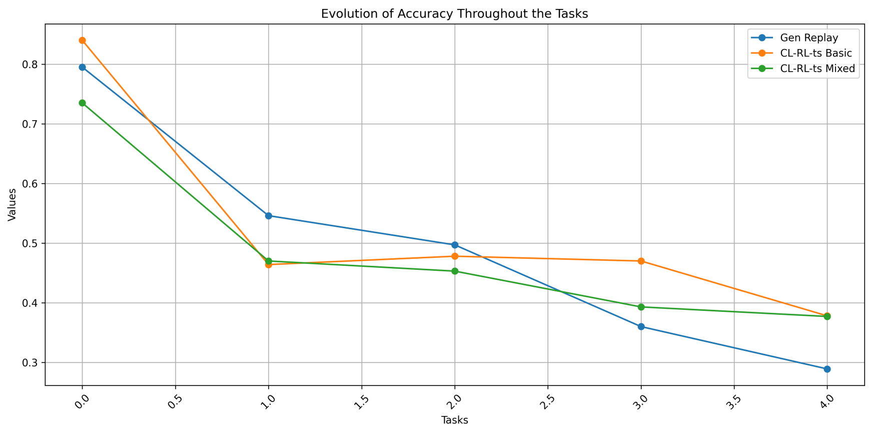


Figure 5.6: Accuracy Evolution Teacher-Student Methodology: Evolution for both reward configurations

5.1.1.2. Reference Images for Baselines

To provide more context and a deeper analysis, we present reference images of the generations obtained for every continual learning scenario. These images offer a more qualitative analysis that is not fully captured by the metrics obtained previously.

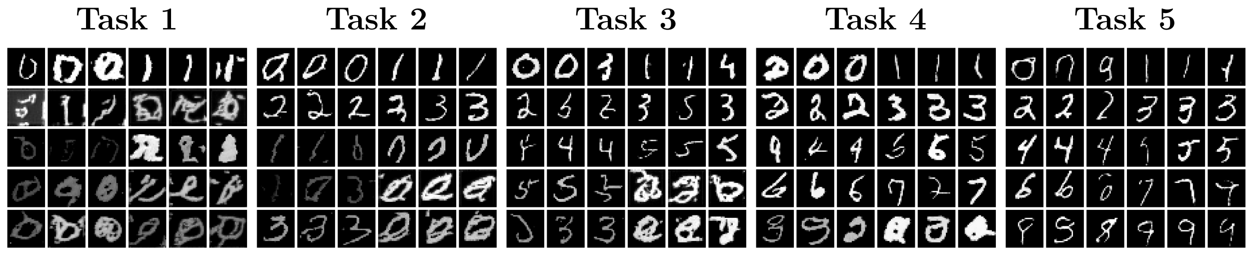


Figure 5.7: **Experience Replay**: Evolution of the generation capabilities for each class after each task. Each row corresponds to two classes, with the first three images of the first class and the last three images of the second class.

Figure 5.7 represents the best results we obtained in continual learning via replay methods. We observe a clear evolution over the different stages, with a mode collapse to previously seen classes, as expected due to the experience replay training setup. However, the performance aligns with the behavior observed in the previous metrics.

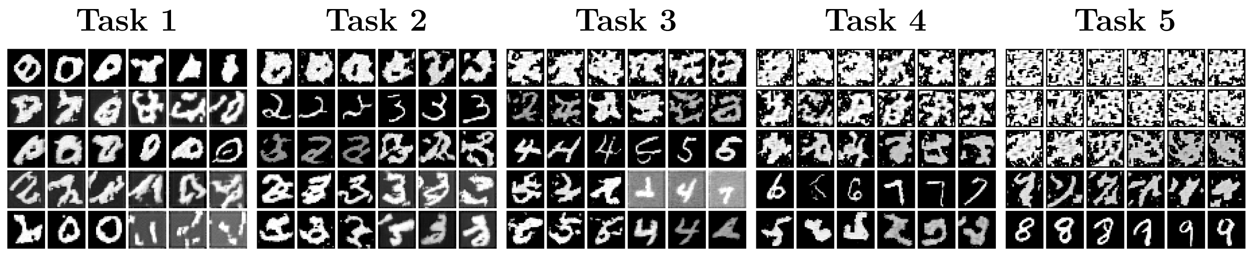


Figure 5.8: **Generative Replay**

In Figure 5.8, a representation of the baseline generative replay methodology is presented. We observe poor performance closely related to the previously obtained metrics. These visual representations provide a qualitative comparison with our methodology. As this implementation closely resembles our method, it is important to get a broader view when evaluating the impact and improvements of the proposed method.

5.1.1.3. Reference Images for New Implementations

The following figures provide reference outputs from our implementations and methodology. We describe the results and offer a qualitative comparison with the baselines. These findings are related to previously obtained quantitative metrics, further validating the promising performance of our methodology.

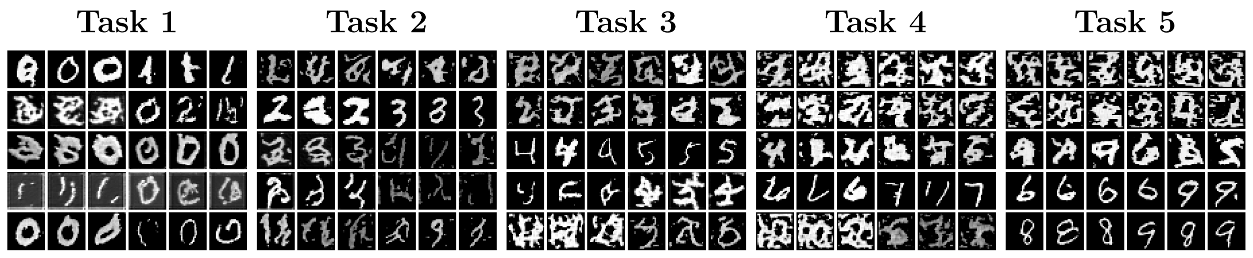


Figure 5.9: **Continual Generative Replay Base Reward**

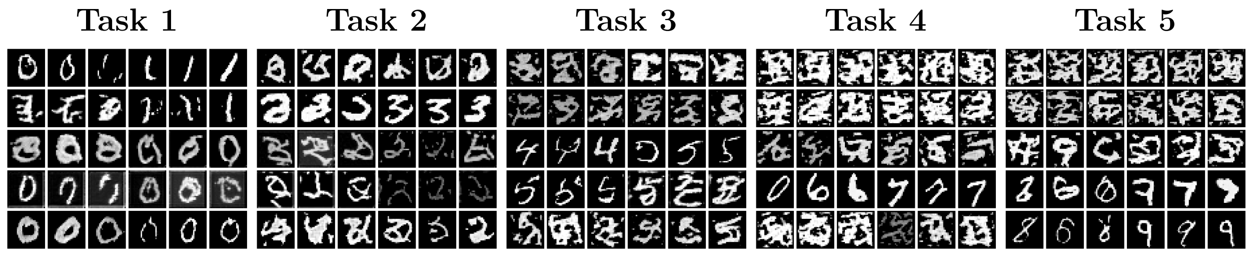


Figure 5.10: **Continual Generative Replay Base Mixed Reward**

From the results in Tables 5.1 and 5.2, further validated by visual inspection, we observe better preservation of previous tasks when training on a new task at each step. Although this preservation is not balanced (for example, in task 2, "zeros" are better preserved than "ones"), there is a clear improvement in structure preservation. Later stages exhibit more structured outputs rather than pure noise, as seen in Figure 5.8.

This indicates that the model aligns better with the preservation task and that the Reinforcement Learning process is identifying and favoring structures that closely resemble numbers. Although it's still far from perfect, this provides a better understanding of the methodology capabilities.

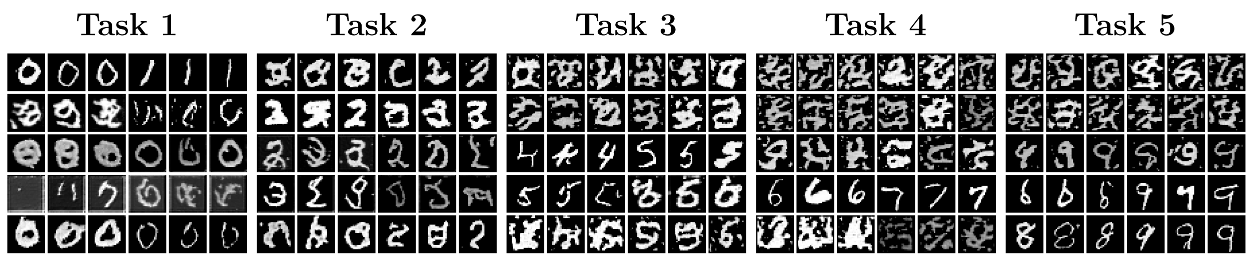


Figure 5.11: **Continual Generative Replay Teacher-Student Reward**

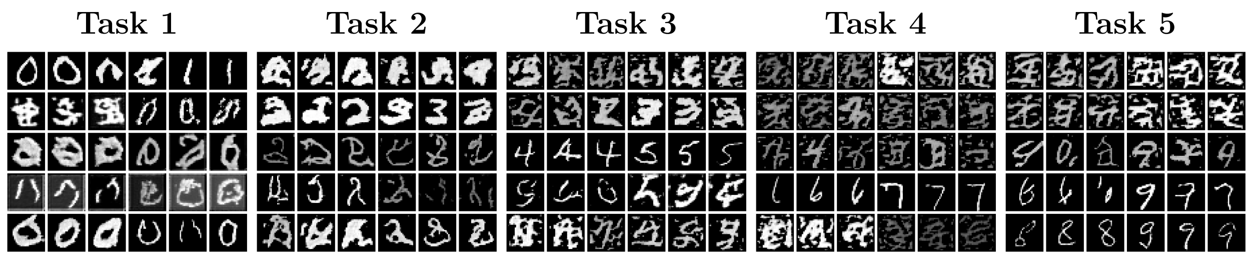


Figure 5.12: **Continual Generative Replay Teacher-Student Mixed Reward**

Figure 5.11 corresponds to the best implementation we achieved in terms of metrics and overall performance, and our best attempt on bridging the gap between Experience Replay and Generative Replay. Although not all classes perform best (for example, in task two, we see difficulties in learning the classes "two" and "three"), the overall performance is the best in contrast with other Generative Replay implementations. By the end of task five, we obtain reasonable results for the classes "six" and even "four" and "five." Additionally, we observe some structural preservation in the previous classes.

Figure 5.12 performs similarly, enhancing the results previously obtained with the base methodology and providing further validation of the proposed Teacher-Student training methodology. Overall this aproach correspond to our best work at enhancing the continual learning capabilities in Diffusion Models using Generative Replay.

5.1.2. Results for Model Trained on FashionMNIST

The following metrics and results come from the implementation of the mehtodology in a more complex scenario, where our reward model is not very strong, as the classifier suffers from less from ideal performance. We will show evidence of the pitfalls of our methodology and where is ill suited to be applied in continual learning. Specifically in the case that the reward model is sub optimal and doesnt yield informative rewards.

5.1.2.1. Metrics for models trained on FashionMNIST

Table 5.3: FashionMNIST Data FID Score Evolution

Task Average Value	Finetuning	Gen Replay	Exp Replay	CL-RL		CL-RL-ts	
				Basic	Mixed	Basic	Mixed
Task 1	267.53	259.46	259.52	241.66	247.45	259.46	257.54
Task 2	301.31	285.79	278.36	292.75	288.88	293.70	289.58
Task 3	254.95	259.11	269.76	294.91	293.31	293.77	287.98
Task 4	261.18	275.02	257.37	317.41	298.98	312.40	289.57
Task 5	265.75	275.61	**261.37**	298.48	**295.27**	296.45	298.82
Average	270.14	270.99	265.27	289.04	284.77	291.156	284.69

The table 5.3 gives us a good representation of the pitfalls of using a subpar reward model, but doesn't tell the whole story. We clearly see that all baseline implementations have better results than the implementation using Reinforcment Learning. We find that the best performance comes from **Experience Replay**, surprisingly the second best performance comes from **Finetuning**. We later see that this doesn´t always correlate with the best preservation of generative capabilities. Also signaling the limitations of the metrics we are using for evaluation.

Table 5.4: FashionMNIST Data Accuracy Evolution

Task Average Value	Finetuning	Gen Replay	Exp Replay	CL-RL		CL-RL-ts	
				Basic	Mixed	Basic	Mixed
Task 1	0.878	0.811	0.838	0.845	0.871	0.808	0.786
Task 2	0.367	0.600	0.725	0.346	0.401	0.409	0.410
Task 3	0.303	0.508	0.633	0.366	0.386	0.401	0.370
Task 4	0.206	0.260	0.567	0.314	0.232	0.286	0.273
Task 5	0.178	0.219	**0.629**	**0.267**	0.258	0.241	0.215
Average	0.386	0.479	0.678	0.427	0.429	0.429	0.410

The table 5.4 provides results that offer a more comprehensive picture. We see that the reward model, which we are using for evaluation, aligns the diffusion model with outputs that we identify as belonging to each specific class. The fact that the accuracy of the classifier we use as a reward does not represent a near perfect preference heavily disrupts the performance of the model. This is reflected in the results of both tables, where clearly the FID score is worse than the base Generative Replay implementation, but the final accuracy results are better or comparable to it. This strongly indicates that we are optimizing for performance over the reward, and that this performance does not align with the diffusion model generative capabilities.

The following figures illustrate the temporal evolution of the obtained metrics, providing a visual representation of behavior across tasks. We will further discuss the results in the next chapter, offering a more complete and comprehensive analysis.

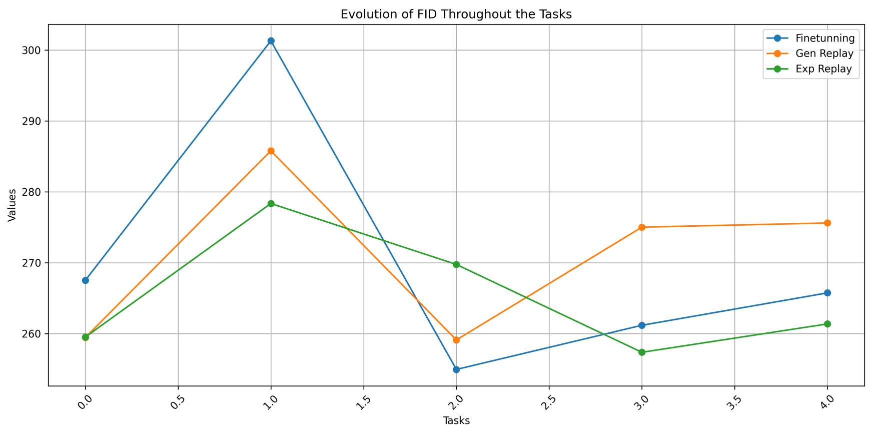


Figure 5.13: FID Evolution of Baselines: Evolution of baselines over the FashionMNIST dataset

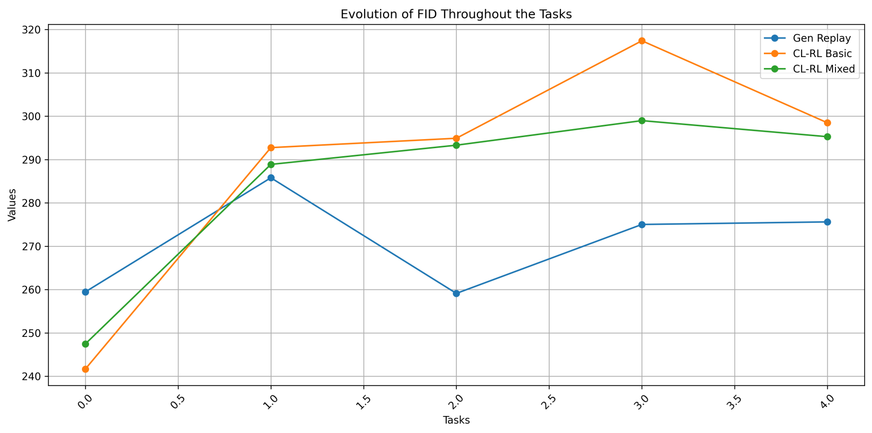


Figure 5.14: FID Evolution Base Methodology: Evolution for both reward configurations

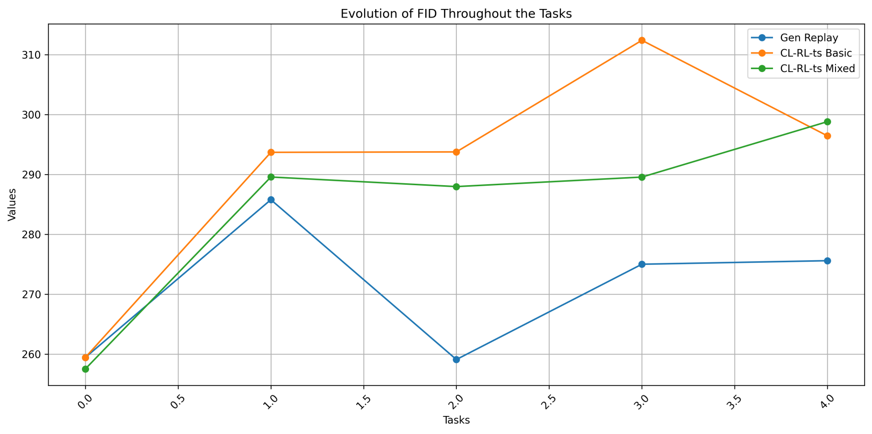


Figure 5.15: FID Evolution Teacher-Student Methodology: Evolution for both reward configurations

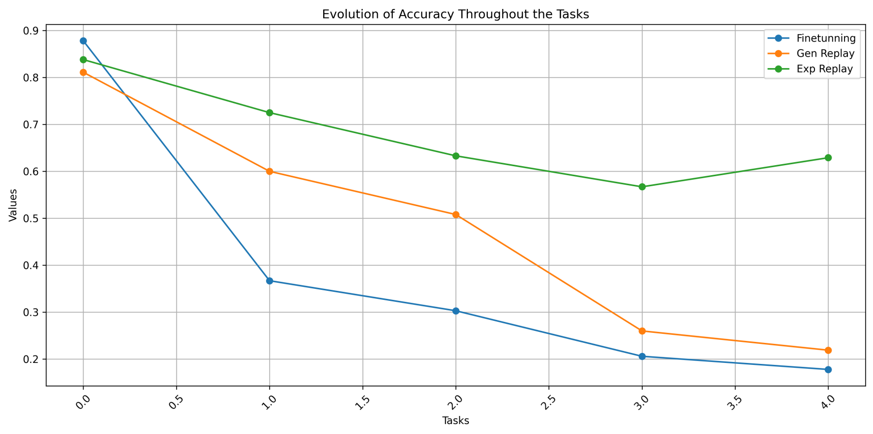


Figure 5.16: Accuracy Evolution of Baselines: Evolution of baselines over the FashionMNIST dataset

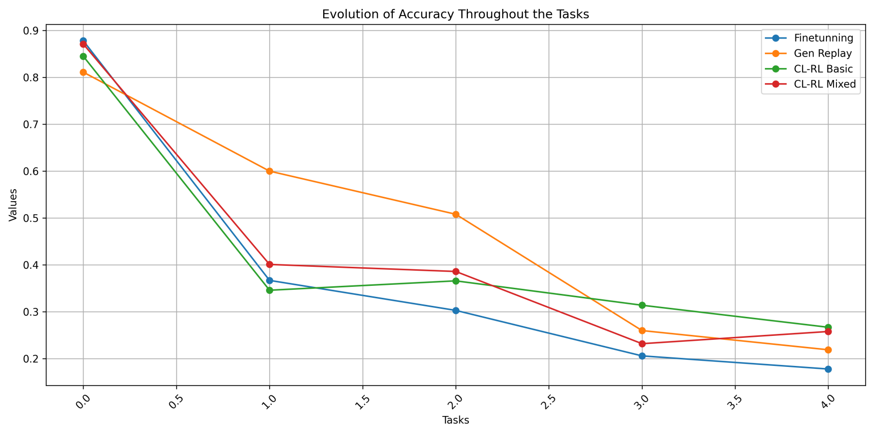


Figure 5.17: Accuracy Evolution Base Methodology: Evolution for both reward configurations

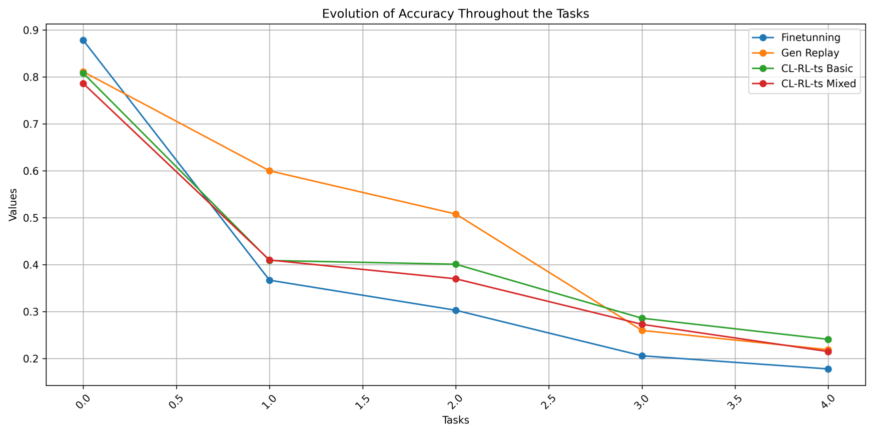


Figure 5.18: Accuracy Evolution Teacher-Student Methodology: Evolution for both reward configurations

5.1.2.2. Reference Images for Baselines

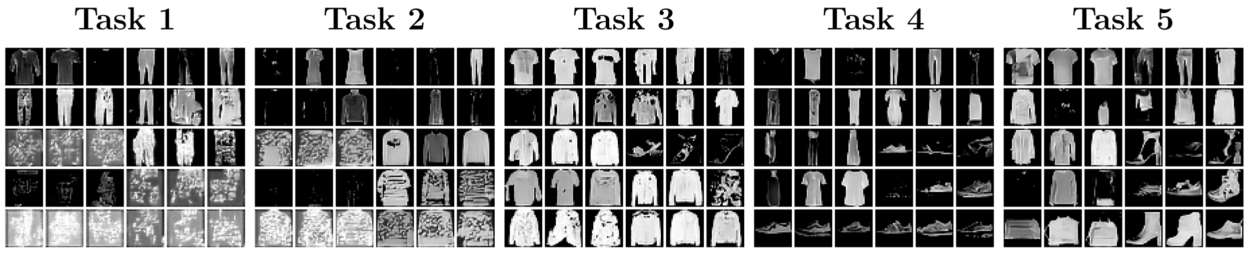


Figure 5.19: **Experience Replay**

In Figure 5.19, we observe reasonable class preservation across tasks, with improvements in class representation in the later tasks for classes that were learned at the beginning. This is related to the fact that later classes have great similarities with early classes, for example, class "T-shirt" (0) and class "Coat" (4). This behavior corresponds to the phenomenon called Backward Transfer, which we previously explained in the **Background and Theory** chapter.

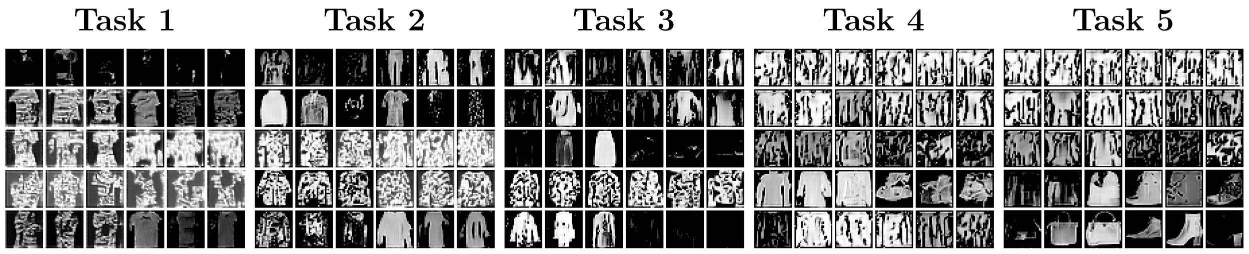


Figure 5.20: **Generative Replay**

Figure 5.20 shows the generative capabilities of the **Generative Replay** approach. Similar to the previous implementation with the MNIST dataset, we see poor class retention as we train on subsequent tasks. Although some structure from previous classes is retained, there is no instance where the classes are clearly defined beyond the task they are trained on. This is coherent with the obtained metrics that we have previously shown.

5.1.2.3. Reference Images for New Implementations

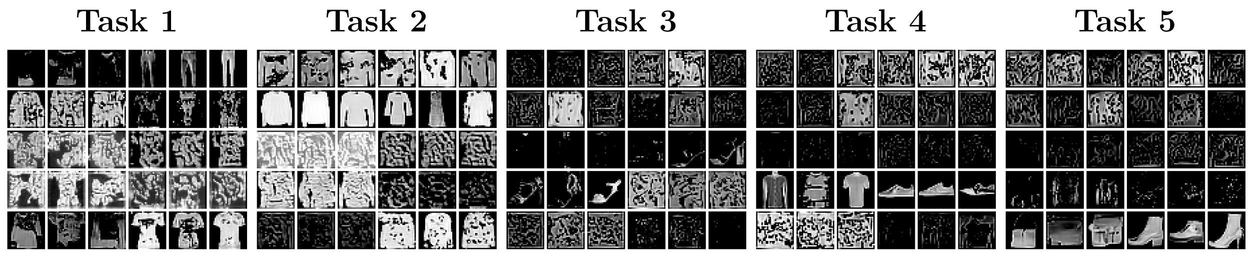


Figure 5.21: **Continual Generative Replay Base Reward**

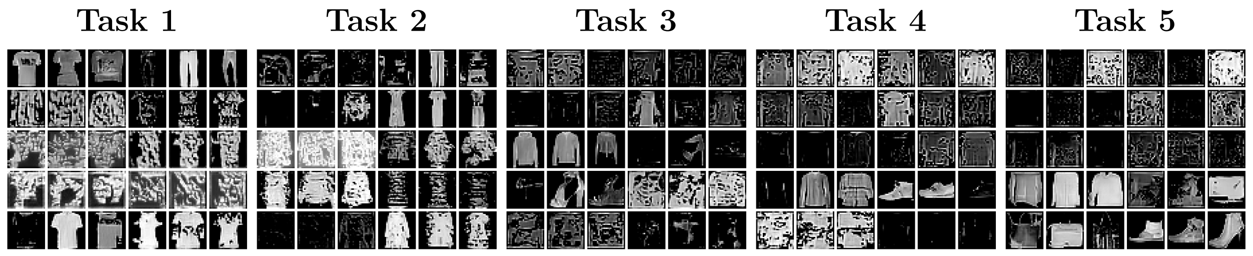


Figure 5.22: **Continual Generative Replay Base Mixed Reward**

Figures 5.21 and 5.22 highlight the issue with our implementation and the use of a subpar preference model as a reward. In this case, the classifier aligns the diffusion model with a downstream objective that is not always aligned with the class we are trying to represent, making it prone to misaligning the generative diffusion model. We see almost no structure preservation (generated images are visually much noisier), and there is no retention of previous knowledge across tasks.

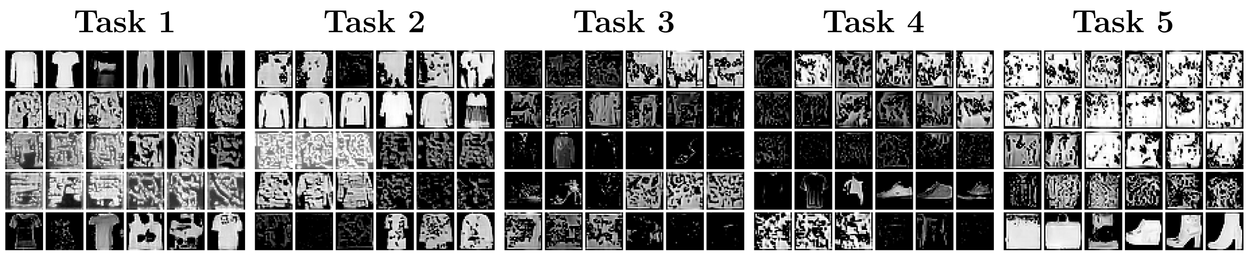


Figure 5.23: **Continual Generative Replay Teacher-Student Reward**

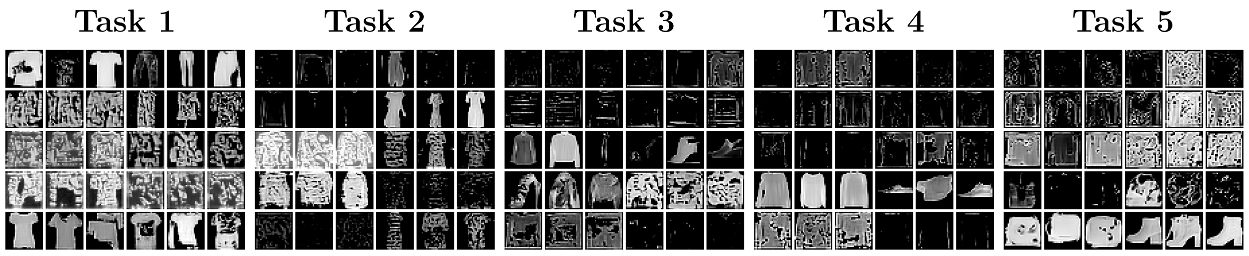


Figure 5.24: **Continual Generative Replay Teacher-Student Mixed Reward**

From visually inspecting the generations in both scenarios of continual learning with the teacher-student methodology (Figures 5.23 and 5.24), we observe the same drop in performance and undesired behavior as seen previously. There is almost no structure preservation and no previous class retention as we progress with training on new tasks.

The use of the teacher-student methodology allows us to prevent overparametrization and mode collapse during training. However, if the reward model yields poor and unrepresentative results, the data used for retraining will inevitably affect the diffusion model negatively, creating a mismatch between the original target distribution and the optimized one.

Chapter 6

Discussion

In this chapter, we will discuss the behavior of the different continual learning scenarios that have been evaluated. We will first review each experiment and highlight the key takeaways, detailing the important conclusions and providing a comparative analysis between them. Then, we will summarize all of our findings, detailing the key takeaways from the experimental process, the proposed methodology, main challenges, and potential improvements.

6.1. Results Overview for MNIST

An overview of the results obtained for MNIST is provided in this section. This includes a discussion of the different implementations, both qualitative and quantitative observations, and a comparison between methodologies.

6.1.1. Baseline Implementations

Experimentally, we confirmed that the best performance, and still the state of the art in replay methods, corresponds to **Experience Replay**. Both **Finetuning** and **Generative Replay** fall short in terms of output quality and performance across the board. Let us inspect them one by one.

The basic approach of **Finetuning** is an initial approximation in an attempt to continually learn new concepts and tasks, but it rapidly falls short in terms of performance. It shows almost no class retention from previous tasks and degrades rapidly as we continually train the model (see Figure 5.1). This behavior is expected, as we are reparametrizing the diffusion model without penalizing forgetting in any way, which inevitably leads to subpar results in continually learning new classes. This provides a good reference for why replay methodologies are so attractive and what kind of performance we can expect when there is no learning from previous tasks during training on new tasks.

The results from **Generative Replay** are objectively better than those from **Finetuning**. There is clear retention of previous classes and tasks during training, but as we move from one task to another, we see a progressive downgrade in the performance on the older classes (more detailed results can be found in Annex D). When visually inspecting the generative performance of our diffusion model, we find that there is some structure preservation of the previous classes when training on new tasks. However, this is highly dependent on how similar the new classes are to the previously learned ones. Most of the images generated from previous tasks show noisier samples with almost no discernible pattern of previous classes. This results serve as the baseline for replay methods and are key to validating the results of our custom implementation, that is based on this methodology.

Finally, **Experience Replay** is the state of the art in terms of performance on the FID score evaluation and accuracy, exceeding by far the results obtained with **Generative Replay**. We observe signs of backward transfer during training (see Figure 2.5 for reference). This behavior is interesting because it suggests that the order in which the different classes are learned is important. In this particular case, the class "seven" is improved upon when we learn the class "nine". We will see that in our proposed methodology, this behavior appears again, indicating that the order in which we learn the classes might be as important as trying to preserve old classes via replay.

The qualitative results further validate **Experience Replay** as the goal we are aiming to achieve. There is visual class retention across the board, and we still get informative and clear images of the initially learned classes in the later stages of training on new tasks.

6.1.2. Custom Implementations

The use of the proposed methodology, leveraging Reinforcement Learning to refine the sample quality of the diffusion model in replay scenarios, yields positive results. The best performing experiment, utilizing **Teacher-Student CL-RL with a Basic Reward**, obtains the best results in the Generative Replay based methodologies, all implementations are significantly better than **Generative Replay** across the board. In terms of metrics, both the FID score and accuracy are consistent, strongly indicating that the model is exhibiting previous class retention behaviors.

Using Reinforcement Learning here enhances two properties of the diffusion model simultaneously. Firstly, it improves the sample quality of the task we have just trained on, giving us control over the type of samples we aim to generate via a reward. Secondly, it realigns the model on previously seen classes. However, the main restriction is that the model already needs to be capable of generating samples from these classes to be further optimized, as the algorithm is a preference enhancement algorithm. so as we progress training and the generative capabilities deteriorate, we would expect the performance of the algorithm to deteriorate too, wich is indeed the case.

As we previously observed in the **Experience Replay** scenario, there are signs of backward transfer during training. This is a desired characteristic and further validates the fact that using Reinforcement Learning in these scenarios is an interesting approach to improve the performance of generative models in continual learning settings.

Through visual inspection of the generated samples, we see that all implementations using our methodology clearly outperform the baseline with **Generative Replay**. We observe strong previous class retention, in line with the obtained metrics, and much more structure preservation of the old classes in later stages of training on new tasks.

6.2. Results Overview for FashionMNIST

This section provides an interpretation of the results obtained from training a Diffusion Model on the FashionMNIST dataset. We dive deeper into the results and the complexities of the implementation that gave us this outcomes, highlighting the pitfalls of our proposed methodology along with a comparative assessment of the qualities of the trained models.

6.2.1. Baseline Implementations

The baseline implementations provide us with significant information on the expected behavior of our diffusion models. Similar to the experiments conducted on the MNIST dataset, **Experience Replay** is the most performant. The FID score and accuracy greatly favor this methodology, with **Generative Replay** following close behind. In this particular case, the FID score indicates performance but doesn't paint the full picture. With higher values, it is difficult to determine whether the model performance are comparable with one another. For example, in terms of accuracy, **Generative Replay** clearly outperforms **Finetuning**, but for the FID score, it slightly favors **Finetuning**. The main reason is that the distribution of the images is quite complex for this scenario, unlike numbers, so these metrics suffer from imprecision when evaluating. This pertains its own challenges, so we have to levearge not just the metrics but also a visual evaluation.

Visually, we can get a clearer picture. We see that **Experience Replay** performs quite well, showing some level of backward transfer, which is validated both by the resulting images in Figure 5.19 and Table 5.3. The evolution of the replay methodologies is far more stable than finetuning (see Figures 5.13 and 5.16 for reference). This shows that although learning subsequent tasks is hindered by the complexity of the dataset, there are still advantages in stability and retention of previous knowledge using replay strategies.

6.2.2. Custom Implementations

The performance of our proposed methodology, leveraging Reinforcement Learning, is worse across the board both in the qualitative results and in the visual inspection and analysis. The mean accuracy and FID score are slightly better than **Finetuning** during training, and they barely manifest any of the benefits of replay methods. The only instance where the performance is better than **Generative Replay** is in the later stages of training, which correlates with the fact that the classifier is more performant on the last few classes (see the confusion matrix for reference in Figure 4.3).

The slight improvement in performance in the later stages using our implementation further highlights the importance of a good reward model. By using a reward that is less than ideal, we effectively erase all the benefits of the replay methodologies, rendering the method useless. It even erases the prevailing structures of the dataset that are otherwise sustained in **Generative Replay**. This means that the diffusion model is worsened across the board with data that is contaminated and hinders the generative capabilities of both seen and unseen classes. Previously learning some classes in advance parametrizes the model to generate some structures that is lost when leveraging a sub par reward.

This can be seen in the metrics performance and evolution (see Figures 5.17 and 5.18), where there exists some level of forward transfer (Figure 2.4) in subsequent tasks structures compared to training on noisier images that do not yield information and are classified incorrectly.

6.3. Implications of the Findings

There are several key findings that are relevant in the context of continual learning with diffusion models, especially considering our implementation. There are great advantages in using **Experience Replay**, as we have clearly shown through our experiments and validated with a comprehensive analysis. The use of a generative model to leverage the advantages of replaying past data emerges as a promising approach. The fact that we are using it to retrain the generative model itself is even more appealing, as there is a lower computational cost associated with it, and it could pave the way for improving generative models without accessing the entire corpus of data needed to train it from scratch.

With this in mind, exploring new alternatives to improve the performance of generative models, especially in a continual learning scenario, is an important area of work. The proposed methodology builds upon these necessities and shows promising results. The main advantages are a clear performance boost in its continual learning capabilities, especially with the **Teacher-Student CL-RL** implementations, leveraging a form of reinforcement learning that improves performance when executed correctly.

The main challenges of the proposed methodology are also linked to the use of Reinforcement Learning. It requires that the generative model is capable of generating samples that still correspond to the class of the data of interest on some level. Additionally, the reward function used plays a key role, in this case, the use of a classifier with nearly perfect performance is vital, as implementing a subpar classifier yields catastrophic results.

To further improve the results, the design of well calibrated rewards is crucial. We experimented with a very basic but powerful form of reward, leveraging a model that classifies the produced samples individually. This could be further studied and improved on by trying rewards that maximize a teacher model to produce better samples, not only one by one, but considering the whole batch of images generated. The main strength of the methodology is generating better individual samples, but it does not generate sufficiently diverse samples, which inherently leads to mode collapse and reparametrizations that negatively affect the generative model.

Chapter 7

Conclusion

In this thesis, we investigated different approaches to continual learning applied to diffusion models, with a particular focus on replay methods. We conducted a comprehensive study evaluating the generative capabilities of the main known approaches. After establishing these benchmarks, we developed and implemented two novel approaches leveraging a form of Reinforcement Learning called DDPO, to later assess their effectiveness compared to existing methods.

We successfully conducted experiments replicating existing methodologies, providing clear and well evaluated benchmarks to compare our implementations. This allowed us to establish an evaluation methodology and set clear objectives for our research moving forward.

We implemented a novel methodology that took advantage of reinforcement learning to improve the sample quality of generative models used for replay. Further improvements were made by designing a teacher-student training scheme, which separated the tasks of each training step into more focused objectives for each model. This implementation showed considerable improvement.

We explored the use of two reward functions that leverage a classifier as the main driving force. This type of reward provided key insights into our methodologies and highlighted the importance of well calibrated reward functions, which are crucial for the success of the methodology and defining its limitations in this regime.

Our research contributes to the advancement of generative models, extending their applicability and efficacy in settings that require continual learning. The integration of reinforcement learning with generative replay offers a promising direction for future research and development, indicating a field that requires further study.

While our study provides significant insights, there are clear limitations that need to be addressed, such as the need for a reward model with perfect preferences and knowledge of the class of interest. Future work should explore more sophisticated reward functions and alternative reinforcement learning strategies to enhance the performance and diversity of generative models.

In conclusion, our research addresses a critical challenge in continual learning for generative diffusion models. By leveraging reinforcement learning, we have developed methodologies that enhance the quality of generated samples used for replay methods. This work opens new avenues for research and shows a promising path for further improving generative replay approaches.

Bibliography

[1] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, Curran Associates, Inc., 2020.

[2] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *International Conference on Machine Learning*, pp. 2256–2265, 2015.

[3] M. Zając, K. Deja, A. Kuzina, J. M. Tomczak, T. Trzcinski, F. Shkurti, and P. Miłos, “Exploring continual learning of diffusion models,” *arXiv preprint arXiv:2303.15342*, Mar 2023. Available online at https://arxiv.org/abs/2303.15342.

[4] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, “Embracing change: Continual learning in deep neural networks,” *Trends in Cognitive Sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[5] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[6] T. Lesort, M. Caccia, and I. Rish, “Understanding continual learning settings with data distribution drift analysis,” 2021.

[7] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, “Online continual learning in image classification: An empirical survey,” *Neurocomputing*, vol. 469, pp. 28–51, 2022.

[8] L. Wang, X. Zhang, H. Su, and J. Zhu, “A comprehensive survey of continual learning: Theory, method and application,” 2023.

[9] S. Thrun, “Lifelong learning algorithms,” in *Learning to Learn* (S. Thrun and L. Pratt, eds.), pp. 181–209, Kluwer, 1998.

[10] Z. Chen and B. Liu, *Lifelong Machine Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2018.

[11] K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics.* MIT Press, 2023.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, second ed., 2018.

[13] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, “Training diffusion models with reinforcement learning,” 2024.

[14] D. Dowson and B. Landau, “The fréchet distance between multivariate normal distributions,” *Journal of Multivariate Analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained

by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[16] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," 2017.

[17] S. Booth, Y. Zhou, A. Shah, and J. Shah, "Bayes-probe: Distribution-guided sampling for prediction level sets," *Reference of the journal or conference*, vol. Volume number, no. Issue number, p. Page range, 2020.

ANNEXES

Annex A. Detailed Architecture of ClassConditionedUnet

```
class ClassConditionedUnet(nn.Module):
    def __init__(self, num_classes=10, class_emb_size=8):
        super().__init__()

        # Increase the embedding size for more rich class representations
        self.class_emb = nn.Embedding(num_classes, class_emb_size)

        # Enhance the UNet model with increased depth and wider layers
        self.model = UNet2DModel(
            sample_size=28,  # The target image resolution
        in_channels=1 + class_emb_size,  # Additional input channels for class conditioning
            out_channels=1,  # The number of output channels
        layers_per_block=3,  # Increased number of ResNet layers per UNet block for more depth
            block_out_channels=(64, 128, 256),  # Wider layers for increased capacity
            down_block_types=(
                "DownBlock2D",
                "AttnDownBlock2D",  # Retain spatial self-attention in downsampling
                "AttnDownBlock2D",
            ),
            up_block_types=(
                "AttnUpBlock2D",  # Retain spatial self-attention in upsampling
                "AttnUpBlock2D",
                "UpBlock2D",
            ),
        )

    def forward(self, x, t, class_labels):
        bs, ch, w, h = x.shape

        # Map class labels to embeddings and reshape for concatenation
        class_cond = self.class_emb(class_labels)
      class_cond = class_cond.view(bs, class_cond.shape[1], 1, 1).expand(bs, class_cond.shape[1],

        # Concatenate input with class conditioning
```

```
net_input = torch.cat((x, class_cond), 1)  # Shape: (bs, 1 + class_emb_size, 28, 28)

    # Pass through UNet and return prediction
    return self.model(net_input, t).sample
```

Annex B. Study over Stabilization of DDPO

B.1. FID Score Variance Across Sample Size

To assess the resampling impact on our diffusion model's generative stability, a structured experimental procedure was implemented, as outlined in Algorithm 1. The model, trained on digits 0 and 1 from the MNIST dataset, was used to generate a reference set of 300 images. Iterative FID score computations over sample sizes [2, 5, 20, 50, 200, 500] were conducted. These scores allowed us to measure the variance, thereby evaluating the model's stability across different sampling conditions.

Algorithm 2 Sample Generation and FID Calculation

```
Require: model, digits, num_iterations, sample_sizes, ref_dataset_size
Ensure: FID_scores
1: Generate ref_dataset_size images for digits using model
2: for each size in sample_sizes do
3:     for i from 1 to num_iterations do
4:         Sample images of current size from model
5:         Calculate and record FID score compared to reference images
6:     end for
7:     Compute and add variance of recorded FID scores to FID_scores
8: end for
9: return FID_scores
```

The experimental results, depicted in the following figures, illustrate the variance in FID scores for the digits 0 and 1 across a range of sample sizes. These box plots reveal a trends in the models image generation capabilities and its self consistency:

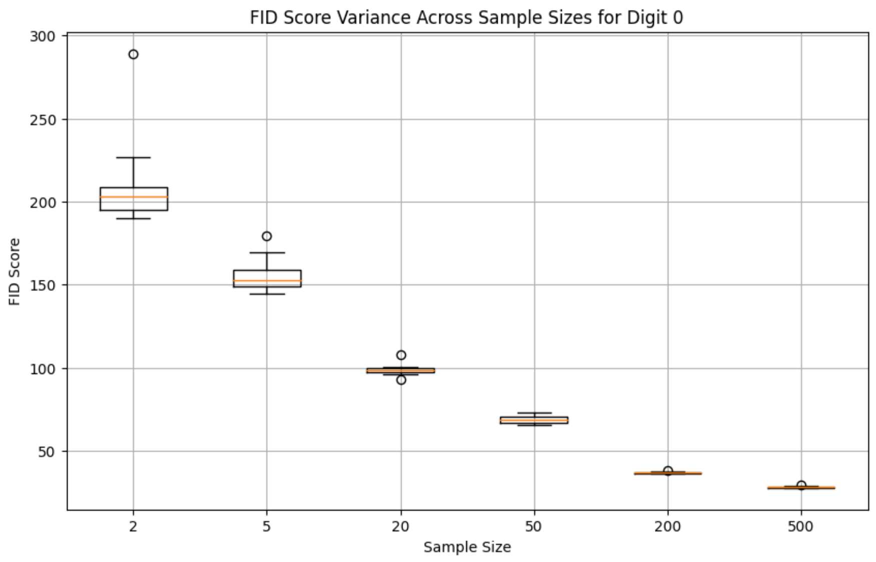


Figure B.1: FID Score Variance Across Different Sample Sizes Label 0

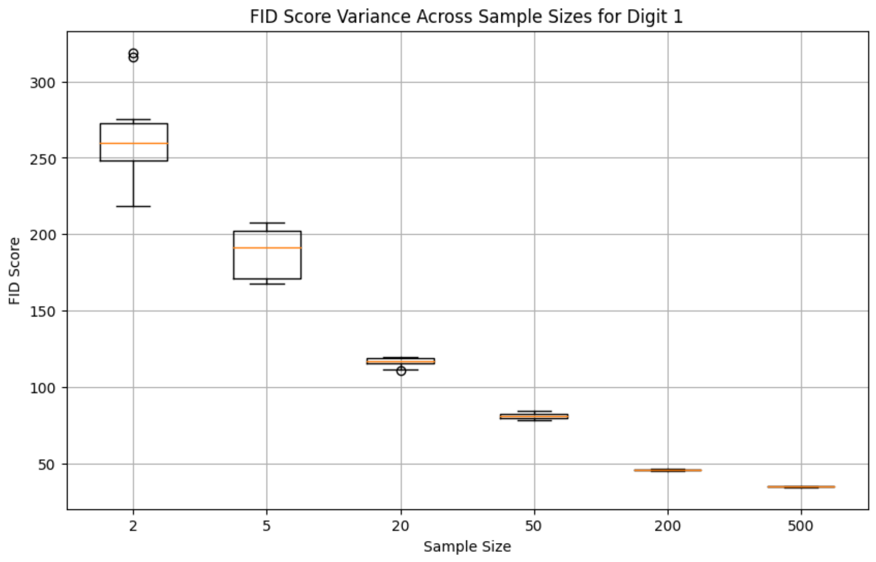


Figure B.2: FID Score Variance Across Different Sample Sizes Label 1

As the sample size increases, the variance in FID scores tends to decrease, suggesting enhanced stability in image generation from the model.

B.2. Evolution of Learning in Exploration

In the exploration phase of DDPO, the model iteratively learns to navigate the generative space. With each epoch, it generates new images, refining its ability to produce high-quality outputs.

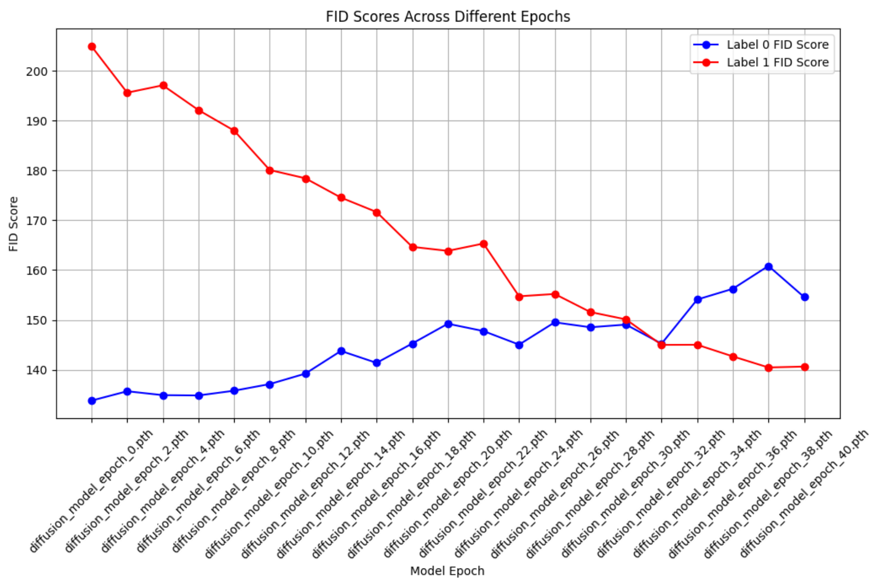


Figure B.3: FID Score Evolution during the Exploration Phase for Labels 0 and 1

Figure B.3 illustrates the FID scores for labels 0 and 1. While label 1 sees an improvement in image quality over epochs, indicated by decreasing FID scores, label 0 exhibits an increase in FID scores, suggesting a need for further model refinement.

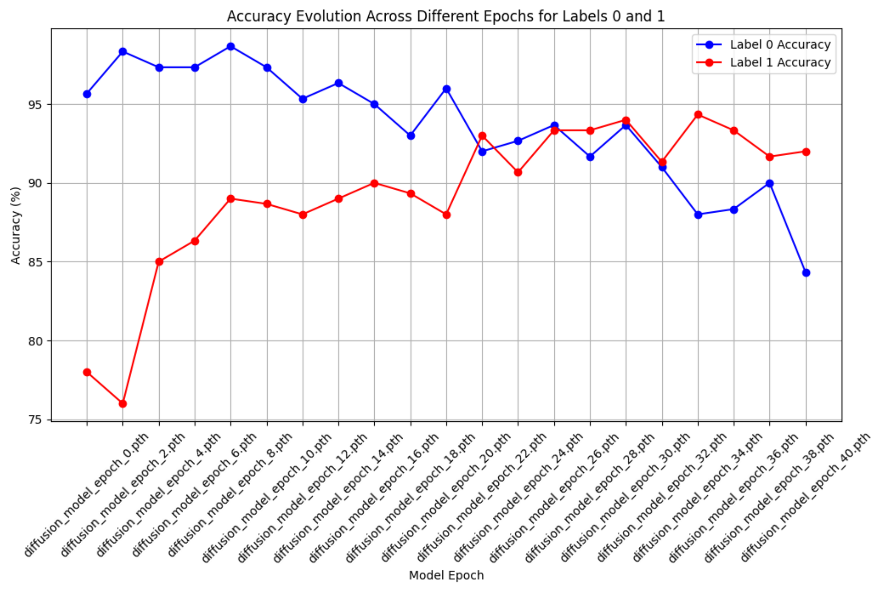


Figure B.4: Accuracy Evolution during the Exploration Phase for Labels 0 and 1

In Figure B.4, we observe that the model's accuracy for label 1 stabilizes and ascends post-initial fluctuations, reflecting successful learning. Conversely, the accuracy for label 0 decreases, which corresponds with the FID score trends and highlights disparate learning trajectories within the model.

B.3. Evolution of Learning in Exploitation

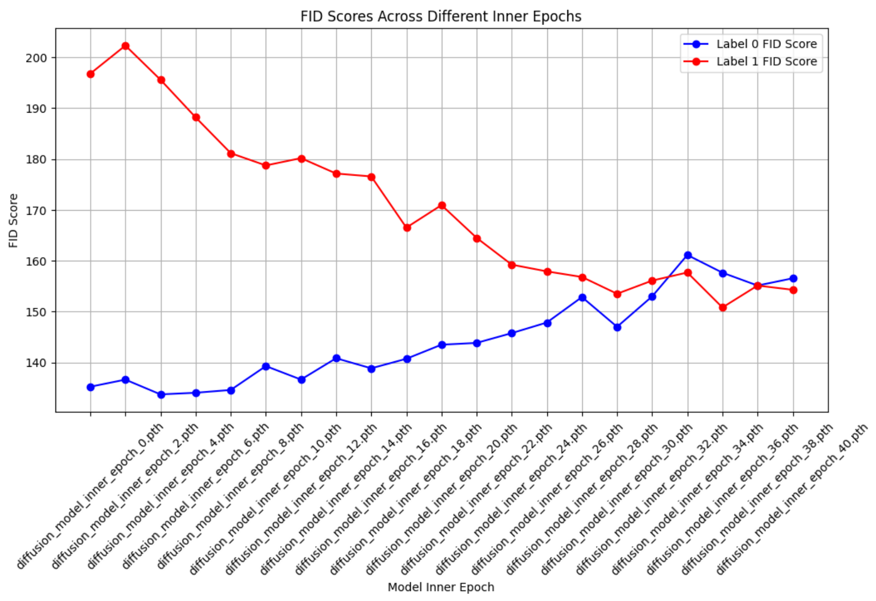


Figure B.5: FID Evolution of Learning in Exploitation Phase for Labels 0 and 1

The optimization phase is similar to the exploration phase but exhibits a much more stable behavior. This suggests that we deviate more slowly from the original data distribution when using the same data for optimization. Top performance is achieved at the 10th training step.

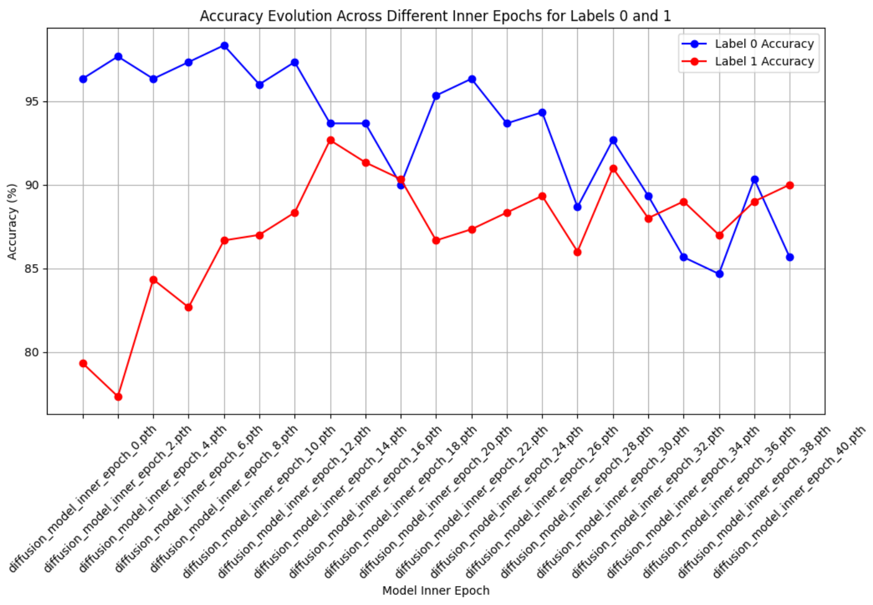


Figure B.6: Accuracy Evolution of Learning in Exploitation Phase for Labels 0 and 1

The same holds fo the Accuracy evolution, indicating the top performance at the 10th training step aproximatley.

Annex C. FashionMNISTWithArtifacts Class Implementation

The *FashionMNISTWithArtifacts* class is an extension of the standard FashionMNIST dataset used to introduce artificially corrupted images into the training process. This class helps simulate the effects of image degradation that might occur in practical applications, providing a robust testing scenario for the classifier. Below is the Python code used to implement this dataset extension.

Code C.1: FashionMNISTWithArtifacts Class Implementation

```python
import torch
from torchvision import datasets, transforms
from torchvision.datasets import VisionDataset
from torch.utils.data import DataLoader
from PIL import Image
from io import BytesIO
import numpy as np

class FashionMNISTWithArtifacts(VisionDataset):
    def __init__(self, root, train=True, transform=None, target_transform=None, download=False, num_artifacts=1000, quality=6):
        super().__init__(root, transform=transform, target_transform=target_transform, download=download)
        self.dataset = datasets.FashionMNIST(root=root, train=train, transform=None, download=download)
        self.num_artifacts = num_artifacts
        self.quality = quality
        self.artifact_indices = np.random.choice(len(self.dataset), self.num_artifacts, replace=False)

    def rotate_compress_and_rotate_back(self, img, angle, quality):
        rotated_img = img.rotate(angle)
        img_io = BytesIO()
        rotated_img.save(img_io, format='JPEG', quality=quality)
        img_io.seek(0)
        compressed_img = Image.open(img_io)
        return compressed_img.rotate(-angle)

    def __getitem__(self, idx):
        if idx >= len(self.dataset):
            original_idx = self.artifact_indices[idx - len(self.dataset)]
            img, _ = self.dataset[original_idx]
            img = transforms.ToPILImage()(img)
            angle = np.random.randint(360)
            img = self.rotate_compress_and_rotate_back(img, angle, self.quality)
            label = 10
        else:
            img, label = self.dataset[idx]
            img = transforms.ToPILImage()(img)

        if self.transform:
            img = self.transform(img)
        return img, label

    def __len__(self):
        return len(self.dataset) + self.num_artifacts

# Usage example
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
train_dataset = FashionMNISTWithArtifacts(root='./data', train=True, transform=
     ↪ transform, download=True, num_artifacts=6000, quality=6)
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
```

Annex D. Detailed Metrics and Results

D.1. Fine-tuning

Table D.1: MNIST data FID Score Evolution Evaluated on Each Task

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	180.89	292.02	317.01	321.20	340.46
Task 2	-	157.95	274.32	320.99	329.32
Task 3	-	-	133.50	282.02	337.04
Task 4	-	-	-	145.80	294.86
Task 5	-	-	-	-	144.73
Average	180.89	224.98	241.61	267.50	289.28

Table D.2: FashionMNIST data FID Score Evolution Evaluated on Each Task

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	267.53	297.34	293.31	277.07	288.71
Task 2	-	305.29	234.47	232.72	254.52
Task 3	-	-	237.07	276.42	275.09
Task 4	-	-	-	258.52	270.22
Task 5	-	-	-	-	240.23
Average	267.53	301.31	254.95	261.18	265.75

Table D.3: MNIST Data Accuracy Evolution Evaluated on Each Task

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	0.845	0.016	0.038	0.076	0.010
Task 2	-	0.743	0.006	0.038	0.003
Task 3	-	-	0.916	0.011	0.021
Task 4	-	-	-	0.940	0.015
Task 5	-	-	-	-	0.885
Average	0.845	0.380	0.320	0.266	0.187

Table D.4: FashionMNIST Data Accuracy Evolution Evaluated on Each Task

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	0.878	0.016	0.005	0.020	0.039
Task 2	-	0.718	0.053	0.011	0.015
Task 3	-	-	0.853	0.133	0.001
Task 4	-	-	-	0.661	0.008
Task 5	-	-	-	-	0.825
Average	0.878	0.367	0.303	0.206	0.178

D.2. Experience Replay

Table D.5: MNIST Data FID Score Evolution Evaluated on Each Task (Experience Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	179.86	170.05	162.07	154.44	162.03
Task 2	-	136.82	125.97	143.41	137.01
Task 3	-	-	132.54	133.44	131.15
Task 4	-	-	-	135.73	140.25
Task 5	-	-	-	-	134.10
Average	179.86	153.43	140.19	141.75	140.90

Table D.6: FashionMNIST Data FID Score Evolution Evaluated on Each Task (Experience Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	259.52	261.61	269.81	241.10	272.77
Task 2	-	295.11	297.65	280.07	297.87
Task 3	-	-	241.81	243.01	240.25
Task 4	-	-	-	265.31	254.76
Task 5	-	-	-	-	241.18
Average	259.52	278.36	269.76	257.37	261.37

Table D.7: MNIST Data Accuracy Evolution Evaluated on Each Task (Experience Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	0.830	0.868	0.578	0.858	0.653
Task 2	-	0.878	0.581	0.731	0.745
Task 3	-	-	0.924	0.798	0.745
Task 4	-	-	-	0.951	0.840
Task 5	-	-	-	-	0.811
Average	0.830	0.873	0.695	0.835	0.759

Table D.8: FashionMNIST Data Accuracy Evolution Evaluated on Each Task (Experience Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	0.838	0.653	0.760	0.696	0.691
Task 2	-	0.798	0.428	0.498	0.546
Task 3	-	-	0.711	0.488	0.561
Task 4	-	-	-	0.588	0.433
Task 5	-	-	-	-	0.911
Average	0.838	0.725	0.633	0.567	0.629

D.3. Generative Replay

Table D.9: MNIST Data FID Score Evolution Evaluated on Each Task (Generative Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	183.76	277.15	310.75	318.10	338.15
Task 2	-	134.42	228.94	265.32	284.42
Task 3	-	-	133.37	246.61	296.31
Task 4	-	-	-	133.25	275.71
Task 5	-	-	-	-	148.18
Average	183.76	205.78	224.35	240.82	268.55

Table D.10: FashionMNIST Data FID Score Evolution Evaluated on Each Task (Generative Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	259.46	280.86	281.39	291.90	318.46
Task 2	-	290.71	264.56	257.40	267.94
Task 3	-	-	231.38	280.66	242.15
Task 4	-	-	-	270.13	287.02
Task 5	-	-	-	-	262.47
Average	259.46	285.79	259.11	275.02	275.61

Table D.11: MNIST Data Accuracy Evolution Evaluated on Each Task (Generative Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	0.795	0.238	0.108	0.056	0.036
Task 2	-	0.855	0.503	0.300	0.278
Task 3	-	-	0.880	0.203	0.068
Task 4	-	-	-	0.880	0.155
Task 5	-	-	-	-	0.881
Average	0.795	0.546	0.497	0.360	0.284

Table D.12: FashionMNIST Data Accuracy Evolution Evaluated on Each Task (Generative Replay)

Evaluated Task	After Task 1	After Task 2	After Task 3	After Task 4	After Task 5
Task 1	0.811	0.443	0.315	0.206	0.198
Task 2	-	0.758	0.330	0.151	0.146
Task 3	-	-	0.880	0.131	0.083
Task 4	-	-	-	0.553	0.168
Task 5	-	-	-	-	0.501
Average	0.811	0.600	0.508	0.260	0.219

D.4. Custom Implementations

D.4.1. CL-RL

Table D.13: MNIST Data FID Score Evolution for CL-RL

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	155.13	157.85	251.08	260.16	290.52	283.34	311.57	308.73	327.37	334.87
Task 2	-	-	142.42	149.30	208.49	201.71	255.15	239.95	295.64	288.32
Task 3	-	-	-	-	136.81	125.38	252.98	243.94	233.55	241.91
Task 4	-	-	-	-	-	-	136.63	149.99	172.49	174.06
Task 5	-	-	-	-	-	-	-	-	166.76	145.01
Average	155.13	157.85	196.75	204.73	211.94	203.47	239.08	235.65	239.16	236.83

Table D.14: FashionMNIST Data FID Score Evolution for CL-RL

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	241.66	247.45	291.42	289.58	331.78	326.39	330.90	315.30	342.06	340.08
Task 2	-	-	294.08	288.17	319.73	307.29	317.31	301.05	303.47	315.82
Task 3	-	-	-	-	233.20	246.26	355.21	308.19	344.01	307.30
Task 4	-	-	-	-	-	-	266.23	271.39	258.77	280.42
Task 5	-	-	-	-	-	-	-	-	244.10	232.73
Average	241.66	247.45	292.75	288.88	294.91	293.31	317.41	298.98	298.48	295.27

Table D.15: MNIST Data Accuracy Evolution for CL-RL

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	0.913	0.898	0.156	0.120	0.066	0.035	0.066	0.030	0.046	0.033
Task 2	-	-	0.758	0.746	0.253	0.375	0.241	0.315	0.216	0.27
Task 3	-	-	-	-	0.896	0.863	0.280	0.278	0.128	0.166
Task 4	-	-	-	-	-	-	0.955	0.976	0.59	0.39
Task 5	-	-	-	-	-	-	-	-	0.855	0.856
Average	0.913	0.898	0.457	0.433	0.405	0.424	0.385	0.399	0.367	0.343

Table D.16: FashionMNIST Data Accuracy Evolution for CL-RL

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	0.845	0.871	0.191	0.163	0.083	0.123	0.150	0.091	0.116	0.111
Task 2	-	-	0.501	0.639	0.135	0.210	0.106	0.131	0.101	0.078
Task 3	-	-	-	-	0.881	0.825	0.176	0.106	0.060	0.075
Task 4	-	-	-	-	-	-	0.823	0.600	0.114	0.108
Task 5	-	-	-	-	-	-	-	-	0.946	0.916
Average	0.845	0.871	0.346	0.401	0.366	0.386	0.314	0.232	0.267	0.258

D.4.2. CL-RL-ts

Table D.17: MNIST Data FID Score Evolution for CL-RL-ts

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	165.60	181.53	255.07	265.37	279.24	281.94	303.18	307.52	319.28	327.08
Task 2	-	-	173.10	144.43	205.56	204.23	237.52	240.59	257.48	257.79
Task 3	-	-	-	-	138.91	125.70	231.92	232.07	201.42	206.57
Task 4	-	-	-	-	-	-	128.46	136.65	166.35	164.53
Task 5	-	-	-	-	-	-	-	-	143.92	142.48
Average	165.60	181.53	214.09	203.40	207.90	203.96	225.27	229.21	217.69	219.69

Table D.18: FashionMNIST Data FID Score Evolution for CL-RL-ts

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	259.46	257.54	293.94	291.25	326.28	325.08	324.67	309.55	315.57	328.80
Task 2	-	-	293.47	287.91	316.18	294.03	304.48	287.8	267.34	323.22
Task 3	-	-	-	-	238.85	244.82	352.03	292.39	300.19	309.33
Task 4	-	-	-	-	-	-	268.41	268.53	339.39	279.08
Task 5	-	-	-	-	-	-	-	-	259.77	253.66
Average	259.46	257.54	293.70	289.58	293.77	287.98	312.40	289.57	296.45	298.82

Table D.19: MNIST Data Accuracy Evolution for CL-RL-ts

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	0.840	0.735	0.166	0.136	0.130	0.056	0.058	0.048	0.055	0.033
Task 2	-	-	0.763	0.803	0.390	0.436	0.238	0.286	0.241	0.211
Task 3	-	-	-	-	0.916	0.868	0.390	0.256	0.148	0.093
Task 4	-	-	-	-	-	-	0.945	0.981	0.570	0.570
Task 5	-	-	-	-	-	-	-	-	0.876	0.781
Average	0.840	0.735	0.464	0.470	0.478	0.453	0.407	0.393	0.378	0.337

Table D.20: FashionMNIST Data Accuracy Evolution for CL-RL-ts

Evaluated Task	After Task 1		After Task 2		After Task 3		After Task 4		After Task 5	
	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed	Basic	Mixed
Task 1	0.808	0.786	0.231	0.135	0.125	0.095	0.159	0.053	0.176	0.055
Task 2	-	-	0.586	0.686	0.195	0.211	0.088	0.163	0.015	0.028
Task 3	-	-	-	-	0.883	0.804	0.131	0.153	0.013	0.036
Task 4	-	-	-	-	-	-	0.765	0.723	0.069	0.046
Task 5	-	-	-	-	-	-	-	-	0.931	0.910
Average	0.808	0.786	0.409	0.410	.0401	0.370	0.286	0.273	0.241	0.215