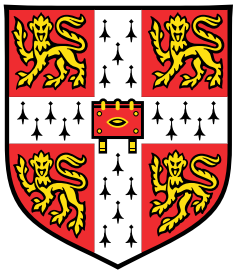
Learning to Forget with Diffusion Hypernetworks



José Miguel Lara Rangel

Supervisor: Prof. David Krueger

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Trinity College August 2024

I dreamed of tears cascading upon the earth,
The sun's rays echoed in the air,
Scorching the backs of men and women,
And feet sowing seeds in the dry soil.

Feathered serpents grew into toys for children,
Playing through fields of sadness,
Suffering was veiled by their illusions.
And we laughed to keep from crying.

Uncertain ways walked with bare feet,
Brimming with faith in the future,
We ventured to sketch a smile,
Following the winding railway of destiny.

Life, remember our steps and tears,
I owe you nothing and you owe me nothing.
Feel your brown soil moistening once more,
As maize springs in the fields of your heart.

Life, embrace me like a Flower,
My eyes see beauty, even among thorns,
Lift that cup filled with the tears of your harvest.
Toast with me to all the moments,
You longed to cradle in your memories,
And to those you begged the heaven to forget.

Life, face up when recalling your path.
Blessed are you among all women,
May God bless the fruit of your spirit and suffering.
Yes, little girl, this is what the wise call living—
And you my Flower, you have danced with life itself.

- I love you Mom

Declaration

I, José Miguel Lara Rangel of Trinity College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains in the main text fewer than 15,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 40 figures.

José Miguel Lara Rangel
August 2024

Acknowledgements

This acknowledgment is mainly dedicated to the individuals who directly or indirectly contributed to the completion of this work.

First, I would like to thank my supervisor, David Krueger, for his time and guidance, always willing to provide insightful feedback and continuous support throughout this journey. His expertise was valuable in shaping the direction of this work.

Equally important was my co-supervisor, Usman Anwar, to whom I extend my deepest gratitude. His constant availability and constructive discussions were instrumental in shaping and refining this work. His insightful suggestions significantly contributed to the progress of it.

A special thanks goes to Stefan Schoepf and Jack Foster for their invaluable insights and their consistently welcoming approach to my questions and ideas. Their willingness to engage together with the project and provide thoughtful dialogue has greatly enhanced the quality of this work. Additionally, they provided valuable extracts of code for Section 3.5, and their previous work on Machine Unlearning has been relevant for this work.

I would also like to acknowledge my dear friend, Fernando Cocoletzi. He was a hidden figure, hard to contact but whose questions and keen insights were among the most important I received during this project. My friend, you remain the most impressive genius I have met so far.

Additionally, I would like to extend my heartfelt appreciation to everyone who has supported me in various ways throughout this journey. In particular, I am grateful to those who noticed and felt the absence of my presence during the final weeks of this project. Your encouragement, patience, and understanding have been invaluable and deeply appreciated.

Finally, I extend my gratitude inward, to the self that has walked this path with unwavering resolve. For being there in moments of doubt, for finding strength in the quiet hours, and for persisting when the road seemed steepest—I thank the self that stood by me, a silent companion through every challenge. To you my always present friend: thank you.

Abstract

In this work we explore the application of hypernetworks, neural networks that learn to generate parameters for other networks, for the field of Machine Unlearning. We introduce HyperForget, a framework for Machine Unlearning that leverages the power of hypernetworks to dynamically generate model parameters capable of selectively forget specific data while retaining critical knowledge and capabilities.

By integrating diffusion models within the HyperForget framework, we introduce two approaches for constructing a Diffusion HyperForget Network. These models are capable to sample unlearned models with different configurations as requested by an user.

We present Proof-of-Concept (POC) scenarios to evaluate the applicability of these models on different unlearning tasks. The corresponding results indicate the potential application of this type of models, specifically hypernetworks, for Machine Unlearning.

Finally, we discuss the models limitations and current challenges for their practical use, including issues related to scalability, potential recovery of forgotten information, generalization capabilities, and robustness. And comment on future research directions aiming to refine the HyperForget framework and extend its applicability and potential benefits.

Overall, the HyperForget framework represents a potential approach for the development of adaptive and dynamic Machine Unlearning algorithms that align with modern data governance and emerging AI standards.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AI Artificial Intelligence

CNN Convolutional Neural Network

DNN Deep Neural Network

DDPM Denoising Diffusion Probabilistic Model

DiHyFo Diffusion HyperForget Network

DiT Diffusion Transformer

FIM Fisher Information Matrix

GDPR General Data Protection Regulation

LLMs Large Language Models

MIA Member Inference Attack

ML Machine Learning

MLP Multi-Layer Perceptron

MSE Mean Squared Error

MU Machine Unlearning

POC Proof-of-Concept

RTBF Right to be Forgotten

SGD Stochastic Gradient Descent

Chapter 1

Introduction

1.1 An Enquiry Concerning Human Forgetting

The principal interest in this work is to equip Machine Learning models with the capacity to forget. As with any new AI development, we may start by rethinking how this process functions in humans. Traditionally, when it comes to human memory, the main interest is on how to remember, often viewing forgetting as negative. However, numerous studies in education, psychology, and neuroscience highlight that forgetting is just as crucial as remembering in cognitive processes (Cuddy and Jacoby, 1982; Fawcett and Hulbert, 2020; Kuhl et al., 2007; Storm et al., 2008).

Forgetting in a healthy brain happens dynamically, not merely as a memory error, and though often unintentional, it is crucial for human reasoning (Shuang Sha et al., 2024). While the psychological, biological, and neural mechanisms of forgetting are not fully understood yet, it is recognized as a key part of the brain's memory management system, alongside attention, memory acquisition, and consolidation (Davis and Zhong, 2017; Shuang Sha et al., 2024). And just as memory is vital for human learning and understanding, forgetting is equally important, which also applies to Machine Learning and possibly one day to Machine Understanding and Reasoning.

Similar to what happens at the moment with machines, at some point in history the human brain was regarded as a biological machine that forms and stores memory, with most forgetting happening naturally. This natural or passive forgetting includes memory decay over time, difficulties in information retrieval due to the erosion of cellular memory traces, or disconnections within the memory engram that render cells unresponsive to recall mechanisms, as well as the loss of context cues that can make certain memories harder to recall (Davis and Zhong, 2017).

While models do not experience natural forgetting as humans, they do exhibit processes analogous to certain active human forgetting mechanisms, which involves an effort to forget or suppress certain memories or information (Costanzi et al., 2021; Davis and Zhong, 2017; Shuang Sha et al., 2024). For instance, intrinsic forgetting in humans refers to the brain's ability to weaken memory traces, allowing it to prioritize important information while pushing less relevant memories to the background. This process helps maintain working memory capacity, reduces memory overload, and enhances comprehension, decision-making, and learning (Davis and Zhong, 2017; Shuang Sha et al., 2024). In Machine Learning, similar processes occur during training, where the model prioritizes and preserves essential knowledge within its parameters, and regularization techniques are employed to ensure the model focuses on relevant information, thereby preventing overfitting and information overload (Jagielski et al., 2022; Kulikovskikh and Prokhorov, 2018; Norman et al., 2007).

Studies on human forgetting have evolved to understand it not merely as a natural process of the brain but also as one that can be voluntary and conscious. Classical psychology, influenced by Freud and Jung's work on ego and repression, suggests that people forget as a defense mechanism to cope with painful, overwhelming, or distressing experiences (Cohen, 1985; Freud, 1922, 2014; Jung, 2014). Modern approaches recognize both unconscious and conscious processes, including the deliberate avoidance of thoughts related to traumatic events. Voluntary forgetting is crucial for mental health, as it allows individuals to manage undesirable memories and concentrate on more relevant information. This active forgetting can be triggered by external or internal stimuli, and recent clinical psychology studies suggest that deficits in active forgetting are associated with psychopathologies such as stress disorders, depression, and obsessive-compulsive disorder (Costanzi et al., 2021; Geraerts and McNally, 2008; Shuang Sha et al., 2024).

Forgetting can, therefore, be an active, voluntary effort to suppress or discard specific memories, and there are several methods for this. Motivated forgetting occurs when individuals consciously engage cognitive mechanisms to suppress certain thoughts, and with practice, people can learn to manage and reduce the impact of these memories. Directed forgetting is another cognitive process where individuals are instructed to forget specific information, which can happen through active suppression (weakening the memory trace) or retrieval inhibition (blocking the memory from being recalled). In cognitive-behavioral therapy, cognitive restructuring is used to help individuals change their perspectives on certain memories. By altering the emotional significance of a memory, its impact can be reduced, making it easier to forget. Additionally, replacing a distressing memory with a more neutral or positive one can cause the negative memory to fade into the background over time (Basden et al., 1993; Costanzi et al., 2021; Marks et al., 1998).

Emotional regulation techniques like mindfulness, meditation, and relaxation exercises can help manage emotional responses tied to specific memories, aiding in the forgetting process. Exposure therapy, often used to treat phobias and trauma, involves controlled exposure to distressing memories or cues, gradually reducing their emotional impact and making them less intrusive (Costanzi et al., 2021; Gamboa et al., 2017; Van Vugt and Jha, 2011; Zoellner et al., 2011).

Retrieval-Induced Forgetting occurs when actively recalling some memories suppresses related, non-retrieved ones, helping to focus cognitive resources on the most relevant memories but sometimes leading to the unintentional loss of valuable information (Davis and Zhong, 2017; Murayama et al., 2014). Interference-based forgetting, where competing information or activities accelerate memory decay, is particularly relevant in contexts requiring a balance between retaining and discarding information. These forms of active forgetting are reflected in some approaches to make Machine Learning models to forget (Endress and Johnson, 2021; Norman et al., 2007; Riemer et al., 2018).

In Machine Learning, the intentional removal of specific data influences from a trained model is part of an emerging area known as Machine Unlearning (Bourtoule et al., 2021; Shuang Sha et al., 2024). This process is complex because the influence of data on model performance and decision-making is significant, and the boundaries of what should be forgotten must be carefully considered to avoid negative outcomes.

In cognitive psychology, where forgetting is more about retrieval than storage, excessive forgetting can have severe consequences, such as loss of identity, impaired learning, poor decision-making, and cognitive decline, as seen in Alzheimer's disease (Moulin et al., 2002; Shuang Sha et al., 2024; Wixted, 2004). Similarly, in Machine Learning, unmanaged data forgetting can degrade model performance. While humans adapt to forgetting through cognitive strategies, Machine Learning models need careful design and tuning to mitigate the effects of excessive unlearning. Balancing forgetting is essential for both humans and machines to ensure memory remains useful.

Just as forgetting and memory are vital in human learning, refining methods for Machine Learning models to forget is key to create systems that not only learn effectively but also strategically unlearn and adapt, leading to more robust and ethical AI applications.

This work explores a methodology inspired by human forgetting theories that view memory as an active reconstruction process. Similar to learning, forgetting can be seen as a generative process involving a sequence of states with varying levels of forgetting. The process transitions from lesser to greater forgetting (Fawcett and Hulbert, 2020; Frise, 2018; Heng and Soh, 2024; Shuang Sha et al., 2024). This perspective positions forgetting as the

inverse process of learning—an unlearning process. By reconstructing these states, we aim to place the Machine Learning model in a state where it effectively forgets targeted information.

1.2 From a Nice-To-Have to a Requirement

The need for Machine Learning models to forget their training data has evolved from a research interest to a crucial requirement in today's Artificial Intelligence (AI) landscape. Recent studies highlight the security risks posed by poisoned attacks and backdoor techniques, where malicious actors manipulate models by introducing poisoned data (Carlini et al., 2024; Goel et al., 2024; Schoepf et al., 2024). Additionally, adversarial attacks have been used to reveal whether specific data instances were part of a model's training set (Cao and Yang, 2015; Marchant et al., 2022; Nasr et al., 2023; Ren et al., 2020), making data forgetting not just a security measure but a critical defense mechanism.

Moreover, societal biases related to aspects such as race or gender often originate from features within the training data, which Machine Learning models may unintentionally learn and propagate (Chen et al., 2024; Dinsdale et al., 2020, 2021). In such cases, unlearning the data associated with these biases is a strategy to correct model behavior and mitigate bias in decision-making.

The most compelling reason for implementing forgetting mechanisms, however, comes from the evolving landscape of data privacy regulations and AI governance. As society becomes increasingly data-driven, vast amounts of personal information are continuously collected and processed. The legal implications of handling such data have become a focal point for regulators, particularly when it pertains to identifiable individuals. Privacy is recognized as a fundamental human right in documents like Article 12 of the Universal Declaration of Human Rights (United Nations, 1948) or Article 8 of the European Convention on Human Rights (Council of Europe, 1950). Further, Article 8 of the EU Charter of Fundamental Rights explicitly guarantees the protection of personal data:

"Everyone has the right to the protection of personal data concerning him or her. Such data must be processed fairly for specified purposes and on the basis of the consent of the person concerned or some other legitimate basis laid down by law. Everyone has the right of access to data which has been collected concerning him or her, and the right to have it rectified. Compliance with these rules shall be subject to control by an independent authority" (European Union, 2000).

As a result, data privacy has become a primary concern for regulators worldwide, leading to the establishment of constitutional provisions, national laws, or international treaties, that

define the legal boundaries for accessing, processing, and utilizing personal data (Zhang et al., 2023a).

In the current regulatory environment, with emerging AI regulations on the horizon, frameworks like the European Union's General Data Protection Regulation (GDPR) have set a precedent for data privacy protection (Shastri et al., 2019; Voigt and Von dem Bussche, 2017). GDPR grants individuals key rights, including the Right to be Informed (Articles 13-14), which requires immediate notification when data is collected directly, or within a month if indirectly. The Right of Access (Article 15) allows individuals to request access to and understand how their data is processed. The Right to Rectification (Article 16) ensures that inaccuracies in personal data are corrected, maintaining data accuracy. The Right to Erasure (Article 17), also known as the Right to be Forgotten (RTBF), allows individuals to request the deletion of their data under specific conditions, with data controllers typically required to comply within a month. This right balances individual privacy with the legitimate interests of organizations in processing data for research, freedom of expression, or public interest. Together, these rights empower individuals to control their personal data and enforce privacy protections (European Union, 2016).

The RTBF is particularly relevant, allowing individuals to request the deletion of their data. Initially created in response to search engine data collection (Hoofnagle et al., 2019), this right now extends to organizations using AI models, requiring them to have mechanisms to delete data used for training models. Moreover, the recent EU AI Act states that:

"the right to privacy and to protection of personal data must be guaranteed throughout the entire lifecycle of the AI system... the principles of data minimisation and data protection by design and by default... are applicable when personal data are processed. Measures taken by providers to ensure compliance with those principles may include not only anonymisation and encryption, but also the use of technology that permits algorithms to be brought to the data and allows training of AI systems without the transmission between parties or copying of the raw or structured data themselves..." (European Commission, 2024)

But different to search engines, Machine Learning models capture information of their training data in their parameters. Thus, to align Machine Learning models with ethical standards and regulatory requirements it is necessary to not only delete stored data but also to eliminate its influence on the model's parameters, which directly affects the model's capabilities (Bourtoule et al., 2021; Foster et al., 2024b).

On the other hand, regulations governing predictive models and model risk have long existed, particularly in sectors where their use has had historically significant impacts, such as credit scoring, insurance underwriting, and financial trading (Bessis, 2011; Gatzert and Wesker, 2012; King and Tarbert, 2011). But different to traditional statistical models, AI

models such as Deep Neural Networks (DNNs) and Large Language Models (LLMs) have shown capabilities for memorizing and reproducing training data, posing new challenges for compliance with data privacy regulations (Liu et al., 2024a; Zhang et al., 2023a).

As AI continues permeating various aspects of society, the frequency of attacks to Machine Learning models increase, and personal data sources expand, These mechanisms are no longer just research interests; they are critical for meeting regulatory, ethical, and security standards. In this context, this work delves into Machine Unlearning, a growing field in AI focused on enabling models to forget specific training data while retaining overall performance. We propose a novel approach using diffusion models and hypernetworks, offering a potential promising direction for develop AI systems that align with modern AI and data standards.

1.3 Outline

This dissertation is structured as follows:

- Chapter 2. Introduces the foundational concepts and methodologies in Machine Unlearning, setting the stage for the following chapters. A high-level overview of diffusion models is provided, followed by an exploration of hypernetworks and their potential in developing adaptive Machine Learning methods. The chapter also reviews key prior works that form the basis for the methods proposed in this dissertation.

- Chapter 3. Presents the HyperForget framework, detailing how hypernetworks can be applied to Machine Unlearning in classification tasks. Two implementations, DiHyFo-1 and DiHyFo-2, are introduced, each utilizing diffusion models to sample unlearned neural networks. The chapter explores their unlearning capabilities, demonstrating the framework's potential benefits and its promising direction for future research.

- Chapter 4. Examines the limitations of the proposed models and framework, addressing challenges and implications for broader deployment. The chapter also summarizes the key insights from this research and suggests directions for future work to further advance the field of Machine Unlearning.

Chapter 2

Background

This chapter introduces the theoretical foundations for the rest of this work. Section 2.1 covers the core concepts of Machine Unlearning, including definitions, model development methodologies, evaluation approaches, and current challenges, establishing key terminology for subsequent chapters.

Section 2.2 explores diffusion models, focusing on their mathematical principles and their role in generative modeling.

Finally, Section 2.3 examines hypernetworks, discussing their architecture, functionality, and how they enable dynamic parameter generation. The integration of hypernetworks with diffusion models is highlighted, setting the stage for subsequent chapters.

2.1 Machine Unlearning

Machine Unlearning focuses on developing algorithms capable of efficiently and effectively removing the influence of specific data on a model, while ensuring that unrelated model's knowledge or capabilities remain unaffected (Liu et al., 2024a). The data to forget can include specific data points, features, classes, or even entire learning tasks upon request (Bourtoule et al., 2021; Nguyen et al., 2022).

In this context, the subset of a dataset D that should be forgotten D_f is referred to as the "unlearning target" or "forget set", while the data that should remain unaffected $D_r = D \backslash D_f$ constitutes the "retain set". The primary goal is to remove the influence of the forget set on the model's performance while preserving its performance for the retain set (Liu et al., 2024a). Definition 2 formally defines an unlearning algorithm upon the Definition 1 of a learning algorithm.

Definition 1 (Learning Algorithm) *Let Z denote an example space with an associated power set $P(Z)$. Let $D \subseteq Z$ represent the training dataset, and H the set of all possible hypothesis or models that map inputs to outputs based on the training data. A learning algorithm $A : P(Z) \to H$ is a function that takes the training dataset and identifies the best hypothesis that fits it $h \in H : A(D) = h$ by minimizing over time an objective error measure of the model's predictions, $E : H \times P(Z) \to \mathbb{R}$, $h = argmin_{h' \in H} E(h', D)$. The final model obtained can then be used to make predictions or decisions based on new inputs $y = h(x)$, where $x \in Z$.*

Definition 2 (Unlearning Algorithm) *An unlearning algorithm is a function U that takes as input a training set $D \subseteq Z$ with two identified disjoint subsets, the forget set $D_f \subseteq D$ and retain set $D_r = D \backslash D_f$, along with a learning algorithm $A(D)$. The output is an unlearned model $U(D, D_f, A(D)) \in H$, which is expected to perform equivalently or similarly to a model that has been trained without the forget set, $A(D \backslash D_f)$.*

Thus, unlearning guarantees that training on a point and subsequently unlearning it produces a distribution of models similar to what would have been obtained if the point had never been included in training (Bourtoule et al., 2021; Tarun et al., 2023a). Depending on how "unlearning" is defined, different objective measures can be established. According to our definition, unlearning refers to updating a pre-trained model to lose its capabilities on the forget set while retaining them for the retain set. This leads to the commonly used unlearning objective measure presented in Definition 3 (Liu et al., 2024a; Nguyen et al., 2022).

Definition 3 (Unlearning Objective Measure) *Let $u = U(D, D_f, A(D)) \in H$ be an unlearned model with output after unlearning $y_f = u(x)$, $x \in Z$. The goal is to find the parameters that minimize the expected loss on the forget set post-unlearning while retaining a level λ of error on the retain set as it was pre-unlearning:*

$$\min_{\theta} \left(E_{D_f} \left(\mathscr{L}(y_f \mid x; \theta) \right) + \lambda E_{D_r} \left(\mathscr{L}(y \mid x; \theta) \right) \right) \tag{2.1}$$

Consequently, unlearning is closely related to analyzing the influence of model parameters or architectural components in the input-output interactions within the model (Liu et al., 2024a; Zhang et al., 2023b). And the definition of "unlearning" and its objectives characterize the unlearning problem and establish the design principles underlying unlearning algorithms, which we now briefly explore.

2.1.1 Exact Unlearning

According to Definition 2, a necessary and sufficient condition for a model to achieve exact unlearning is that it produces the same results as a model that has not encountered the forget set during training. More generally, for randomly initialized models, exact unlearning is achieved when the distribution of unlearned models is identical to the distribution of models trained on the dataset excluding the forget set (Kurmanji et al., 2024; Thudi et al., 2021).

$$P(U(D,D_F,A(D))) = P(A(D\backslash D_f)) \tag{2.2}$$

As pointed out by Brophy and Lowd (2021); Ginart et al. (2019); Nguyen et al. (2022) an issue with this condition is that while two models trained with the same dataset should belong to the same distribution, defining this distribution is complicated. To avoid the unlearning algorithm being specific to a particular training dataset, a more general definition for exact unlearning can be provided in Definition 4. This allows for defining a metric space for the models and, consequently, for the distributions, which depends on whether we treat a model as a mapping between inputs and outputs or as architectures defined by specific parameters. Therefore, we can define and work with unlearning methods over a function space or a weight space.

Definition 4 (Exact Unlearning) *A sufficient and necessary condition for a process U(.) to be an exact unlearning process is that*

$$\forall B \subseteq H, D \in Z*, D_f \subset D : P(U(D,D_F,A(D)) \in B) = P(A(D\backslash D_f) \in B) \tag{2.3}$$

Indeed, unlearning can be accomplished either by equating the distribution of parameters of the learned and unlearned models or the distribution of their outputs. [1] However, comparing distributions over either of these spaces is computationally expensive. As a result, common approaches utilize alternative metrics on a point-by-point basis to compute the distance between the two models (Bourtoule et al., 2021; Chen et al., 2024; Goel et al., 2024; Nguyen et al., 2022).

The simplest and most effective approach to achieve exact unlearning for a pre-trained model is to retrain it from scratch after removing the unlearning targets. This guarantees absolute compliance with Definition 4. As result, retraining is considered the only exact unlearning method and serves as gold standard for benchmarking most Machine Unlearning methods (Cao and Yang, 2015; Zhang et al., 2023b).

[1] In literature this is sometimes refereed as weak exact unlearning (Baumhauer et al., 2022; Nguyen et al., 2022)

Nevertheless, retraining has several practical disadvantages. It requires access to the full training dataset, which might not always be possible, and can be costly in terms of time and computational resources. These limitations are more prominent in scenarios involving multiple or recurrent deletion requests, where the model would need to be retrained from scratch each time it is required to forget a data point (Chundawat et al., 2023b; Liu et al., 2024a). In response, Machine Unlearning research has focused on developing methods to approximate exact unlearning as closely as possible while mitigating its associated drawbacks.

2.1.2 Approximate Unlearning

In principle, neither Definition 4 nor Equation 2.2 requires the unlearning model to be a model retrained without the unlearning targets, but that it behaves as it had not been exposed to the forget set during training.

As stated in Definition 5, approximate unlearning methods aim to replicate the behavior of a retrained model as closely as possible without retraining from scratch, thereby addressing some of the practical drawbacks of retraining. Common approaches include performing less computationally intensive operations on the final weights, modifying the model architecture, filtering outputs, or developing strategies to not require access to training data (Bourtoule et al., 2021; Chundawat et al., 2023b; Graves et al., 2021; Guo et al., 2019; Sekhari et al., 2021).

Definition 5 ($\varepsilon-$Approximate Unlearning) *Given $\varepsilon > 0$, an unlearning algorithm U performs ε-certified removal of the sample z for a learning algorithm A if*

$$\forall B \subseteq H, D \in Z*, z \in D : e^{-\varepsilon} \leq \frac{P(U(D,z,A(D)) \in B)}{P(A(D\backslash z) \in B)} \leq e^{\varepsilon} \tag{2.4}$$

Equivalently, for an arbitrary absolute distance metric $||.||$ in the weight space or the output space:

$$log(P(U(D,z,A(D)) \in B) - P(A(D\backslash z) \in B)) \leq e^{\varepsilon} \tag{2.5}$$

Approximate unlearning is often related to Differential Privacy (Definition 6), which focuses on ensuring that the parameters of a trained model do not leak information about any particular individual. This presents a stronger condition than approximate unlearning, requiring that the distribution of parameters remains almost unchanged even after replacing a sample. Also, most differentially private models suffer a significant loss in accuracy, even for large values of ε, because the privacy constraints limit the model's ability to effectively learn from the data (El Ouadrhiri and Abdelhadi, 2022; Guo et al., 2019; Ji et al., 2014).

Definition 6 (Differential Privacy) *For $\varepsilon > 0$, an algorithm U is differentially private if it performs ε-certified removal of the sample z for a learning algorithm A, accomplishing*

$$\forall B \subseteq H, D, D' : e^{-\varepsilon} \leq \frac{P(A(D) \in B)}{P(A(D \backslash z) \in B)} \leq e^{\varepsilon} \tag{2.6}$$

Notably, Definition 5 describes approximate unlearning with respect to a single sample z. Providing constant bounds for larger subsets of D is still an open problem, but a relaxed definition can be provided in Definition 7, by introducing an upper bound on the probability for maximum divergence, covering potential failures with Definition 5 (Liu et al., 2024a; Wang et al., 2024b).

Definition 7 ((ε, δ)-Approximate Unlearning) *Given $\varepsilon, \delta > 0$, an unlearning algorithm U performs ε-certified removal of a sample z for a learning algorithm A if $\forall B \subseteq H, D \in Z^*, z \in D$:*

$$P(U(D, z, A(D)) \in B) \leq e^{\varepsilon} P(A(D \backslash z) \in B) + \delta \tag{2.7}$$

and

$$P(A(D \backslash z) \in B) \leq e^{\varepsilon} P(U(D, z, A(D)) \in B) + \delta \tag{2.8}$$

Through these definitions and our previous discussion on exact unlearning, we can identify desirable conditions for a robust unlearning algorithm, [2] which have been highlighted in related works (Bourtoule et al., 2021; Chundawat et al., 2023b; Ginart et al., 2019; Golatkar et al., 2020; Guo et al., 2019; He et al., 2021; Micaelli and Storkey, 2019; Nguyen et al., 2022; Shuang Sha et al., 2024; Tarun et al., 2023b; Yoon et al., 2022; Zhang et al., 2023b):

1. **Completeness (Consistency)**. An unlearned model should yield the same predictions as a retrained model. This can be measured by the percentage of matching predictions on a test dataset, which also serves as an optimization target in unlearning setups.

2. **Accuracy.** The unlearned model should maintain accuracy similar to the original model, ideally comparable to a retrained model. In the absence of a retrained model, comparing the accuracy of the unlearned model with the original on a new test set is sufficient.

3. **Timeliness.** The unlearning process should be faster than retraining from scratch. While there may be a trade-off between completeness and timeliness, unlearning is preferred when the forget set is small, minimizing time and accuracy impact. For large forget sets, retraining may be more suitable to prevent performance degradation.

[2] From this point onward, unless otherwise specified, with unlearning algorithm we refer to an approximate unlearning algorithm.

4. **Efficiency.** The unlearning algorithm should be lightweight and scalable with data size. Extra computational costs beyond unlearning time and storage should be minimized, especially in methods requiring model checkpoints or historical data storage.

5. **Provable guarantees.** Providing provable guarantees of unlearning is essential to ensure the forgotten data's influence is effectively removed.

6. **Model-agnostic.** The unlearning method should be applicable to various machine learning models, making it versatile across different architectures and use cases.

7. **Verifiability (Privacy guarantees).** Implementing a mechanism to verify that the unlearned model adequately protects users' privacy is important.

8. **Irrecoverably.** In strict policy compliance, an unlearned model should not recover deleted data or associated knowledge. However, in some scenarios, recoverable forgetting is allowed, offering flexibility and control over specific knowledge while ensuring data protection.

While these conditions are crucial for policy compliance and model fairness, achieving them all simultaneously is challenging. When selecting or designing an unlearning algorithm, it's important to consider the involved trade-offs. For instance, completeness and accuracy are vital in applications like medical diagnosis or autonomous driving, where even minor deviations can have serious consequences (Chatterjee et al., 2024; Nasirigerdeh et al., 2024; Song et al., 2023). In contrast, scenarios requiring quick data removal may favor faster, less complete unlearning methods, particularly for small forget sets. Ultimately, the unlearning strategy should be tailored to the application's specific needs, the forget set size, and available computational resources.

2.1.3 Restricted Data Scenarios

As previously discussed, retraining requires full access to training data, and unlearning algorithms often rely on the entire dataset or subsets of it. However, in some scenarios access to training data is restricted at different levels or unavailable, even for unlearning purposes. This requires adapting unlearning algorithms to work with only a portion of the data or none at all, introducing stricter conditions and new algorithm types:

- **Zero-Glance Unlearning**: It presents an scenario where only the retain set can be used, and the unlearning algorithm must rely entirely on a subset of this retained data, $D_{r_{sub}}$, establishing the problem setting as $U(D_{r_{sub}}, A(D))$. It often involves approximating the

forget set using indirect methods, such as learning a noise matrix to maximize loss for forgotten classes (Chundawat et al., 2023b; Tarun et al., 2023b)

- **Few-Shot Unlearning**: Only a small portion of the forget set is available, leading to the problem setting $U(D, D_{f_{sub}}, A(D))$. It is useful for addressing mislabeled data or mitigating malicious effects when full access to D_f is restricted (Yoon et al., 2022). Methods include Model Inversion and Influence Approximation, which estimate sample impact using the Hessian matrix or Fisher Information Matrix, followed by noise injection to reduce influence (Chundawat et al., 2023b; Nguyen et al., 2022).

- **Zero-Shot Unlearning**: This method operates without access to both retained and forget sets (Chundawat et al., 2023b). Existing methods rely on existing model parameters to simulate unlearning, using noise or modifications to minimize forget set errors, or applying constrained knowledge distillation to prevent the transfer of forgotten knowledge (Micaelli and Storkey, 2019).

- **Retrieval-Enhanced Unlearning**: We have named Retrieval-Enhanced to a group of approaches that rely on stored metadata or auxiliary data saved during training, distinct from the actual training data. For example, approximating the steps of SGD used in retraining from scratch. A relevant concern is scalability, particularly in large models where metadata storage and retrieval may be challenging (Chundawat et al., 2023b; Liu et al., 2024a; Nguyen et al., 2022; Thudi et al., 2021)

These methods are crucial for scenarios with restricted data access but introduce new complexities and trade-offs. Ongoing research should focus on improving robustness, scalability, and providing stronger guarantees of data removal, even in restrictive environments, while establishing benchmarks and evaluation criteria to assess their effectiveness across use cases.

2.1.4 Unlearning Algorithms Categorization

Unlearning algorithms can be categorized based on the implementation moment. Pre-training methods are proactive, embedding unlearning capabilities into the model's design or training process. These methods are efficient at removing data points, as forgetting is built into the model, making them robust and reliable. However, they often require more complex design and higher computational overhead during training (Bourtoule et al., 2021; Wang et al., 2024b; Zhang et al., 2023b).

Post-training methods are reactive, applied after the model is trained. These methods are popular because they can be used on existing models without redesign, allowing for quick

updates. However, they may not always fully remove data influence, potentially falling short of forgetting objectives or policy requirements (Chundawat et al., 2023b; Foster et al., 2024b; Goel et al., 2024).

Further, we can categorize unlearning algorithms by their scope. Model-Intrinsic methods are tailored to specific models, e.g., Fisher-based unlearning for DNNs, leveraging the model's architecture for effective unlearning but lacking broad applicability (Chen et al., 2024; Xu et al., 2024).

Model-Agnostic methods work across various models, offering flexibility. While they provide practical unlearning guarantees, theoretical guarantees are often limited to specific model classes, such as linear models. Examples include differential privacy, scrubbing functions, and certified removal mechanisms (Golatkar et al., 2020; Guo et al., 2019; Kurmanji et al., 2024; Nguyen et al., 2022).

On the other hand, Data-Driven approaches involve modifying or configuring the data to facilitate unlearning or speed up retraining. A notable example is SISA (Sharded, Isolated, Sliced, and Aggregated), which partitions data into shards, each tied to a specific model. When unlearning is needed, only the model linked to the shard containing the forget target requires retraining, greatly accelerating the process (Bourtoule et al., 2021). Other methods might use data augmentation to prompt the model to ignore certain data or apply influence functions to relate changes in training data to the model's parameters (Chen et al., 2024; Chundawat et al., 2023b; Li et al., 2023; Wu et al., 2022).

Table 2.1 Categorization of Machine Unlearning Algorithms

Request Type	**Unlearning Level**	**Entity of Application**	**Data Access Requirement**	**Unlearning Properties**
Item	Exact	Model-Agnostic	Full Training Data	Completeness
Feature	Approximate	Model-Intrinsic	Zero-Glance	Accuracy Preservation
Class		Data-Driven	Few-Shot	Timeliness
Task			Zero-Shot	Lightweight
Stream			Metadata	Unlearning Guarantee
				Verifiability
				Irrecoverability

Together with previous discussion, Table 2.1 summarizes the distinct attributes that serve to categorize an unlearning algorithm. Clearly, an unlearning algorithm can fulfill different criteria at different degrees. This categorization framework, though not exhaustive, provides a comprehensive enough structure for understanding and analyzing different unlearning methods.

2.1.5 Catastrophic Unlearning

As discussed in Section 1.1, similar to cognitive processes in humans, excessive or improper forgetting in Machine Learning models can significantly reduce their capabilities. It's crucial to preserve as much of the original model's functionality as possible during unlearning.

While some performance degradation is expected in unlearned models compared to the original or a retrained model, severe losses can lead to catastrophic unlearning, making the model unusable (Kurmanji et al., 2024; Wang et al., 2024b). This can occur when the unlearning process inadvertently causes the model to lose essential capabilities or introduces new vulnerabilities, such as the Streisand effect—where the unlearning process itself or specific elements of the forget set become identifiable, evidencing the purpose of unlearning and increasing the risk of adversarial attacks (Foster et al., 2024a,b).

For example, Chundawat et al. (2023a) describes a case where an unlearned model misclassifies an aircraft as a mushroom, a deliberate misprediction that could leak information to an attacker. This scenario shows that the model hasn't truly unlearned the data but has instead learned to predict incorrect labels for the targeted forget samples.

In critical domains like medicine, finance, or autonomous systems, such failures can have serious consequences. If a medical model forgets key data, it might missrecognize a condition, leading to incorrect diagnoses and treatments (Nasirigerdeh et al., 2024), undermining trust in the model. Such failures not only undermine trust in the model but can also result in significant harm.

From Definition 7, greater deviation in behavior from a retrained model increases the risk of catastrophic unlearning. To mitigate this, some studies introduce additional safeguards, such as evaluating a model's susceptibility to Membership Inference Attacks (MIA)—a type of attack where an adversary attempts to determine whether a specific data point was part of the model's training data- as a measure of unlearning effectiveness (Foster et al., 2024b; Tarun et al., 2023a).

Catastrophic unlearning represents a significant challenge in Machine Unlearning, emphasizing the need for careful algorithm design and thorough evaluation to prevent compromising model integrity or introducing new risks, especially the Streisand effect.

2.1.6 Unlearning Evaluation and Verification

Machine Unlearning evaluation focuses on assessing how well an unlearning method meets the requirements defined in Section 2.1.4. As previously mentioned, the gold standard is a model retrained from scratch, and common criteria include:

- **Completeness (Prediction alignment):** This measures how closely the unlearned model's predictions align with those of a retrained model on the forget, retain, and the test set, often by comparing output overlaps (Foster et al., 2024a; Zhang et al., 2023b).
- **Timeliness:** This assesses the time efficiency of the unlearning algorithm compared to retraining from scratch. An efficient unlearning algorithm should offer a significant time advantage, especially for large models or datasets
- **Relearn Time:** The time needed for an unlearned model to recover performance on the forget set, serving as a proxy for residual information retention, as with more retained information it is easier to relearn (Nguyen et al., 2022).
- **Amnesia Index (AIN).** This metric introduces a margin to accuracy to calculate the relearn time, addressing the variability in relearn time across different models and datasets when used as a proxy measure of residual information retention. This metric ranges from 0 to ∞, with values close to 1 indicating good forgetting. Lower values suggest retained information, and higher values may signal the Streisand effect (Graves et al., 2021).
- **Activation Distance and JS Divergence:** These metrics assess how closely the unlearned model's parameters and activations align with a retrained model, aligning with Definition 4 (Foster et al., 2024a; Jeon et al., 2024).
- **MIA and Inversion Attacks:** These evaluate the model's resilience to adversarial attacks aimed at identifying or recovering forget set data. Ideally, the probability of a successful adversarial attack should be significantly lower in the unlearned model compared to the original model (Foster et al., 2024b; Graves et al., 2021).
- **Zero-Retrain Forgetting:** This metric is independent of retraining. It compares the unlearned model's predictions to those of an unskilled model, usually set to be a randomly initialized model, indicating whether the model behaves randomly or retains patterns on the forget set. Alternatively, fictitious data can be introduced into the model training, then fine-tuning can be used to simulate the retraining over this set (Chundawat et al., 2023a; Nguyen et al., 2022).
- **Epistemic Uncertainty:** This measures the certainty that the current model parameters are optimal for a given dataset, often using the FIM's trace to measure the information contained in the model parameters after unlearning. A drawback of this specific implementation with FIM is that it assesses overall information reduction, not specifically the reduction related to the forget set.

Combining these metrics provides a comprehensive view of unlearning effectiveness. For example, MIA alongside accuracy metrics can help ensure that models are less exposed to information leaks still maintaining good predictive performance (Becker and Liebig, 2022; Foster et al., 2024b).

On the other hand, while unlearning evaluation assesses how well an unlearning method performs, unlearning verification focuses on certifying the actual success of the unlearning process. This distinction is similar to the difference between model evaluation and stress testing in traditional Machine Learning. Although some methods overlap, unlearning verification emphasizes proving that the model has genuinely forgotten the specified data, while evaluation offers bounded guarantees useful for algorithm optimization (Nguyen et al., 2022; Sommer et al., 2020; Wang et al., 2024b). Together, these processes ensure the effectiveness and reliability of unlearning.

A key aspect of unlearning verification is privacy certification, ensuring no information from the forget set remains in the model. Popular methods for unlearning verification include:

- **Privacy Attacks:** Privacy attacks, such as MIA, are used to assess the degree to which the unlearned model has forgotten data. If an attack's success rate is bounded by α, the model is said to have α-forgotten the data, aligning with Definition 5.

- **Feature Injection Test:** This test checks if the unlearned model has adjusted its parameters based on a feature associated with the forget set. This method is particularly applicable to models with explicit parameters, e.g., linear models (Nguyen et al., 2022).

- **Information Leakage:** This method compares models before and after unlearning to measure how much information has leaked, often by analyzing output distributions (Chen et al., 2021; Guo et al., 2019).

- **Backdoor Attacks:** By injecting backdoor samples into data, these attacks test whether the unlearned model can forget these samples effectively. For example, training a model mixing clean and poisoned data items and then testing whether the poisoned samples can be forgotten effectively (Goel et al., 2022b; Liu et al., 2022).

- **Slow-Down Attacks:** These attacks introduce poisoned data subsets to slow down the unlearning process, used when comparing an unlearned model with a retrained one is not feasible. Developing metrics to efficiently compute the cost of removing these subsets remains an open research area (Carlini et al., 2024; Liu et al., 2024b; Marchant et al., 2022).

- **Inter-Class Confusion Test:** This test swaps the labels of two selected classes in a data subset, then retrains the model with the swapped labels as the forget set. The confusion matrix from the unlearned model is analyzed to compute a forgetting score, determining whether information from the forget set remains inferable (Goel et al., 2022a,b; Nguyen et al., 2022).

Combining these evaluation and verification methods offers a comprehensive assessment of an unlearning algorithm's effectiveness. The interplay between evaluation and verification ensures unlearning methods achieve real and reliable forgetting. Notably, while the field of unlearning has gained more attention recently, there is still no universally accepted frameworks or benchmarks for evaluating unlearning methods rather than the model retrained from scratch without forgetting targets.

2.1.7 Challenges in Machine Unlearning

As a relatively new area of research, Machine Unlearning faces significant challenges, particularly the lack of standardized frameworks and benchmarks. Unlike other Machine Learning fields, there are no universally accepted datasets or evaluation criteria, making it difficult to assess and compare unlearning methods, which slows progress. Establishing shared resources and benchmarks is essential for rigorous testing and validation of unlearning algorithms, facilitating their improvement and broader adoption (Ginart et al., 2019; Liu et al., 2024a; Wang et al., 2024b).

Currently, most unlearning algorithms focus on data item removal and class unlearning, often overlooking other requests like feature removal or task-specific unlearning. Additionally, these methods are mainly developed for computer vision, leaving other domains with their own complexities underexplored. For instance, the vast data used to train LLMs complicates the precise definition and localization of unlearning targets or even to define the scope of the unlearning procedure (Liu et al., 2024a; Zhang et al., 2023a).

From what has been explored so far, one of the most critical challenges is the limited understanding of how individual data points and model components influence behavior and final outputs. In some scenarios, even with a small forget set, extensive data removal may be required to effectively erase its influence. A jointly analysis of both data and model components is crucial to accurately eliminate unlearning targets and preserve model capabilities. Methods that attempt to measure the influence of specific training points on model parameters often involve expensive computational procedures, and while some research has focused on identifying critical model components for unlearning tasks, such as layers and neurons, this area remains underexplored, particularly for large-scale models

(Bourtoule et al., 2021; Foster et al., 2024a,c; Goel et al., 2024; Liu et al., 2024a; Wang et al., 2024b)

Additionally, training is an incremental procedure, making each update depended on all previous updates. When a model is updated based on a particular training point, subsequent updates implicitly rely on that point, making it difficult to isolate and remove its influence during unlearning (Bourtoule et al., 2021).

Moreover, the stochastic nature of unlearning algorithms can also lead to variability in outcomes, complicating efforts to ensure consistency across different runs of the unlearning process (Nguyen et al., 2022). Additionally, challenges with timeliness—ensuring the unlearning process is both effective and efficient—and the lack of rigorous criteria for measuring the persistence of forget set information further complicate the field. More research is needed to understand and protect against weaknesses and attacks on Machine Learning and Unlearning systems (Goel et al., 2022a,b; Sommer et al., 2020).

Despite these challenges, Machine Unlearning presents opportunities for innovation. Unlearning can repair obsolete models by forgetting outdated data that conflicts with evolving data streams, and can also be applied to out-of-distribution data, imbalanced learning, and bias mitigation (Chen et al., 2024; Dinsdale et al., 2020; Liu et al., 2024a). Developing explainable Machine Unlearning algorithms could enhance unlearning verification, facilitate auditing of removed data, and improve transparency and accountability, thereby building trust in human-AI interactions (Nguyen et al., 2022; Wang et al., 2024b; Zhang et al., 2024).

These challenges and opportunities mark new frontiers for future exploration in Machine Unlearning. Addressing these issues will require a multidisciplinary approach, integrating advances in Machine Learning, privacy, security, and explainability to develop more reliable, efficient, and broadly applicable methods that enhance the safety and privacy of AI systems.

2.2 Diffusion Models

Since the seminal work of Sohl-Dickstein et al. (2015), later expanded by Ho et al. (2020) and Song et al. (2020), diffusion models have achieved significant success in various domains, including image generation (Dhariwal and Nichol, 2021; Saharia et al., 2022), audio synthesis (Huang et al., 2023; Kong et al., 2020; Zhu et al., 2023), and molecule design (Soleymani et al., 2024; Xu et al., 2022), positioning them as leading state-of-the-art techniques for generative modelling.

Diffusion models are a type of deep generative model inspired by diffusion processes in Stochastic Processes Theory and Non-Equilibrium Thermodynamics (Dobrow, 2016; Sohl-Dickstein et al., 2015). They operate in two main steps, a forward process that progressively

adds noise to the data, gradually degrading it into a simple noise distribution, and a reverse process that learns to reconstruct the original data from these noisy samples, effectively transforming random noise back into structured data. Thus, the model acts as a bridge between random noise and the structured data, which endows it with generative capabilities.

2.2.1 Denoising Diffusion Probabilistic Models

The general setup for a generative modeling problem involves a set of training data x drawn from an underlying distribution $q(x)$. The goal is to learn to sample from this distribution by approximating it with a probabilistic model $p(x)$, which can then be used to generate new samples that resemble the original data.

Following Ho et al. (2020), the generative modelling problem can be transformed into a supervised learning problem using a Markov Chain. Here, at each time step $t \in [0, \tau]$, $\tau \in \mathbb{N}$, noise is incrementally added to the data, degrading it across τ fidelity levels. The original data is at the highest fidelity level, x_0, and the lowest fidelity level represents pure noise, making $p(x_t)$ easy to sample from. This setup transforms the generative problem into a regression task, where the goal is to predict higher fidelity data from lower fidelity data, i.e., $p(x_{t-1} \mid x_t)$.

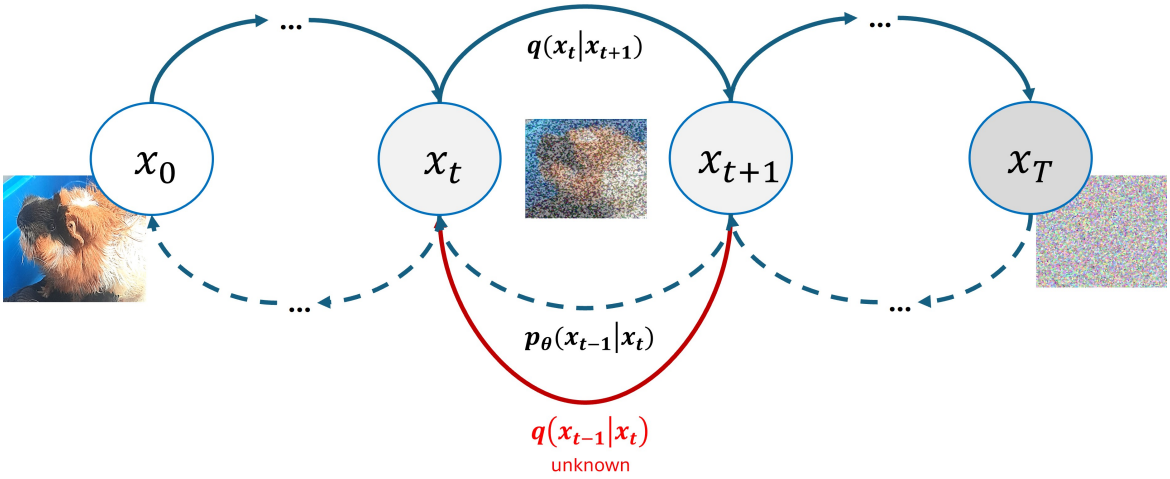


Fig. 2.1 Graphical Representation of Diffusion Process

Forward Process

During the forward process, the initial data x_0 is transformed through a Markov Chain, where at each step t, it is passed through a stochastic encoder $q(x_t \mid x_{t-1})$ that progressively adds random noise. This continues until the data is fully degraded into noise. The process is

typically modeled using a linear Gaussian approach, which defines a first-order autoregressive process characterized by a variance schedule $\beta_1, ..., \beta_\tau$, $\tau \in \mathbb{N}$. Thus, the forward process for some distribution $q(x)$ is defined by Equation 2.9, where as $\tau \to \infty$, the distribution of x_τ converges to an isotropic Gaussian distribution, indicating that the initial data x_0 eventually decays into Gaussian noise.

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta}x_{t-1}, \beta_t I) \tag{2.9}$$

Under the Markov property the joint distribution over all latent states conditioned on the initial input is defined in Equation 2.10.

$$q(x_{1:\tau} \mid x_0) = \prod_{t=1}^{\tau} q(x_t \mid x_{t-1}) \tag{2.10}$$

A key aspect of the forward diffusion process is that it allows for sampling from any arbitrary time step t by leveraging the marginals of the linear Gaussian process. Using a reparameterization trick, a common technique in variational inference and generative modeling (Ho et al., 2020; Luo, 2022), the process is expressed in closed form as follows:

$$\begin{aligned}
\alpha_t = 1 - \beta_t, \bar{\alpha} = \prod_{i=1}^{t} \alpha_i; \\
\Rightarrow x_t &= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1} \\
&= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon_{t-2} \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon \\
\Rightarrow q(x_t \mid x_0) &= \mathcal{N}(x_t; \sqrt{1-\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)
\end{aligned} \tag{2.11}$$

In this formulation, α_t and $\bar{\alpha}_t$ are functions of the variance schedule, and ε represents the standard Gaussian noise added at each step. This expression shows how data at any time step x_t can be directly related to the initial data x_0, with the noise contribution progressively increasing as t increases.

Reverse Process

The diffusion reverse process learns a decoder $p_\theta(x_{t-1} \mid x_t)$ to reconstruct the original data x_0 from the noisy state x_τ. It essentially reverses the forward process, denoising the data step by step to recover the initial structure. This is done by finding the reverse transitions that

maximize the likelihood of the forward transitions at each step. Once trained, the reverse process generates new data from random noise using the optimized denoising parameters θ and the reverse diffusion Markov chains (Wang et al., 2024a).

Given that Gaussian noise is introduced in the forward process, and assuming that this added noise at each step is relatively small, the reverse process can be also naturally modeled as Gaussian. Using Bayes' rule, the reverse process step is mathematically defined in Equation 2.12.

$$\begin{aligned} q(x_{t-1} \mid x_t, x_0) &= \mathcal{N}\left(x_{t-1}; \bar{\mu}_t(x_t, x_0), \bar{\beta}_t I\right) \\ \text{with} \quad \bar{\mu}_t(x_t, x_0) &= \frac{\bar{\alpha}_{t-1}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\bar{\alpha}_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t, \bar{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \end{aligned} \tag{2.12}$$

Here, $\bar{\mu}_t(x_t, x_0)$ represents the mean of the Gaussian distribution, which is a weighted combination of the current noisy state and the initial data. The variance $\bar{\beta}_t I$ reflects the uncertainty at each step of the reverse process.

To generate samples from from $q(x_{t-1} \mid x_t)$ starting from noise, a neural network is trained recursively to approximate the original data x_0 by removing the noise from each x_t. Thus, in the reverse process p, the mean $\mu_\theta(x_t, t)$ and variance $\Sigma_\theta(x_t, t)$ of the Gaussian distribution are estimated with the denoising network parameters θ. The generator takes the form in Equation 2.13, and the joint distribution over all generated variables is in Equation 2.14.

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \bar{\mu}_t(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.13}$$

$$p_\theta(x_{0:\tau} \mid x_t) = p(x_\tau) \prod_{t=1}^{\tau} p_\theta(x_{t-1} \mid x_t) \tag{2.14}$$

Training the reverse process involves optimizing a variational bound on the negative log-likelihood of the data. The objective is to minimize the difference between the true data distribution and the model's approximation, Equation 2.15

$$\mathbb{E}(-\log p_\theta(x_0)) \leq \mathbb{E}_q \left(-\log \frac{p_\theta(x_{0:\tau})}{q(x_{1:\tau} \mid x_0)} \right) = \mathbb{E}_q \left(-\log p(x_\tau) - \sum_{t \geq 1} \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})} \right). \tag{2.15}$$

This involves the expected negative log-likelihood of the original data under the model's distribution and the Evidence Lower Bound (ELBO), which is commonly used in variational inference to approximate complex distributions (Kingma and Gao, 2024). The goal is to

minimize the ELBO, thereby improving the model's ability to generate realistic data from noise. The training process iteratively adjusts the parameters θ of the denoising network to minimize this bound, ensuring that the reverse process accurately reconstructs the data distribution.

2.2.2 Diffusion Transformer

The implementation of DDPMs used to be dominated by Convolutional Neural Networks (CNNs), specially the U-Net. But in recent advancement, replacing CNNs with a transformer have represented a powerful approach for generative modelling.

Transformers, introduced by Vaswani (2017), have revolutionized the way we handle sequential data, being highly effective for tasks where understanding context and dependencies is crucial. They utilize self-attention mechanisms, Equation 2.16, to capture relationships between different parts of an input sequence $X = x_1, ..., x_n$ efficiently. This allows transformers to weight the importance of the elements in the input sequence X relative to each other, capturing long-range dependencies and complex relationships.

$$\text{Attention}(x_i) = \sum_j \text{softmax}\left(\frac{(W_Q x_i)(W_K x_j)^T}{\sqrt{d_k}}\right) W_V x_j \tag{2.16}$$

where d_k is the dimensionality of the key vectors and W_Q, W_K, and W_V are learned weight matrices for queries, keys, and values.

A standard transformer layer includes multi-head self-attention and feed-forward neural networks. The output of the self-attention layer is processed through Equation 2.17, where each head computes attention using different projections of Q,K, and V

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O \tag{2.17}$$

A Diffusion Transformer (DiT) (Peebles and Xie, 2023), Figure 2.2, integrates the transformer architecture into the diffusion framework to enhance the modeling of the reverse diffusion process $p_\theta(x_{t-1} \backslash x_t)$. For this, the input data is first transformed into a form that can be processed by the model, such as dividing it into fixed-size patches, which are then transformed into feature vectors. Noise is gradually introduced to the preprocessed feature vectors and then passed to the model to learn to denoise them.

The self-attention of the transformers allows to capture complex dependencies and relationships in the data, leading to improved sample quality and coherence. Also, it helps to reduce the computational overhead associated with traditional diffusion models (Bao et al., 2023; Fei et al., 2024), and allows to apply diffusion models to larger datasets and more

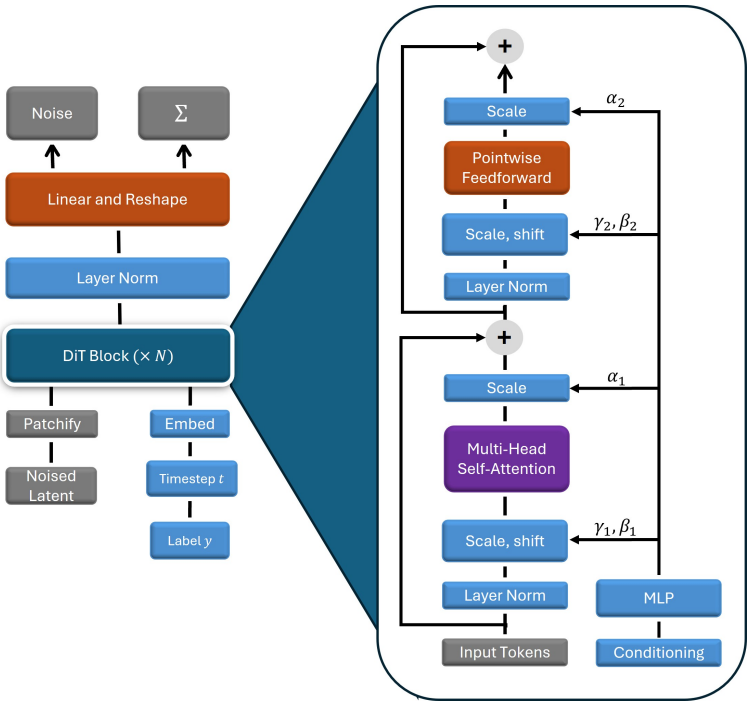


Fig. 2.2 A DiT Architecture with Adaptive layer norm

complex data distributions. The extended use and implementation of transformers has also impulsed the adaptability of diffusion models to suit multiple generative tasks (Feng et al., 2024; Ma et al., 2024; Wu et al., 2024).

2.3 Hypernetworks

Hypernetworks are an emerging Deep Learning architectural paradigm where one neural network (the hypernetwork) generates the parameters of another network (the target network). This approach has shown promising results to enhance the adaptability, flexibility, compression, and performance of DNNs in problems like continual learning, transfer learning, weight pruning, and uncertainty quantification (Chauhan et al., 2023; Li et al., 2020; Volk et al., 2022; Von Oswald et al., 2019).

Consider a dataset $D = \{X, Y\}$ associated with a task T. In the classical Deep Learning framework the learnable parameters θ_F of a DNN $F(X; \theta_F)$ are obtained by solving the optimization problem in Equation 2.18 (Chauhan et al., 2023).

The searching for the optimal configuration is performed within large search spaces formed by millions of potential parameters. The input samples $x \in X$ are passed through the

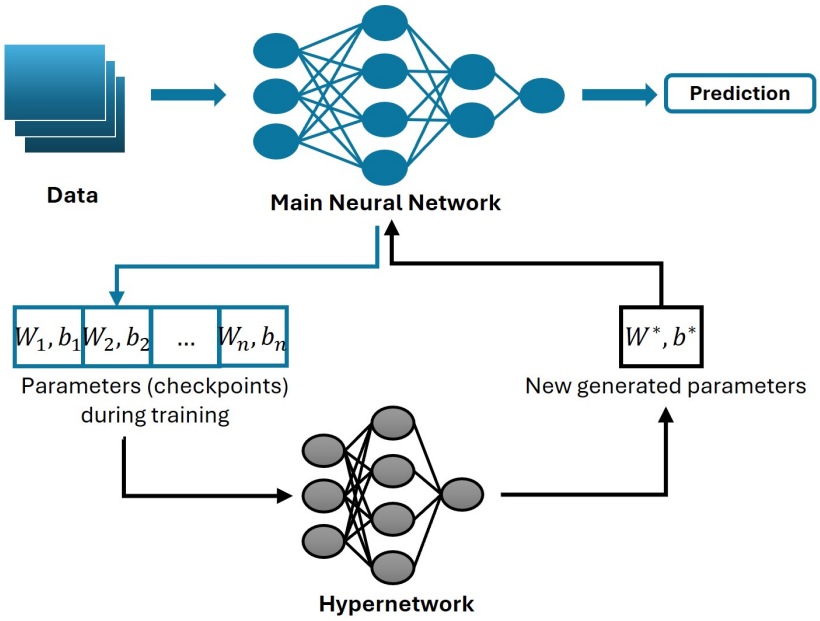


Fig. 2.3 Hypernetwork Framework

layers of F to obtain predictions $y^* \in Y^*$, later compared with the true labels $y \in Y$ using a loss function $L(Y, Y^*)$, which is optimized by updating parameters until convergence.

$$\min_{\phi} F(X; \theta_F) \tag{2.18}$$

Instead of directly optimizing the parameters of the main network, the hypernetwork framework (Figure 2.4) uses a separate network to learn to generate the parameters for the main network, often resulting in a smaller search space and making the optimization more efficient. Both networks are trained in an end-to-end differentiable manner (Chauhan et al., 2023; Ha et al., 2016).

Definition 8 (Hypernetwork) *A Neural Network $G(C; \theta_G)$ is called a hypernetwork with learnable parameters θ_G and context input C if its output are parameters for a second Neural Network $F(X; \theta_F)$ that solves a task T with associated data $D = \{X, Y\}$, i.e. $\theta_F = G(C; \theta_G)$.*

The context input C for the hypernetwork contains information about the structure of the parameters of the main network that enables learning to generate its parameters.

During the forward pass at training time, the parameters θ are generated by passing C through G, which are then inputted to F to process $x \in X$ and obtain predictions $y^* \in Y^*$. The loss $L(Y, Y^*)$ is then computed and during the backward pass, the error is back-propagated through G with the gradients of L computed with respect to θ_G. Consequently, θ_G are

optimized to generate the θ_F that best solves task T. This introduces the optimization problem in Equation 2.19.

At test time, new parameters θ_F^* can be sampled from the optimized hypernetwork and used to make predictions with $F(X;\theta_F^*)$ on the test data.

$$\min_{\phi} F(X; G(C;\phi)) \tag{2.19}$$

Traditional DNN parameters are fixed after training, but hypernetworks offer adaptability by generating new parameters for networks with dynamic architectures. They can be conditioned on input data for specific requirements or on tasks to generate parameters for multiple related tasks (Ha et al., 2016; Mahabadi et al., 2021; Oh and Peng, 2022). This ability to sample diverse parameters also allows for uncertainty-aware models by creating ensembles of networks, providing a way to quantify prediction uncertainty (Kristiadi et al., 2019).

However, given the early stage of this new paradigm, challenges remain for its broader adoption. The size and complexity of hypernetworks scale with the target network, making them less feasible for large models unless optimized strategies are used. Additionally, for models with smaller parameter spaces than the hypernetwork, training can be computationally expensive. Reducing hypernetworks to fit various network sizes is still an open problem (Chauhan et al., 2023).

Layer-wise architecture considerations also pose challenges in finding suitable initial values for hypernetwork parameters, affecting the effectiveness of the generated parameters (Li et al., 2020, 2021). Furthermore, there's a need for a robust framework to assess the validity, variety, and effectiveness of the generated parameters, ensuring they do not simply memorize data (Erkoç et al., 2023; Peebles et al., 2022).

This emerging paradigm shows promise but requires further exploration, particularly in understanding their representational capacity, learning dynamics, generalization capabilities, and the interpretability of task-specific parameter generation (Chauhan et al., 2023). Addressing these challenges could lead to the development of more adaptable models for diverse applications.

2.3.1 Diffusion Hypernetworks

Neural network training typically involves transforming randomly initialized parameters (essentially noise) into a structured set that form a specific distribution, resembling the generative problems solved by diffusion models. This similarity, especially in SGD-based training, has led to the use of diffusion as a backbone for hypernetworks, approach we call here diffusion hypernetworks.

In experiments, Multi-Layer Perceptrons (MLPs) are commonly chosen as the main network due to their simplicity, efficiency, and ease of dataset generation for specific applications. The process usually begins by flattening MLP parameters into one-dimensional vectors, which are then processed by a diffusion model. The diffusion model learns the distribution of the parameter space and predicts denoised parameters, handling varying-dimensional data in an agnostic manner (Erkoç et al., 2023; Wang et al., 2024a).

Gaussian noise is iteratively added to each vector θ during diffusion modeling. The noisy vector and embedded t serve as inputs for the model, which learns to predict denoised parameters. Some methods also incorporate conditional generation, using context such as performance metrics to guide the diffusion model. MSE is often used to train the model by comparing denoised and input parameters.

The first notable application of this type of model was introduced by Peebles et al. (2022). This work generated parameters to achieve specific prompted performance metrics from a dataset created by saving checkpoints from multiple MLP training runs. They used DiT and DDPM to develop G.pt, a learned optimizer capable of updating parameters (starting from potentially random noise) to meet target performance levels.

G.pt demonstrates broad generative properties by sampling parameters for various performance levels, unlike recent approaches that focus solely on generating high-performance parameters. However, it is less precise in achieving target metrics, requires more data, and involves considerable training overhead compared to other methods (Wang et al., 2024a).

Similarly, Erkoç et al. (2023) employed a DiT to model the distribution of parameters, saving optimized parameters for multiple MLPs, which were then encoded and processed by the DiT to unconditionally generate new parameters through DDPM.

Wang et al. (2024a) addressed the computational cost of learning directly from the parameter space by incorporating an autoencoder to project the parameters into a smaller latent space, enabling more efficient learning. They used a U-Net for the diffusion process and created a dataset by training a network to a high-performance state, then saving subsequent iterations. This approach effectively introduces a soft bias that help the model towards generating high-performing parameters, although it still faces limitations with diverse parameter distributions.

Recently, Li et al. (2024) expanded Peebles et al. (2022) ideas by proposing a method for text-to-model generation, allowing personalized models to be generated directly from text prompts. For example, within a large dataset like CIFAR-100, a user might need a network for a specific subset of classes, by inputting text prompt the proposed model can generate the required customized network.

These works show that applying diffusion models as hypernetworks for parameter generation leverages their generative strengths, providing an efficient framework where the number of forward passes depends solely on the diffusion process length, regardless of data dimensionality. While the use of diffusion models as hypernetworks is still emerging with limited studies, the approach shows significant promise.

2.3.2 Learning To Learn

Now, we explore deeper G.pt (Peebles et al., 2022), as its capabilities for learning from a varied distribution of parameters makes it a suitable option for our modeling needs.

G.pt allows to generate neural network parameters directly within the parameter space using a learning-to-optimize approach. It functions as a learned optimizer, indirectly solving optimization problems traditionally handled by SGD. It uses DiT as a hypernetwork [3] conditioned on a vector of initial parameters, their associated performance metric, and a target metric value, with the goal of returning updated parameters that meet the target metric.

They explored both Classification and Reinforcement Learning tasks, using metrics like prediction error, testing loss, and reward. For our purposes, we focus on the Classification case, using testing loss as the conditional metric.

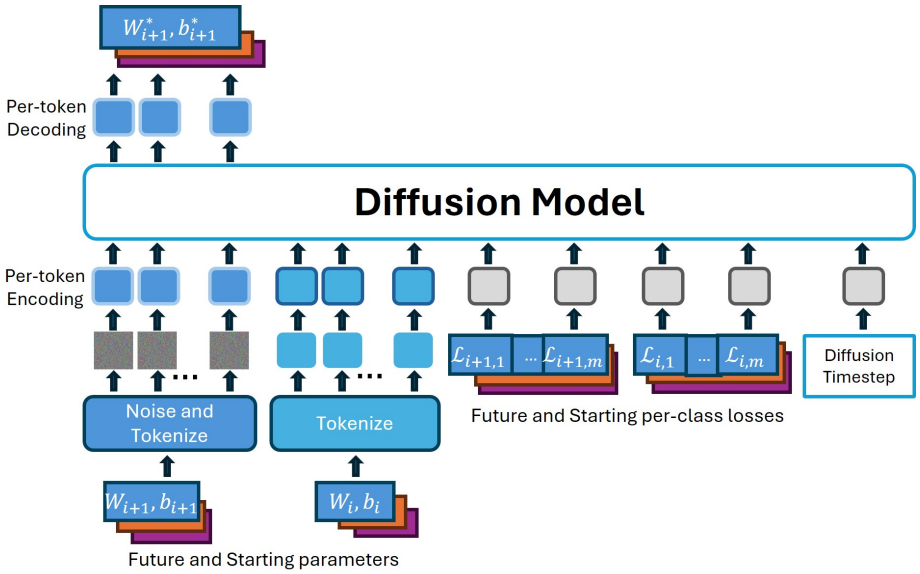


Fig. 2.4 G.pt Architecture

[3] We categorize G.pt as a hypernetwork according to Definition 8

Training Data

To create the training datasets, Peebles et al. (2022) executed multiple training runs of the main network, saving N random checkpoints uniformly, including the initial values before starting the training run. Thus, from each training run $m \in [0,...,M], M \in \mathbb{N}$ a series of parameters $\theta_{m,0}, \theta_{m,1}, ..., \theta_{m,N-1}$ with its corresponding test loss $\mathscr{L}_{m,0}, \mathscr{L}_{m,1}, ..., \mathscr{L}_{m,N-1}$ are saved along the training.

Data augmentation in parameters space is performed using permutation augmentation, which preserves the output of the network (Roeder et al., 2021; Schürholt et al., 2021), ensuring that the parameter level-augmentation $T(\theta)$ is valid, $f_T(\theta)(x) = f_\theta(x)$

They ended up constructing a dataset with 2M checkpoints for MNIST and 11M for CIFAR-10 as underlying tasks, using MLPs and CNNs, respectively.

$$p_G(\theta^* \mid \theta, \mathscr{L}^*, \mathscr{L}) \tag{2.20}$$

Diffusion Transformer Training

To load the data for the training, a saved run is selected uniformly at random, from which a random tuple $(\theta, \mathscr{L}, \theta^*, \mathscr{L}^*)$ is sampled uniformly, with θ always from a step earlier than θ^*, though not necessarily from a consecutive checkpoint. These form the current parameters θ, current loss $\mathscr{L}$, future parameters θ^*, and the corresponding future loss $\mathscr{L}^*$. The generative model aims to predicts the distribution of updated parameters that achieve the desired loss, Equation 3.4, effectively producing a learned optimizer.

The parameters are normalized using the normalization scheme of DALL-E 2 (Ramesh et al., 2022), which scales the data so that the variance of the marginal distribution matches the variance of ImageNet pixels scaled to [-1,1]. This normalization ensures that the forward noising process destroys nearly all signal in θ^*.

As the main neural network F may contain many layers, potentially with different numbers of parameters in each one, both input parameters θ and θ_t^* are tokenized layer-by-layer. This involves flattening them into 1D vectors, the flattened parameter vector of the i-th layer corresponds to the i-th token. Layers with both weight and bias are decomposed into separate tokens. A maximum number of parameters per token is established and if a layer contains more parameters, it is decomposed into multiple tokens, ensuring each partition adheres to this upper bound, which is smaller than the hidden size of the DiT to avoid lossy compression.

The time-step, losses, and the difference between losses are tokenized using a frequency-based encoding scheme. Then, the tokens are linearly projected to the hidden size of the DiT, producing a unique set of weights for each token.

During training, Standard Gaussian noise is added t times to the future θ^* which is passed to the DiT conditioned on the additional inputs. The DiT uses the architecture of GPT-2 without causal masking (Radford et al., 2019), and the final layer is a decoder that linearly projects the i-th token from the DiT's hidden size back to its original size to produce the future denoised parameter vector θ_t^*.

A residual connection is added at the end, such that the parameter update $\theta^* - \theta$ is predicted instead of directly predicting θ^*. The objective metric for the training is the MSE between the generated parameters and the actual parameters.

$$\mathfrak{L}(G) = \mathbb{E}(\|\theta^* - G(\theta_t^*, \theta, \mathscr{L}^*, \mathscr{L}, t)\|_2^2) \tag{2.21}$$

The model is trained with AdamW (Loshchilov and Hutter, 2017), maintaining an exponential moving average (EMA) of G.pt parameters throughout training. Learned positional embeddings initialized to zero are used across all tokens.

At inference time, the model can take a starting (potentially random) parameters θ with its corresponding starting loss $\mathscr{L}$, and sample updated parameters that approximate a desired loss prompted by the user through denoising an input Gausian noise θ^* using DDPM with fixed variances (Ho et al., 2020).

Evaluation Procedure

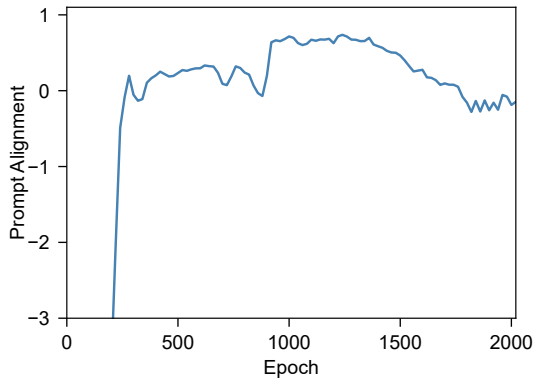
During evaluation, the model is prompted with a value close to the best loss found in the training dataset. Peebles et al. (2022) noted that using a value slightly above or below the best loss can sometimes yield better results for certain tasks.

The model is evaluated using a prompt alignment metric, defined as the R^2 score between the obtained loss and the target loss (Equation 2.22), averaged over the batch size. This score is calculated across 20 regularly-sampled prompts and averaged over multiple sampled networks, using randomly-initialized input parameters to evaluate generalization capabilities.

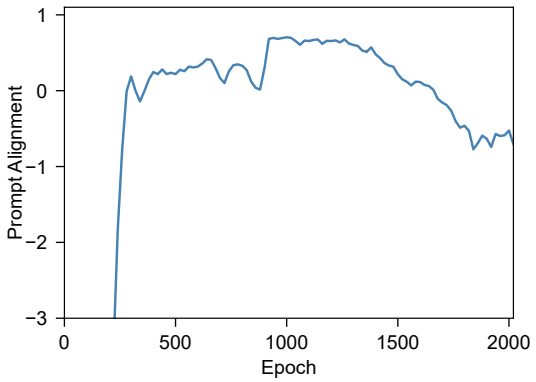
The obtained losses are also plotted against the desired losses to visualize their alignment, including a lower bound (usually set to be the best loss during the data generation) and the identity function.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\mathscr{L}_i^* - \hat{\mathscr{L}}_i^*)^2}{\sum_{i=1}^{n}(\mathscr{L}_i^* - \text{avg})^2 + \varepsilon} \tag{2.22}$$

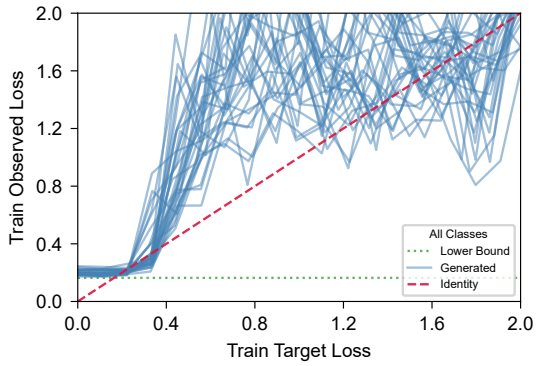
where ε is a small constant to avoid division by zero, and avg is the mean of the targets:



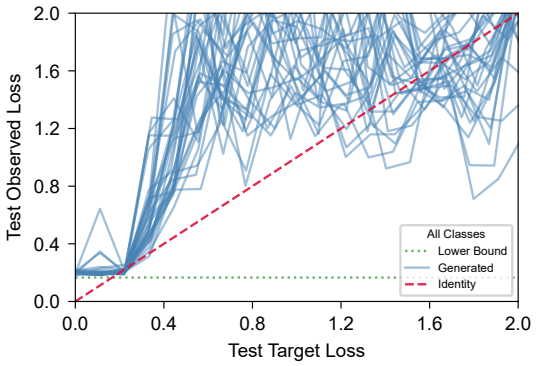
(a) Train Prompt Alignment



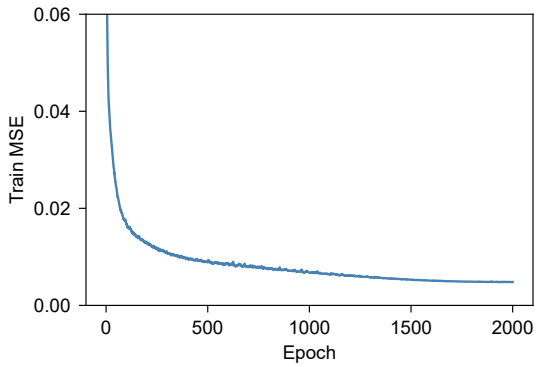
(b) Test Prompt Alignment



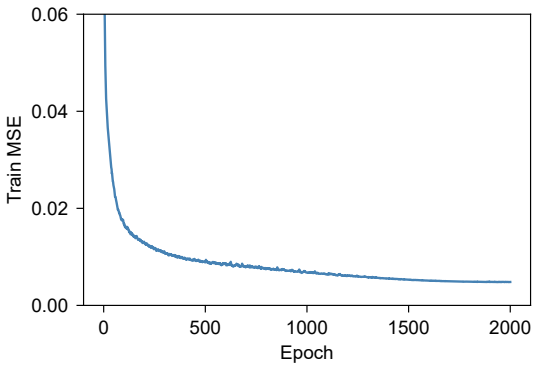
(c) Train Observed vs Target Loss



(d) Test Observed vs Target Loss



(e) Train Learning Curve



(f) Test Learning Curve

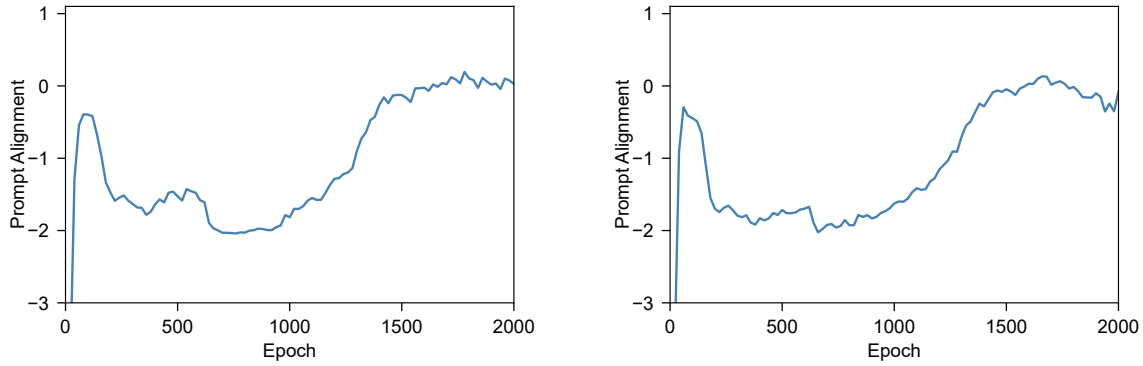
Fig. 2.5 G.pt Training Results.

$$\text{avg} = \frac{1}{n}\sum_{i=1}^{n} \mathscr{L}_i^* \tag{2.23}$$

As shown in Figure 2.5, G.pt's learning curves decrease and converge, with prompt alignments improving over epochs, albeit with some fluctuations. This indicates a generally stable, though slightly variable, alignment between prompts and targets. While the model does not exactly achieve the desired metrics, as noted by Wang et al. (2024a), it effectively generates parameter updates that correlate well with the target losses. However, it struggles to extrapolate to losses beyond the training dataset's range (Peebles et al., 2022).

Experimental Observations

Our experiments suggest that while G.pt can learn to generate parameters using loss, prediction error, or accuracy, it is generally easier to learn with loss. This is likely due to loss values typically spanning a narrower range, making them more sensitive to parameter changes, compared to the broader range covered by accuracy and prediction error.



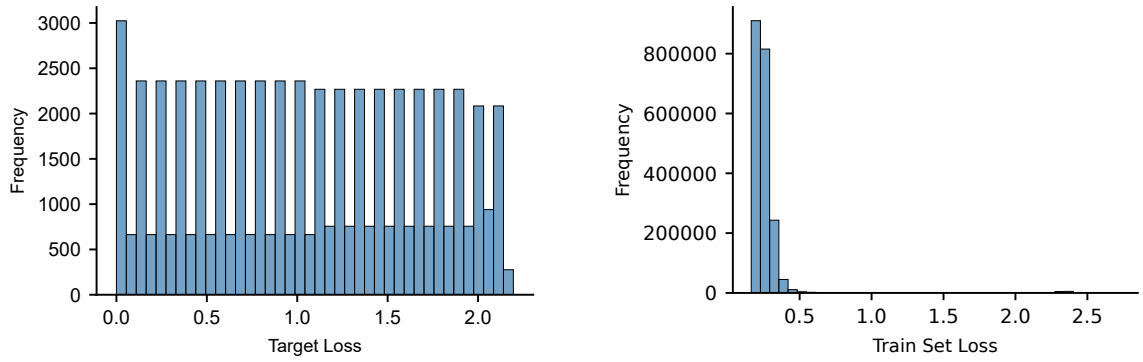
(a) Train (b) Test

Fig. 2.6 Evolution of G.pt Prompt Alignment with Error as metric.

Even when in CIFAR-10 experimental results of Peebles et al. (2022) using prediction error as a metric was beneficial, this is likely due to the narrow range of prediction error values for that specific experiment. When comparing performance across different metrics on the same set, as shown in Figure 2.6, the instances with more samples and a narrower value range, typically loss, performed better

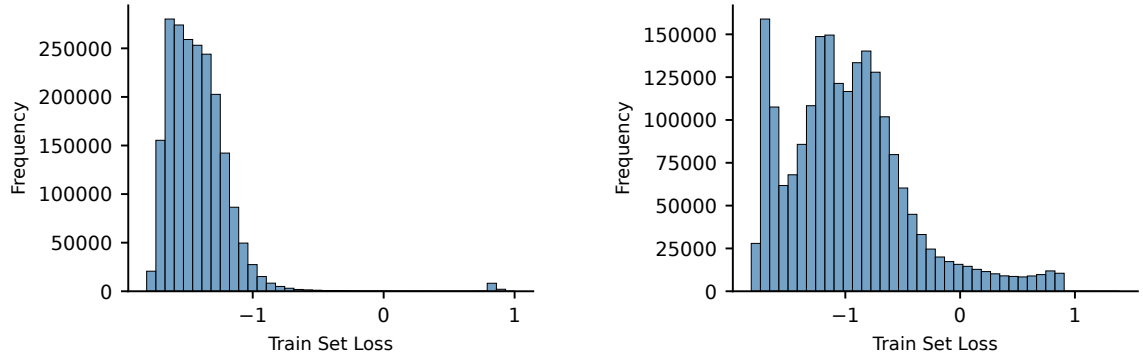
Additionally, 2.7 highlights a disparity between the distribution of losses found during testing and those in the training datasets used to produce the results in Figure 2.5. During testing, metric values tend to be more uniformly distributed, thus, effective learning requires varied examples across the entire testing range. By resampling the training set to cover a

wider range of losses (Figure 2.8), G.pt's performance improves significantly, as shown in Figure 2.9.



(a) Target Losses (b) Losses in Train Set

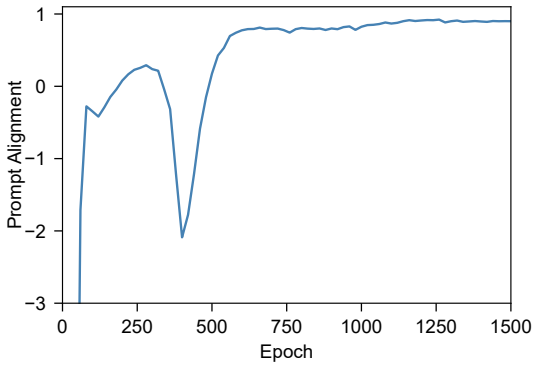
Fig. 2.7 Comparison of Distribution of Test Losses and Losses in Train Set for G.pt.



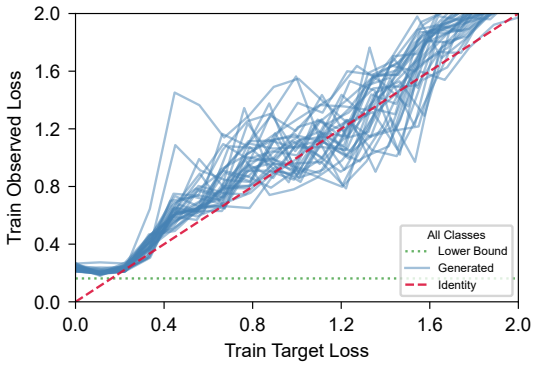
(a) Losses in Train Set (b) Losses in Resampled Train Set

Fig. 2.8 Comparison of Distribution of Losses in Train Set and Resampled Train Set for G.pt in log-scale.

Our findings highlight that G.pt, while capable of generating neural network parameters for diverse losses, relies heavily on well-composed training data to perform effectively within the target metric space. In the next chapter, we build on this approach and the concepts reviewed here to expand the application of diffusion hypernetworks, particularly for Machine Unlearning.



(a) Prompt Alignment - Train



(b) Observed vs Target Loss - Train

Fig. 2.9 Behaviour of G.pt with re-balanced loss data.

Chapter 3

Methodology

This chapter introduces the core contributions of this dissertation by introducing the HyperForget framework and Diffusion HyperForget Networks, which utilize diffusion hypernetworks for Machine Unlearning. The chapter is structured to build a comprehensive understanding of the framework and its potential benefits.

Section 3.1 outlines how hypernetworks can generate parameters tailored for unlearning, particularly in classification tasks, using diffusion models. We present two POCs demonstrating their potential application in Machine Unlearning.

Sections 3.3 and 3.4 delve into the architecture, methodology, and design of each model, followed by evaluations of their parameter generation capabilities.

Finally, Section 3.5 assesses the models' effectiveness in achieving targeted unlearning, discussing their strengths, limitations, and potential applications.

3.1 HyperForget

Inspired by the promising results of hypernetworks and the need for pre-trained unlearning methods, we propose HyperForget, a novel approach that uses hypernetworks to sample neural networks meeting specific forgetting conditions—such as high performance on the retain set and low performance on the forget set. To the best of our knowledge, this is the first application of hypernetworks for Machine Unlearning.

Traditional DNNs have fixed parameters and architectures post-training, requiring complete retraining for any modifications. This rigidity makes them unsuitable for scenarios that demand dynamic adjustments, like unlearning. By using hypernetworks, HyperForget allows for dynamic parameter adaptation to forget specific data while preserving key capabilities. This flexibility makes it well-suited for various unlearning tasks, particularly in scenarios with frequent forget requests or the need for rapid adaptation.

Following Definitions 2, 3, and 5, in the HyperForget framework a hypernetwork, named HyperForget Network (HyFo), is trained to generate parameters that reduce performance on a specific forget set D_f, while preserving performance on the retain set D_r for a given task T. The goal is to reconstruct a network's knowledge to maintain effectiveness on D_r, while forgetting D_f, ideally mimicking the behavior of a model retrained without D_f.

Additionally, diffusion models, with their ability to model complex distributions and transitions from noise to structured data, are relevant for our objectives. Intrroducing them in the HyperForget allows the controlled removal of specific data influences while preserving the model's overall integrity, providing a structured approach to gradual unlearning.

Consider a classification task T with a dataset $D = \{X, Y\}$, each example $x_i \in X$ with an associated label $y_i \in \{1, ..., m\}, m \in \mathbb{N}$, drawn from an unknown distribution P. And that a model trained to solve task T is then requested to forget a specific class or subset of classes, preferably without retraining. This means that the forget set $D_f \subset D$ contains data with a specific class label.

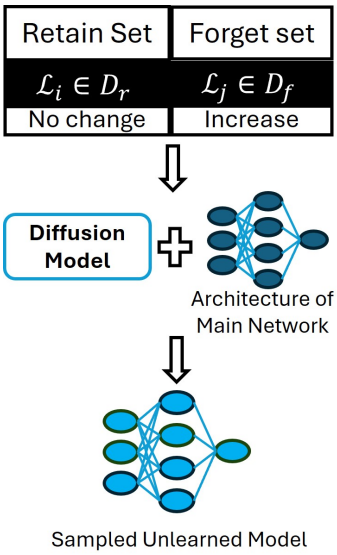


Fig. 3.1 Diffusion HyperForget Process

As depicted in 3.1, for this forgetting problem we can employ a diffusion model conditioned on the specific class performance metrics, such as class losses,[1] to construct a Diffusion HyperForget Network (DiHyFo) capable of generating parameters that yield high-performance on D_r while obtaining low-performance on D_f, effectively forgetting the specified classes.

[1] Given our discussion on Section 2.3.2 we use losses, but other performance metric can be employed.

We explore two mechanisms for constructing the DiHyFo model, depicted in Figure 3.2. The first model, DiHyFo-1, is built on the learning-to-learn framework of Peebles et al. (2022), extending it to generate parameters conditioned on class losses and to be able to learn-to-forget, i.e. learning to de-optimize. Conversely, DiHyFo-2 uses a diffusion model directly conditioned on the desired class losses.

(a) DiHyFo-1

(b) DiHyFo-2

Fig. 3.2 Two implementations of Diffusion HyperForget Networks.

Notably, unlearned models obtained using HyperForget can be interpreted and evaluated using the two interpretations of the definition of unlearning, either as an approximation to exact unlearning on the parameter space or in the output space.

Following Section 2.1.6, the unlearned models are evaluated by comparing them with a model retrained from scratch without the unlearning targets. While a retrained model should be constructed for each forget set, DiHyFo models can sample an unlearned models with any desired forgetting configuration.

3.2 Evaluation Procedure and Metrics

Each model must be evaluated on two key aspects. First, their generative properties as diffusion hypernetworks need to be assessed. This involves verifying whether the models can generate appropriate parameters for the main network, enabling it to approximate targeted performance levels on the classification task for each class. This helps identify limitations in the models' generative capabilities and areas for improvement.

Both DiHyFos use MSE as a training metric, so we analyze learning curves to track MSE evolution, understand the model's learning behavior and convergence, and identify issues like overfitting or underfitting.

To evaluate whether the generated parameters effectively approximate prompted loss values, we use the prompt alignment metric defined in Section 2.3.2, which is based on an R^2 score (see Equation 2.22). A score close to 1 indicates that the obtained loss closely aligns with the target loss. Negative values suggest worse alignment than the mean of the target values. We compute this score separately for each class to assess the model's ability to control losses across different classes simultaneously.

As described in Section 2.3.2, the prompt alignment metric is computed over 20 regularly-sampled prompts and averaged over multiple neural networks sampled with each DiHyFo, using randomly-initialized input parameters. This metric is effective for evaluating conditional generative tasks, such as those in the learning-to-learn framework (Peebles et al. (2022)), as it measures how well the model aligns the observed losses with the target losses in terms of both direction and magnitude.

However, the metric is typically measured in an increasing loss fashion, while our training datasets exhibit bidirectional loss movements, which can affect how the model learns to align the losses. In our experiments, we found that observed losses generally aligned correctly with the direction of the prompted losses, but not in magnitude. Thus, we compute the correlation between observed and target losses to assess if the model is accurately tracking the target, providing a complementary metric to the prompt alignment score.

We also plot the observed versus target losses along the identity line (indicating perfect alignment) and include a lower performance bound—often set to the median or average of losses from checkpoint collection—as a reference for high performance.

The second aspect to evaluate is the DiHyFo models' ability to sample unlearned networks. Following Section 2.1.6, we assess the unlearning properties of networks sampled with each DiHyFo by comparing them to a main network retrained from scratch without the forget targets. As previously discussed, this retrained model [2] is typically considered the gold standard for unlearning evaluation. The closer the behavior of the sampled unlearned model is to the retrained model, the more effective the unlearning (see Section 2.1.2). This evaluation helps determine the usefulness of each DiHyFo for unlearning applications, their limitations, and areas for improvement.

The comparison begins by individually computing the accuracy of each unlearned model and the retrained model on the retain, forget, and complete test sets. Additionally, we calculate the MIA score for each model to assesses the likelihood that an adversary could infer information about the forget set from the unlearned model. Ideally, the unlearned model should have MIA scores similar to or lower than those of the retrained model.

[2]This is called retrained model because the in the Machine Unlearning framework it is assumed that we have an already trained network, which is later required to forget instances of its training data, see Section 2.1 and 2.1.6.

To compute the MIA score, we follow the implementation of Chundawat et al. (2023a), as in Figure 3.3. The model being evaluated generates prediction probabilities for the retain and test sets. These probabilities are used to calculate entropy for each set, and the resulting entropies serve as features for a logistic regression, with labels indicating the origin dataset of each instance. The logistic regression is trained and then used to predict the membership status of instances in the forget set. The average of these predictions provides the MIA score, reflecting how effectively it can infer whether the forget data was partof the training set of the model being evaluated.

Input: Model F, Datasets D_r, and D (retain and test sets)
$P_r \leftarrow F(D_r)$
$P_t \leftarrow F(D)$
$X_r \leftarrow [\mathscr{H}(P_r), \mathscr{H}(P_t)]$ ▷ $\mathscr{H}$ is the entropy measure
$Y_r \leftarrow [1 \text{ (retain)}, 0 \text{ (test)}]$
LR $\leftarrow$ Train Logistic Regression with (X_r, Y_r)
$\hat{Y}_f \leftarrow \text{LR}(\mathscr{H}(P_r))$
MIA $\leftarrow \frac{1}{n_f} \sum_{i=1}^{n_f} \hat{Y}_{fi}$
Output: MIA

Fig. 3.3 Pseudocode for computing MIA score.

To compare each model's output space against the retrained model (Section 5), we measure the overlap in their predictions on each set. For parameter space comparison, we calculate the Activation Distance between the sampled models and the retrained model. Following Equation 3.1, the output logits of both models are passed through a softmax function to generate probability distributions. The difference between these distributions is computed for each input, followed by calculating and averaging the L_2 norm. A lower score indicates greater similarity in behavior between the models for the same inputs.

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$\text{Activation Distance} = \sqrt{\sum_{i=1}^{K} \left(\text{softmax}(\mathbf{z}_{\text{model1}})_i - \text{softmax}(\mathbf{z}_{\text{model2}})_i\right)^2} \tag{3.1}$$

Furthermore, following Foster et al. (2024a), we compute an unlearning score φ based on the Jensen-Shannon Divergence (JSD), which is a symmetric measure constructed using the Kullback–Leibler divergence (KL) between two probability distributions, P and Q. The JDS is computed between the softmax outputs of each pair of models being compared across the entire dataset, Equation . A value close to 1 indicates that both models are very similar.

$$\begin{aligned} \mathrm{JSD}(P \parallel Q) &= \frac{1}{2}\mathrm{KL}(P \parallel M) + \frac{1}{2}\mathrm{KL}(Q \parallel M) \\ M &= \frac{P+Q}{2} \end{aligned} \tag{3.2}$$

$$\varphi = 1 - \mathrm{JSD}(\mathrm{softmax}(\text{Model 1}) \parallel \mathrm{softmax}(\text{Model 2})) \tag{3.3}$$

These metrics provide a thorough assessment of each DiHyFo model's ability to generate loss-conditional parameters and their suitability for Machine Unlearning tasks, which remains our primary goal.

3.3 DiHyFo-1

DiHyFo-1 is a hypernetwork built on a pre-trained DiT, following a similar methodology to Section 2.2. During training, DiHyFo-1 takes in current parameters θ, future parameters θ^*, and their associated class losses $\mathscr{L}_1, ..., \mathscr{L}_m$ and $\mathscr{L}_1^*, ..., \mathscr{L}_m^*$ for the classification task T. These inputs are tokenized along with the time-step, and Standard Gaussian noise is added to the future parameters at each time-step. This allows the model to learn how to denoise them, generating neural network parameters directly in the parameter space with the desired loss properties for each class.

DiHyFo-1's goal is to predict the distribution of updated parameters that achieve the target loss for each class, as detailed in Equation 3.4. The model's architecture is illustrated in Figure 3.4.

$$p_G(\theta^* \mid \theta, \mathscr{L}_1^*, ..., \mathscr{L}_m^*, \mathscr{L}_1, ..., \mathscr{L}_m) \tag{3.4}$$

At inference, the model receives a set of current parameters (which may be initialized randomly) along with their associated current and target losses for each class, and returns updated parameters with the requested loss properties for each class.

For our forgetting goal, specific data considerations are crucial. First, as noted in Section 2.3.2, this architecture performs better when provided with ample examples that capture the full range of loss evolution.

Additionally, the architecture can learn the parameter evolution for individual classes (see Appendix A). However, unlike Peebles et al. (2022), which tackles an unconstrained optimization problem typically addressed with SGD, DiHyFo-1 focuses on a constrained optimization problem. Since parameter adjustments impact multiple classes, the challenge

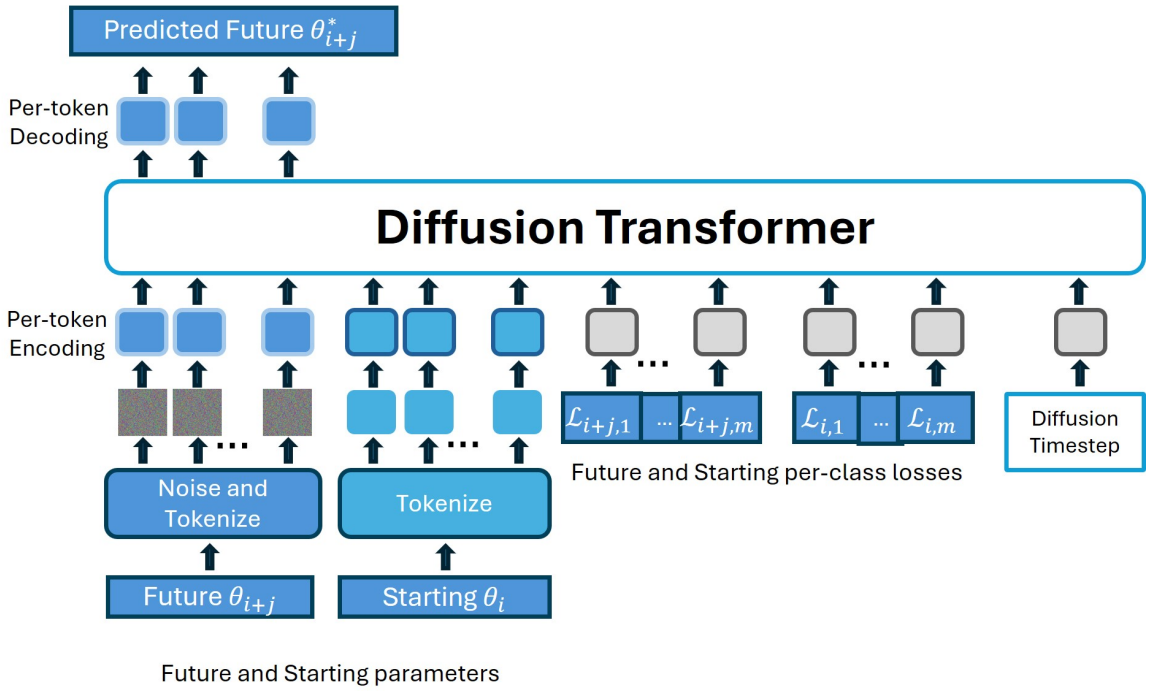


Fig. 3.4 DiHyFo-1 Architecture

is to control loss evolution so that the model can increase losses for some classes while minimizing them for others. This setup presents the optimization problem described in Equation 3.5.

$$\max_{\theta} \left[\sum_{c \in \mathscr{C}_{\text{increase}}} \mathbb{E}_{(x,y)\sim\mathscr{D}} \left[\mathscr{L}_c(y,\hat{y}) \right] - \lambda \sum_{c \in \mathscr{C}_{\text{minimize}}} \mathbb{E}_{(x,y)\sim\mathscr{D}} \left[\mathscr{L}_c(y,\hat{y}) \right] \right] \quad \text{subject to} \quad \theta \in \Theta \tag{3.5}$$

The constrained optimization problem defined here aligns directly with the Machine Unlearning objective in Definition 3. The first term aims to maximize loss on the classes to forget, reducing their influence, while the second term minimizes loss for the classes to retain, preserving model performance on those data points.

To create the checkpoint dataset, it's necessary to track the evolution of class-level losses in a constrained manner. However, since SGD is inherently an unconstrained optimizer, we either need to use a constrained optimizer or apply heuristic constraints to collect checkpoints from the main network. As constrained optimizers are uncommon in neural network training, we opt for heuristic strategies to collect checkpoints, enabling DiHyFo-1 to learn to generate parameters sensitive to class losses.

On the other hand, for forgetting DiHyFo-1 needs to handle bidirectional movements within the loss range—learning both to optimize and to de-optimize, i.e., to forget. However, since the underlying diffusion model naturally behaves as an optimizer, it tends to decrease losses regardless of the prompt. To enable DiHyFo-1 to learn to forget, we provide examples of movements that increase class losses, allowing it to replicate this process. This can be achieved by either reversing the order of checkpoint loading during DiT training or by saving checkpoints from a process where losses are intentionally increased. For our POC, we chose the latter approach, as it directly demonstrates an actual forgetting process.

With these considerations, DiHyFo-1 is designed to learn both learning and forgetting processes, enabling it to generate unlearned models with various configurations. Now, we explore our strategies for checkpoint collection and training.

3.3.1 Training Dataset

To construct the training dataset for DiHyFo-1, we collect checkpoints from multiple training runs of the main network, saving parameters along with their corresponding class losses. The process focuses on capturing a diverse range of loss values across different classes.

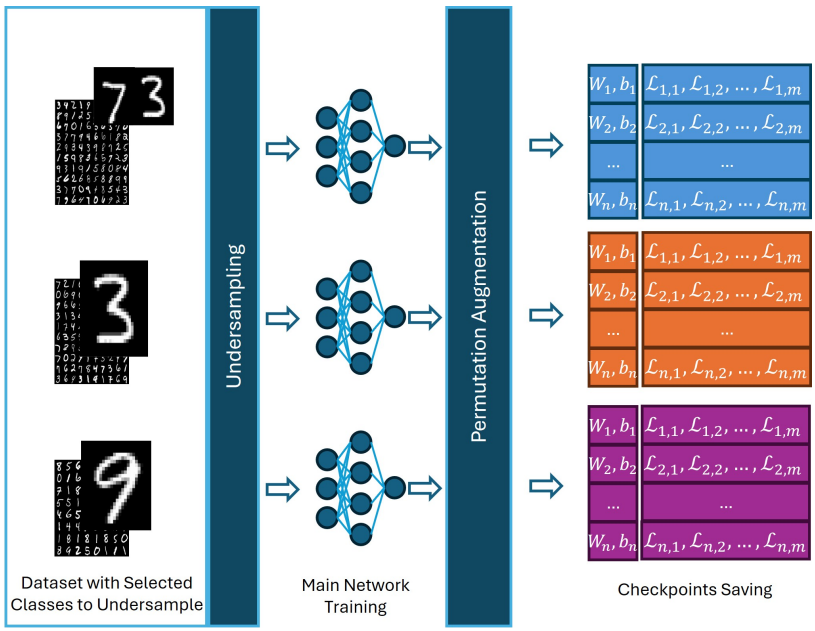


Fig. 3.5 Checkpoints collection for the Optimization Process.

For our experiments, we use MNIST classification with MLPs as the main network. Since a simple MLP achieves good results on MNIST early in training, we randomly select a subset of classes to undersample in each run. This approach helps capture a broader range of class

loss values and behaviors. The MLPs are trained on the dataset, with checkpoints randomly selected and evaluated for potential saving. As in Section 2.3.2, permutation augmentation is applied. The process is illustrated in Figure 3.5.

To collect examples from the forgetting process, we follow a similar training procedure. At a certain point during training, we randomly delete a selection of classes to capture the associated increase in losses. This is similar to forgetting by fine-tuning the main network without the classes to be forgotten 3.6.

We now outline the specific criteria for saving checkpoints, focusing on collecting examples that reflect our target constrained optimization problem. These examples include parameters that perform well for some classes while underperforming for others.

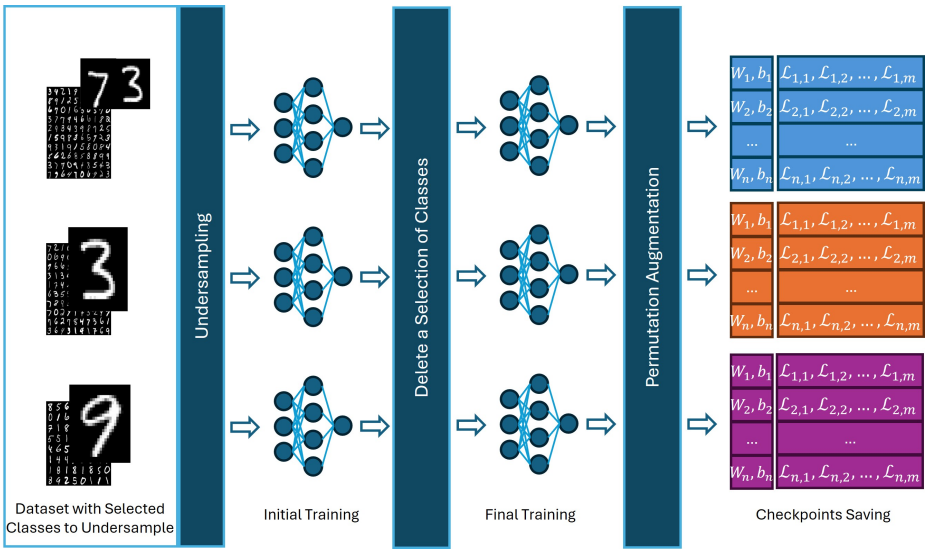


Fig. 3.6 Checkpoints collection for De-optimization Process.

Initially, we create bins corresponding to different loss levels. This allows us to save checkpoints where some classes have high losses and others have low losses. To prevent over-collecting certain loss levels, we set a maximum number of examples for each bin. We prioritize collecting examples from less frequent loss levels to ensure a balanced dataset. [3] However, the wide range of possible loss values across multiple classes leads to a combinatorial explosion of potential loss-level combinations, limiting the practicality of this approach to simpler classification tasks.

For a first POC, we simplify the scenario by focusing on a classification task with a small number of classes, m. We designate a subset of classes, r, as pivots ($r < m$), and only save checkpoints when the model achieves high performance on these pivot classes, using

[3] We experimented with probability decay/increase functions for random iteration saving, but found binning more effective for capturing a broad range of losses

a threshold γ for accuracy or loss. Lowering this threshold increases variability in pivots and expands the combinatorial possibilities. The remaining $m-r$ classes are allowed to vary across the full range of possible loss values. In this setup, forget targets can be any subset of the $m-r$ classes, while the retain set must always includes the pivot classes.

During each training run, a random iteration is reviewed. If the pivots' performance exceeds γ and the corresponding bin isn't full, the checkpoint is saved. This strategy simplifies the problem by focusing on generating parameters for arbitrary losses in $m-r$ classes while maintaining high performance on the pivots. Figure 3.7 summarizes this checkpoint-saving strategy.

For a second POC, we relax the constraints by increasing the number of classes and using fewer pivots or none at all. Instead of using bins to categorize class-level losses, we implement checkpoint-saving conditions that balance the need for diverse loss examples with computational efficiency.

Given that intermediate loss values are less critical for our forgetting objectives, we prioritize capturing high and low performing parameter updates. We apply diverse undersampling rates and increase the random selection rate during early training epochs, where the model's rapid parameter adjustments lead to varied loss behaviors. In later epochs, where changes are more incremental, the selection rate is reduced to focus on capturing significant shifts in loss. This approach ensures our dataset includes key learning moments while avoiding redundancy.

For our first POC we consider a modified MNIST with four classes, MNIST-4, using {0,1} as pivots. We save 150 checkpoints during each training run for 25 epochs of a two-layer MLP with two hidden units and ReLU activation. Our pivot criteria is $\gamma = 80$ on accuracy, and we vary the learning rate, undersampling rates and bins maximums across the training runs. We saved about 1.9M checkpoints in total.

For our second POC we considered MNIST with classes {5-9} as pivots and a two-layer MLP with seven hidden units. We save 200 checkpoints per each training run for a total of 3.7M checkpoints saved. Other details are similar to the first POC.

3.3.2 Training Process

The training process begins with the loading of data by selecting a training run uniformly at random from the dataset. From this run, a random tuple of current, future parameters and their associated losses $(\theta, \mathscr{L}_1, ..., \mathscr{L}_m, \theta^*, \mathscr{L}_1^*, ..., \mathscr{L}_m^*)$ is selected uniformly, with θ always being sampled from an earlier step than θ^*, not necessarily from consecutive checkpoints.

Similar to the process described in Section 2.3.2, the parameters are normalized, and then tokenized layer-by-layer flattening them into 1D vectors, with each layer's flattened

Input: γ (performance threshold), F (main network), D_T (task data), β (checking threshold), $bin_1, ..., bin_r$ (loss bins), $b_1, ..., b_r$ (max bin sizes), Y_p (pivots), max_checkpoints
$D_1 \leftarrow [X_{\text{training}}, Y_{\text{training}}]$ from D
$D_2 \leftarrow [X_{\text{test}}, Y_{\text{test}}]$ from D
$J \leftarrow$ Random sample from unique(Y_{training})
$D_1 \leftarrow$ subsample(D_1, J)
$n_{\text{checkpoints}} \leftarrow 0$
for epoch in N_{epochs} **do**
    if $n_{\text{checkpoints}} <$ max_checkpoints **then**
        training Step $F(D_1)$
        $\mathbf{L_1}, ..., \mathbf{L_m} \leftarrow$ Compute class losses of $F(D_1)$ on D_2
        if $\beta <$ sampled random value **then**
            if every L_j associated with $Y_p > \gamma$ **then**
                if L_u in bin_u, not L_u in Y_p **then**
                    if count$(bin_u) < b_u$ **then**
                        save checkpoint
                        $n_{\text{checkpoints}} \leftarrow n_{\text{checkpoints}} + 1$
                    end if
                end if
            end if
        end if
    end if
end for

Fig. 3.7 Pseudocode for Collecting Checkpoints with Bins.

parameter vector corresponding to a unique token. Layers containing both weight and bias are decomposed into separate tokens, and each token accepts a maximum number of parameters set to be smaller than the DiT hidden size.

The time-step t, input losses, and the differential between the current and future losses $\Delta_{\mathscr{L}_1,\mathscr{L}_1^*}, ..., \Delta_{\mathscr{L}_m,\mathscr{L}_m^*}$ are tokenized using a frequency-based encoder. All the obtained tokens are linearly projected to the transformer hidden size, resulting in a unique set of weights for each token.

During training, Standard Gaussian noise is added to the future parameters at each time step, simulating the forward diffusion process. The noisy parameters are then passed through the DiT, which uses a decoder as final layer that linearly projects each token back to the original size of the corresponding layer's flattened parameter vector to reconstruct the denoised parameter vector θ_t^*. A residual connection to the input θ is added at the end of the process to predict the parameter updates $\theta_t^* - \theta_t$.

The training objective is to minimize the MSE between the original future parameters and the parameters generated by the DiT so that it learns to generate parameters that closely match the original future parameters conditioned on the input losses.

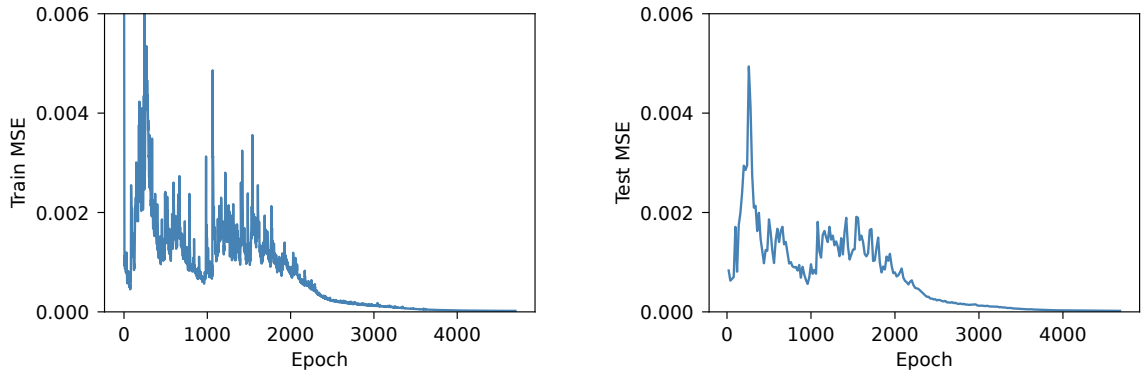
$$\mathfrak{L}(G) = \mathbb{E}(\|\theta^* - G(\theta_t^*, \theta, \mathscr{L}_1^*, ..., \mathscr{L}_m^*, \mathscr{L}_1, ..., \mathscr{L}_m, t)\|_2^2) \tag{3.6}$$

At inference time, the model can take a set of starting parameters θ (which may be initialized randomly) and their corresponding starting losses $\mathscr{L}_1, ..., \mathscr{L}_m$, to sample updated parameters that approximate a set of desired losses $\mathscr{L}_1^*, ..., \mathscr{L}_m^*$ by denoising an input Gaussian noise vector via DDPM with fixed variances.

For the MNIST, the DiT uses a hidden dimension of 1536 with 12 hidden layers with 16 attention heads, trained using AdamW (Loshchilov and Hutter, 2017) with an exponential moving average of the model weights maintained throughout the training process. Additionally, learned positional embeddings, initialized to zero, are applied across all tokens to help the model understand the sequential nature of the data.

3.3.3 Training Evaluation

The learning curves on MNIST-4 depicted in Figure 3.8 show a progressive decrement of the MSE across epochs, albeit with some fluctuations, stabilizing at a low value. This indicates that DiHyFo-1 achieves to capture the underlying patterns in this case. Although the testing MSE exhibits some fluctuations, the behaviour between training and testing is similar, indicating that the model is correctly applying the knowledge gained during training to unseen data.



(a) Training (b) Testing

Fig. 3.8 DiHyFo-1 Learning Curves with MNIST-4.

The prompt alignment and correlation between the target and observed losses are shown in Figure 3.9, and the direct comparison in Figure 3.10. These figures show how the model behaves for both pivot and free classes.

In the early epochs, both metrics fluctuate significantly, reflecting the model's initial exploration of the parameter space as it seeks a stable configuration. As training continues, the metrics stabilize, indicating successful convergence where predictions align closely with the prompts. The early fluctuations highlight the learning process's complexity, while the eventual convergence demonstrates the model's effectiveness in prompt conditioned parameter generation.

The direct comparison plots show that during training for Class 2, the observed losses generally align with the identity line, indicating the model's ability to approximate target losses. However, some deviations suggest difficulties in perfectly matching the desired loss. Similar behavior is seen during testing, with increased variability. These deviations highlight that while the model generalizes well to unseen data, it struggles more with lower target losses.

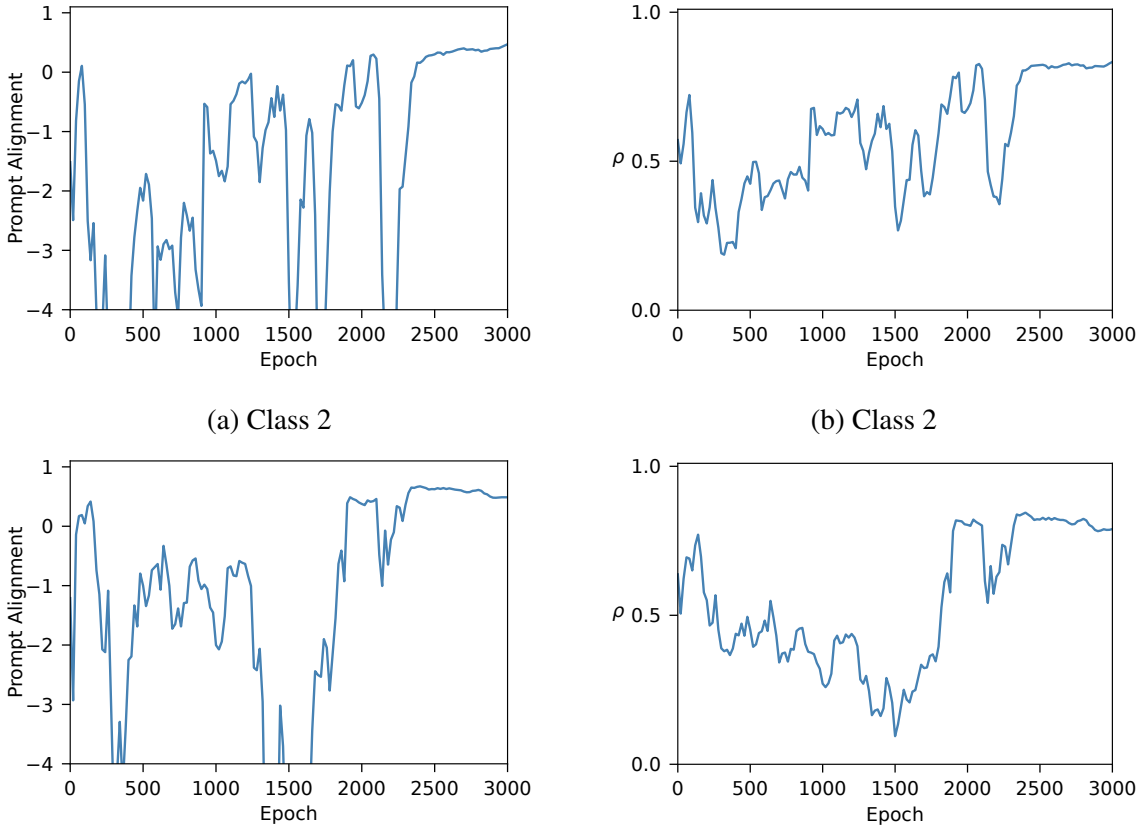
For Class 0, a pivot class, observed losses also track target losses, though with greater divergence and noticeable jumps between levels. This is expected, as training examples primarily represent low losses for these classes.

For MNIST, similar results are observed. The learning curves in Figure 3.11 show a progressive decrease, with some fluctuations, before stabilizing. The sharp drop in MSE suggests that the model quickly adjusts its parameters to fit the training data. Occasional spikes indicate moments of adjustment where errors temporarily increase, likely due to the model balancing the conflicting objectives of forgetting specific classes while retaining others.

Figure 3.12 illustrates how the prompt alignment for two forgettable classes stabilizes, showing that the model effectively adjusts its parameters to align with the forgetting prompts. For the pivot classes, while early fluctuations are more pronounced, the alignment eventually stabilizes, meeting our objectives.

Contrasting with Figure 3.13, the observed losses for forgettable classes generally align well with the target losses, following the identity line's direction with some deviations. The model effectively guides the losses toward the desired targets, though adjusting to the exact magnitude remains challenging. During testing, variability increases compared to training, indicating that while the model generalizes well to new data, the task's complexity leads to some inconsistencies.

For pivot classes, observed losses during training align with target losses in the low-loss range, but with more noticeable deviations, particularly in testing. The model successfully



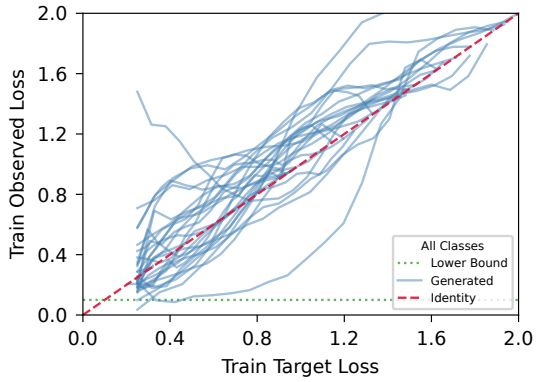
(a) Class 2

(b) Class 2

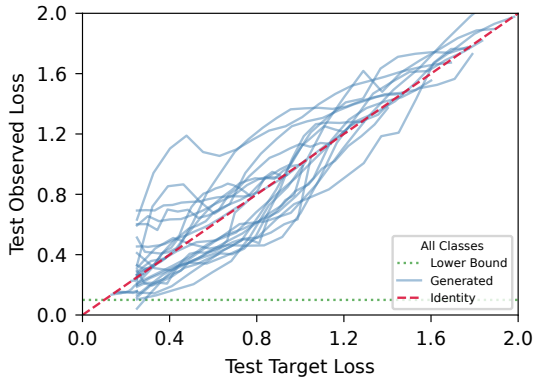
(c) Class 0 (pivot)

(d) Class 0 (pivot)

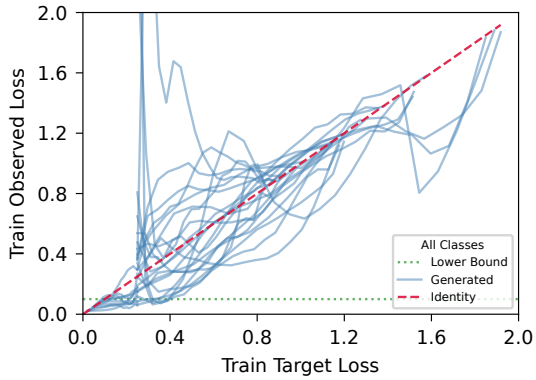
Fig. 3.9 Examples of prompt alignment and correlation obtained by the parameters generated with DiHyFo-1 for MNIST-4.



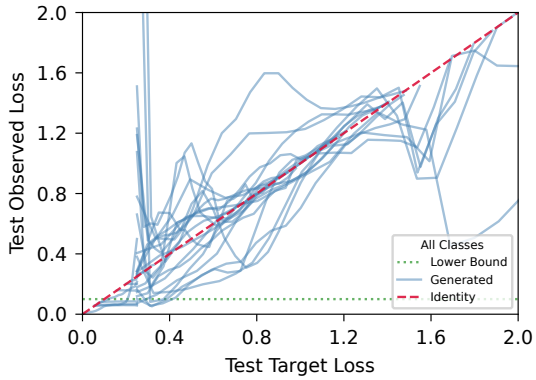
(a) Class 2 - Train



(b) Class 2 - Test

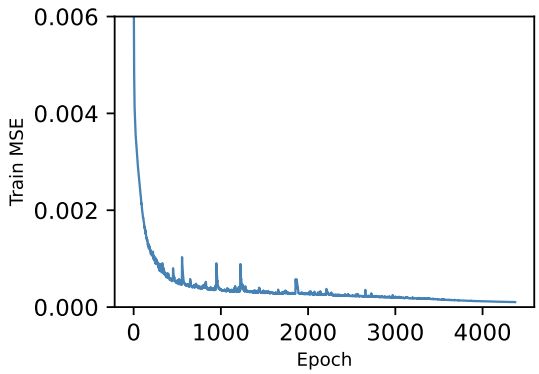


(c) Class 0 (pivot) - Train

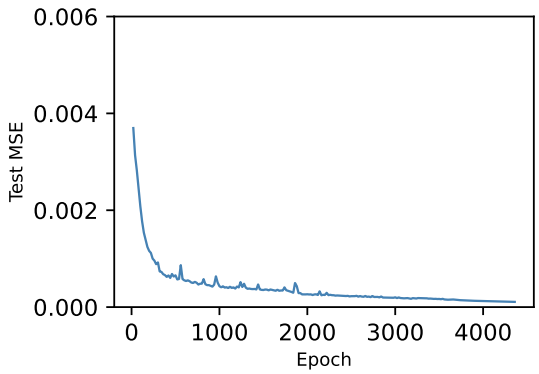


(d) Class 0 (pivot) - Test

Fig. 3.10 Examples of Observed vs Target Losses during training and testing obtained by the parameters generated with DiHyFo-1 for MNIST-4.

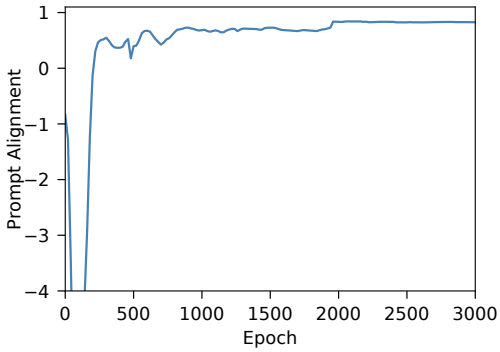


(a) Training

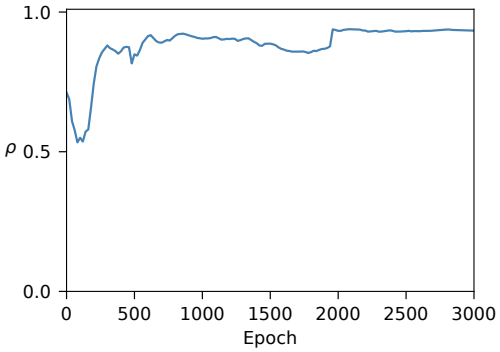


(b) Testing

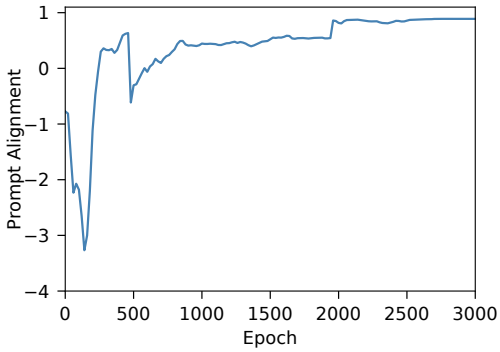
Fig. 3.11 DiHyFo-1 Learning Curves with MNIST.



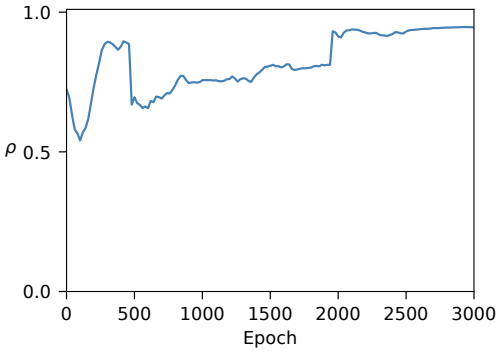
(a) Prompt Alignment on Class 2



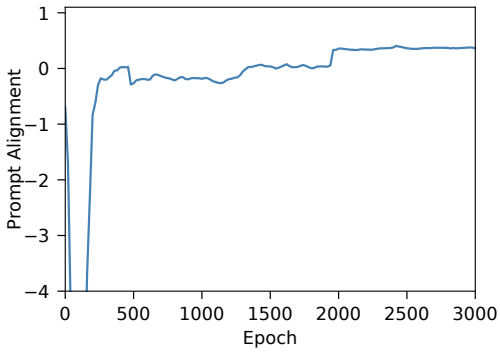
(b) Correlation on Class 2



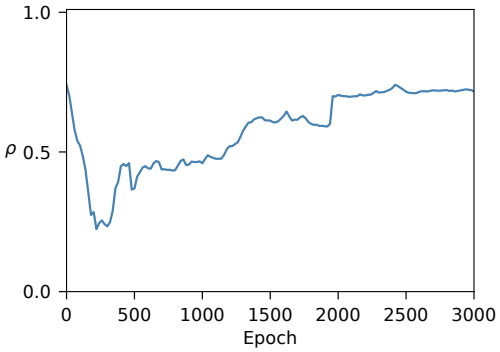
(c) Prompt Alignment on Class 3



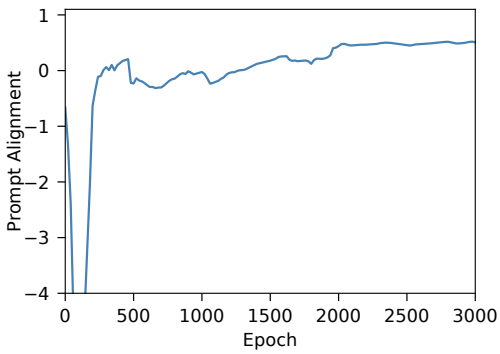
(d) Correlation on Class 3



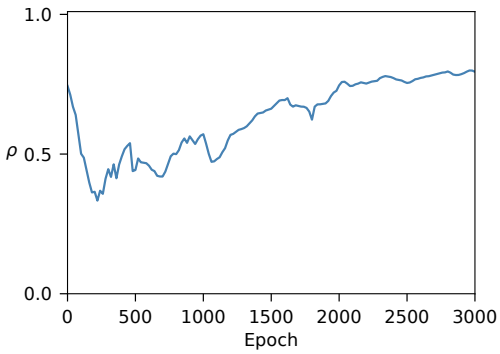
(e) Prompt Alignment on Class 8 (pivot)



(f) Correlation on Class 8 (pivot)



(g) Prompt Alignment on Class 9 (pivot)



(h) Correlation on Class 9 (pivot)

Fig. 3.12 Examples of prompt alignment and correlation obtained by the parameters generated with DiHyFo-1 for MNIST.

retains performance on these pivot classes but struggles to generate high losses, as expected. Increased scatter in the testing phase suggests that the model's ability to maintain low losses on unseen data may need further refinement.

Overall, results on MNIST and MNIST-4 show that DiHyFo-1 can generate parameters that achieve target losses, though with some errors. Deviations from the identity line indicate areas for improvement, but the model generally performs well in both training and testing, maintaining alignment between observed and target losses. Learning bidirectional loss movements at a class level is more challenging, leading to less precise prompt alignment compared to Section 2.3.2. The model captures the direction of loss changes but struggles with magnitude, making generation harder to control—likely due to limited loss variety in the dataset. However, this is not a major issue for our primary objective, as unlearning typically focuses on high and low-performing parameters rather than covering the entire loss range.

3.4 DiHyFo-2

While the conditioning of DiHyFo-1 is designed to make it behave as a leaned optimizer/de-optimizer, DiHyFo-2 is directly conditioned on the desired class losses, Figure 3.14. This type of conditioning have shown good results in other tasks using diffusion hypernetworks (Li et al., 2024; Wang et al., 2024a). We use a DiT conditioned on a large set of examples where parameters exhibit both high and low performance on individual classes. This allows us to synthesize new parameters directly in the parameter space, tailored to the specific loss properties needed for our forgetting objectives.

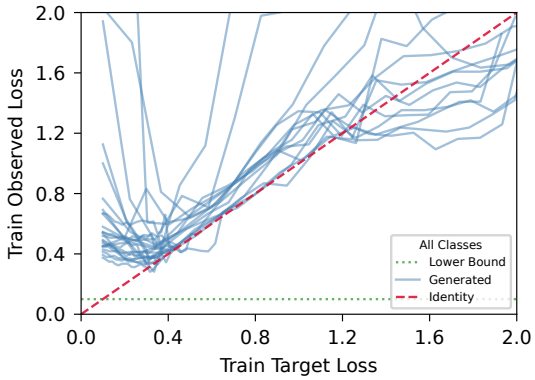
During inference, the model receives input Gaussian noise and target losses for each class, generating parameters that approximate the specified loss conditions. The task of DiHyFo-2 is to predict the distribution of parameters that achieve these desired losses, as expressed in Equation 3.7.

$$p_G(\theta^* \mid \theta, \mathscr{L}_1^*, ..., \mathscr{L}_m^*, \mathscr{L}_1, ..., \mathscr{L}_m) \tag{3.7}$$

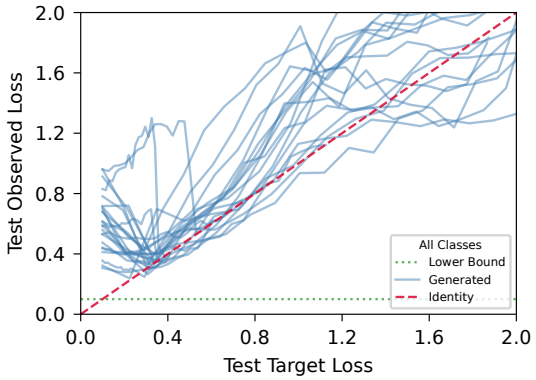
3.4.1 Training Dataset

We consider the same underlying classification tasks and main networks configurations than in Section 3.3.1, and collect checkpoints using a procedure in Figure 3.5, but giving preference for parameters that exhibit either low or high performance.

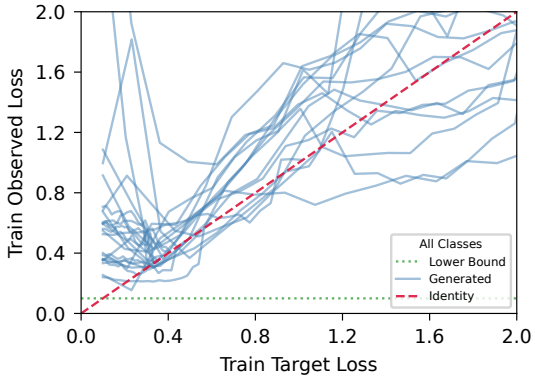
Thus, for the checkpoints random selection we give more priority to epochs at the beginning and end of the training, but also in some runs we randomly select checkpoints



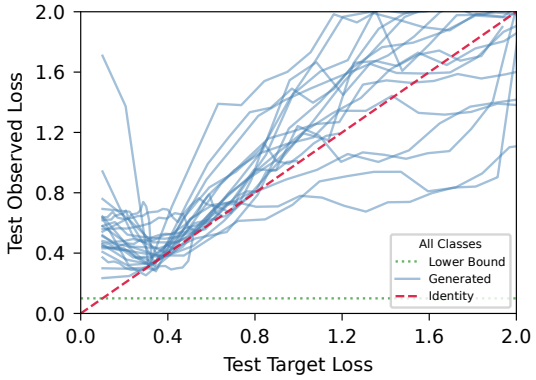
(a) Losses Comparison for Class 2 - Train



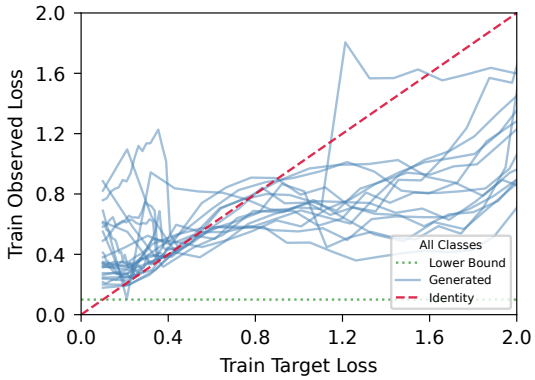
(b) Losses Comparison for Class 2 - Test



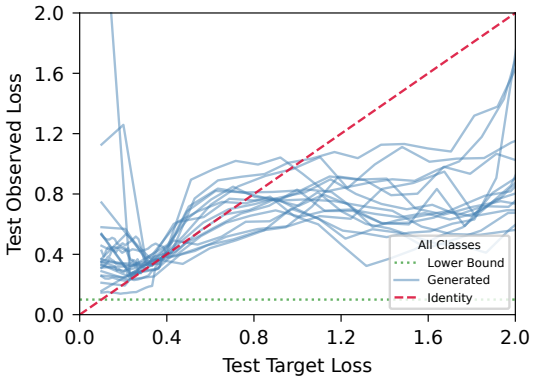
(c) Losses Comparison for Class 3 - Train



(d) Losses Comparison for Class 3 - Test



(e) Losses Comparison for Class 9 (pivot) - Train



(f) Losses Comparison for Class 9 (pivot) - Test

Fig. 3.13 Examples of Observed vs Target Losses obtained during training and testing of parameters generated with DiHyFo-1 for MNIST.

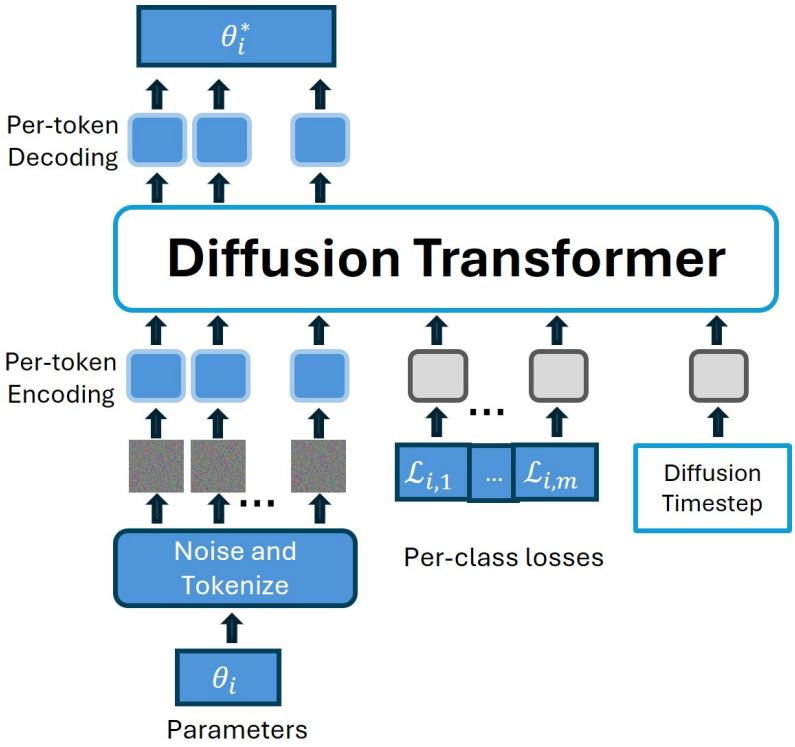


Fig. 3.14 DiHyFo-2 Architecture

across the training, to ensure that DiHyFo-2 has access to examples of both the general loss evolution process and the particular moments we are interested in for the forgetting task.

DiHyFo-2 does not requires to see a process of increments in class losses, just a variety of combinations of losses across classes, which can be done by collecting a vast amount of checkpoints from runs with different configurations.

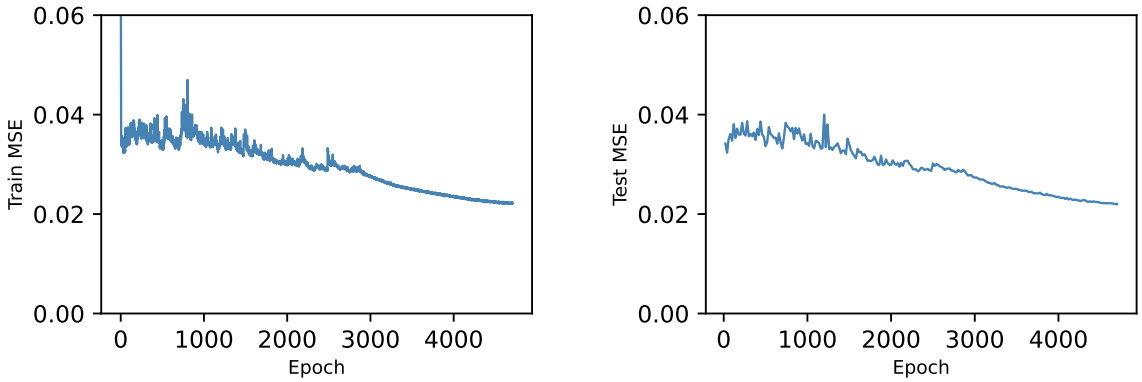
This checkpoint collection approach has proven effective for conditional parameter generation using diffusion hypernetworks (Li et al., 2024; Wang et al., 2024a). It worked well for the MNIST-4 case, however, due to disk space constraints, we reused datasets from DiHyFo-1 for MNIST, which are already aligned with the required properties and facilitate direct comparison with DiHyFo-1.

3.4.2 Training Process

Training data is loaded and preprocessed in consistent manner with the methodology outlined in Section 3.3.3.

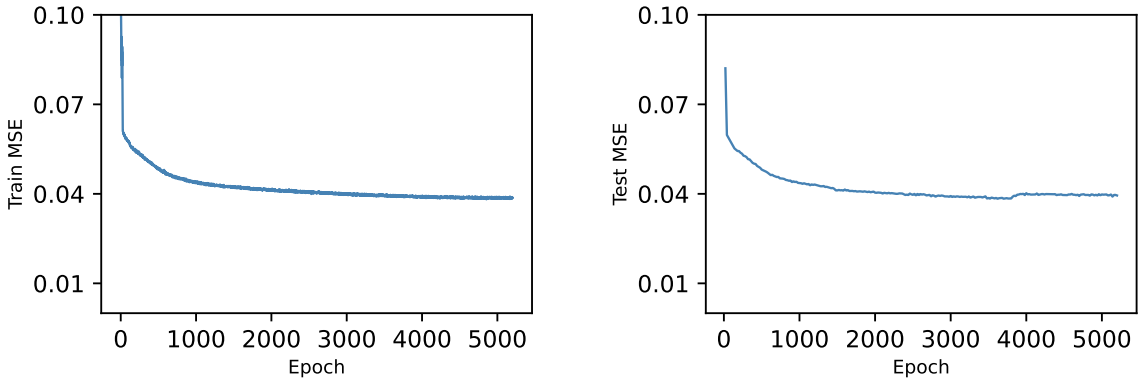
Similarly, the DiT architecture, training process, and inference process follows the methodology described in Section 3.3.3, changing the training objective for Equation 3.8.

$$\mathfrak{L}(G) = \mathbb{E}(\|\theta^* - G(\theta, \mathscr{L}_1, ..., \mathscr{L}_m, t)\|_2^2) \tag{3.8}$$



(a) Training (b) Testing

Fig. 3.15 DiHyFo-2 Learning Curves for MNIST-4.



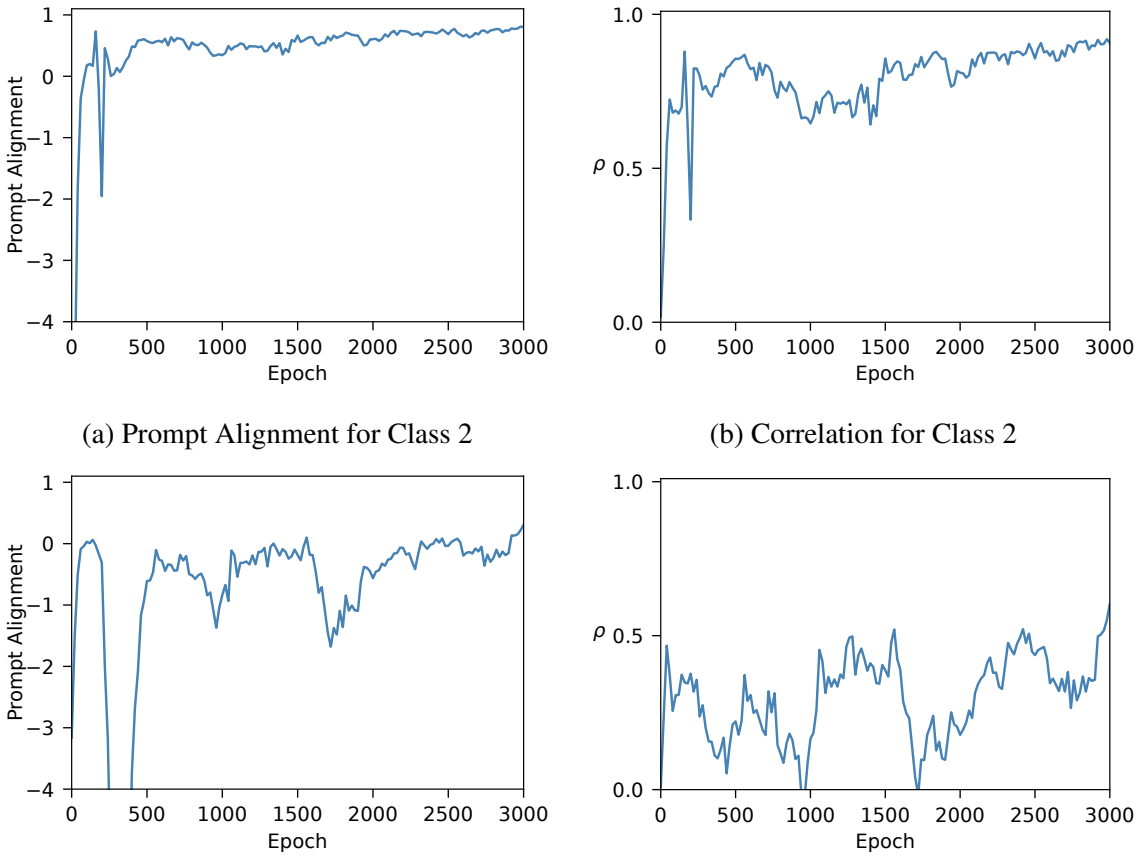
(a) Training (b) Testing

Fig. 3.16 DiHyFo-2 Learning Curves for MNIST.

3.4.3 Training Evaluation

The results in Figure 3.17 show that for Class 2 of MNIST-4, prompt alignment stabilizes early, indicating the model's quick adaptation to align its parameters with the desired performance changes. The strong alignment is further confirmed in Figures 3.18a and 3.18b, where the model closely tracks the identity line in both training and testing, with only minor deviations.

In contrast, pivot Class 1 shows more variability in both alignment and correlation, likely due to the dataset's focus on low-loss examples for pivot classes. This is evident in Figure 3.18c, where the model fits well for low losses but struggles with high losses, a common issue in generative models (Li et al., 2022; Peebles et al., 2022). However, since our focus is on fitting low-loss regions for these classes, the deviations in high-loss areas are less problematic.



(a) Prompt Alignment for Class 2

(b) Correlation for Class 2

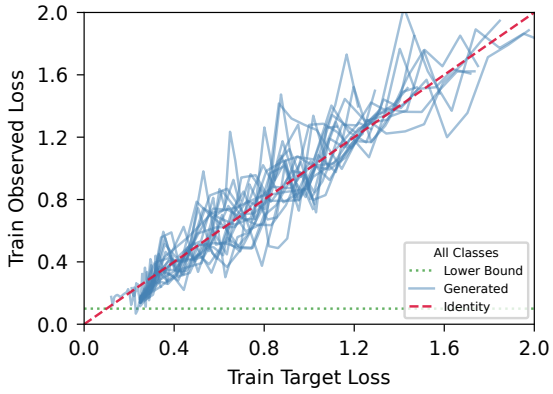
(c) Prompt Alignment for Class 1 (pivot)

(d) Correlation for Class 1 (pivot)

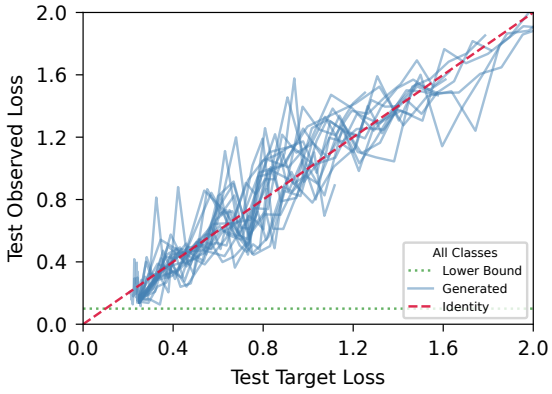
Fig. 3.17 Examples of prompt alignment and correlation obtained by the parameters generated with DiHyFo-2 for MNIST-4.

The training curves of DiHyFo-2 for MNIST indicate that the model gradually learns the conditional parameter generation task and stabilizes over time.

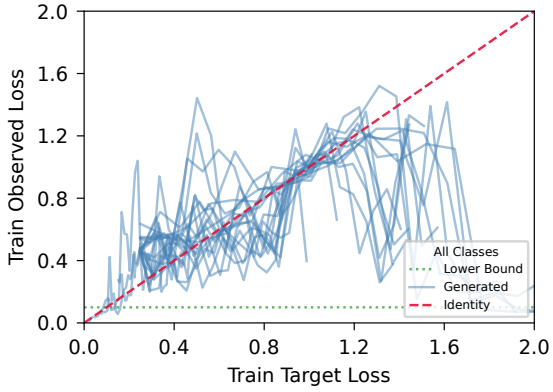
Figure 3.19 shows that as training progresses, the model increasingly aligns generated parameters with the requested performance. However, Figure 3.20 reveals that while the model learns to direct movements in the correct loss directions, it struggles to match the magnitude of these movements, often underestimating high-loss targets, particularly for pivot classes.



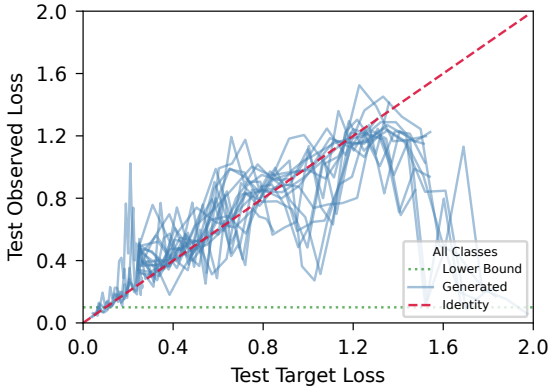
(a) Losses Comparison for Class 2 - Train



(b) Losses Comparison for Class 2 - Test

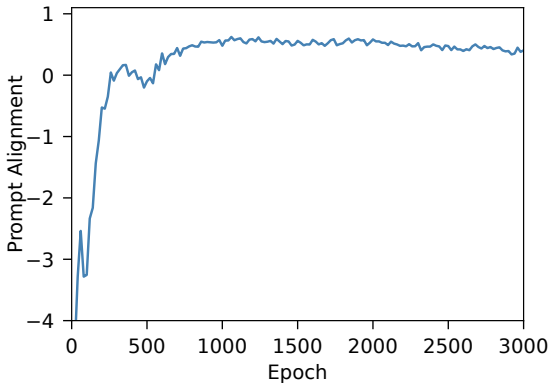


(c) Losses Comparison for Class 1 (pivot) - Train

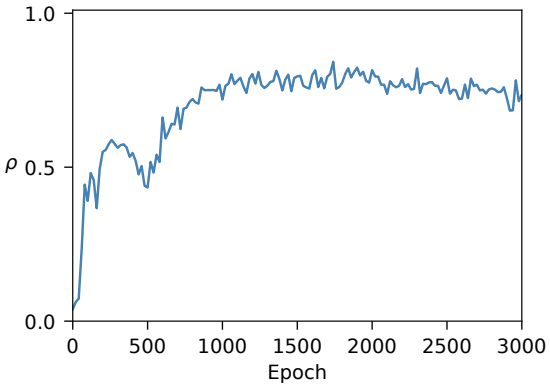


(d) Losses Comparison for Class 1 (pivot) - Test

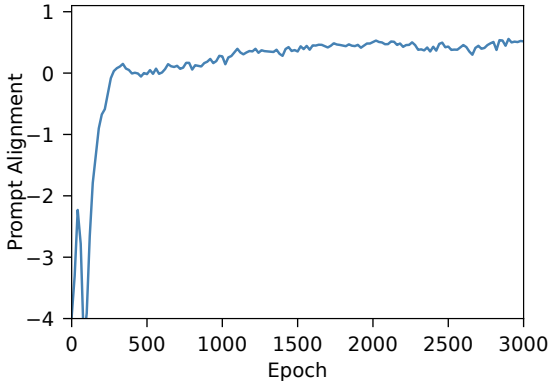
Fig. 3.18 Examples of Observed vs Target Losses obtained during training and testing generated with DiHyFo-2 for MNIST-4.



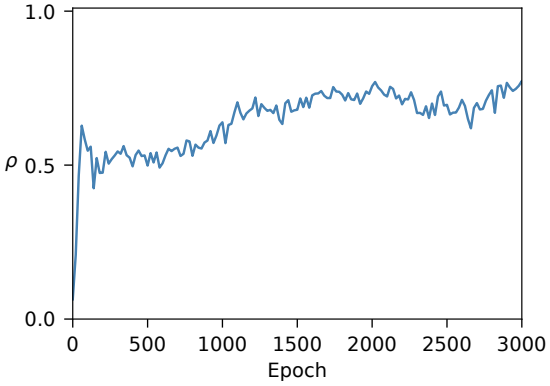
(a) Prompt Alignment for Class 2



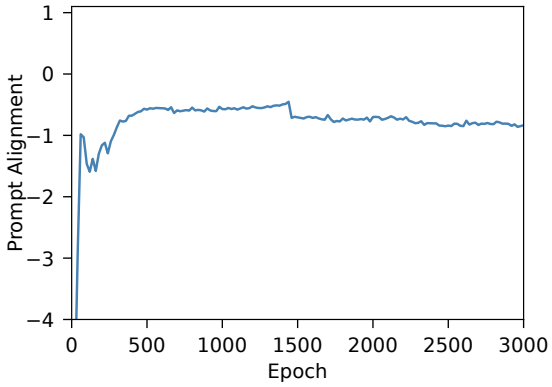
(b) Correlation for Class 2



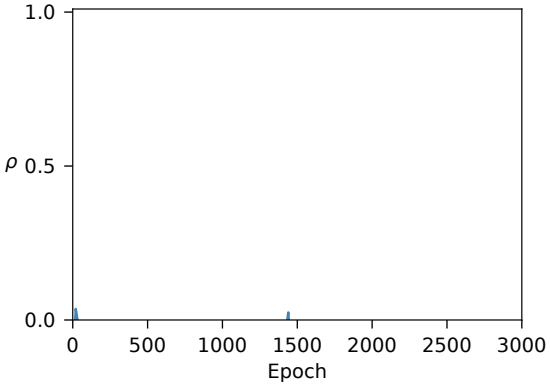
(c) Prompt Alignment for Class 3



(d) Correlation for Class 3

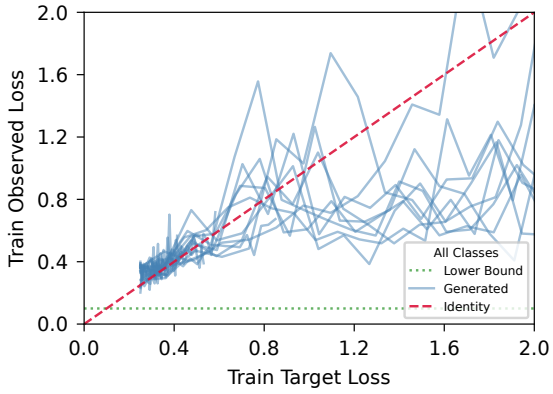


(e) Prompt Alignment for Class 9 (pivot)

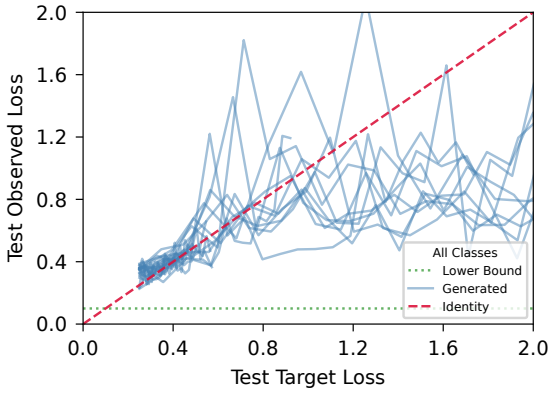


(f) Correlation for Class 9 (pivot)

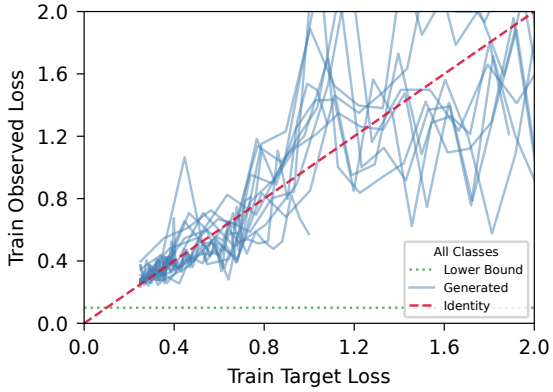
Fig. 3.19 Examples of prompt alignment and correlation obtained by the parameters generated with DiHyFo-2 for MNIST.



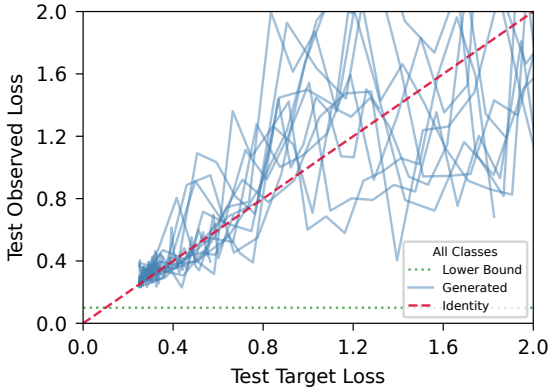
(a) Losses Comparison for Class 2 - Train



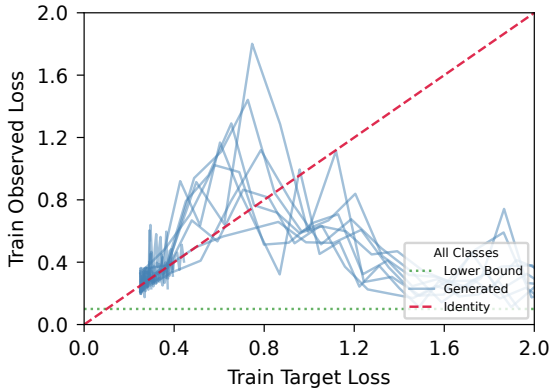
(b) Losses Comparison for Class 3 - Test



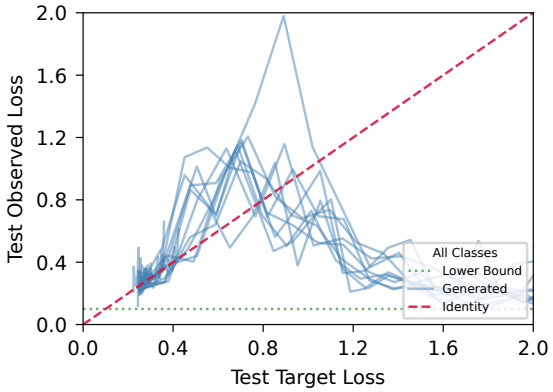
(c) Losses Comparison for Class 3 - Train



(d) Losses Comparison for Class 3 - Test



(e) Losses Comparison for Class 9 (pivot) - Train



(f) Losses Comparison for Class 9 (pivot) - Test

Fig. 3.20 Examples of Observed vs Target Losses obtained during training and testing generated with DiHyFo-2 for MNIST.

Table 3.1 Individual Unlearning Performance Metrics on MNIST-4.

MNIST-4	Sampled with DiHyfo-1		Sampled with DiHyfo-2		Retrained Model	
	D_f={2}	D_f={2,3}	D_f={2}	D_f={2,3}	D_f={2}	D_f={2,3}
Accuracy on D_r	0.9838	0.9967	0.9844	0.9995	0.9938	0.9991
Accuracy on D_f	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MIA	0.6310	0.6338	0.4432	0.3994	0.4171	0.3398

Table 3.2 Paired Unlearning Performance Metrics on MNIST-4.

MNIST-4	Sampled with DiHyfo-1		Sampled with DiHyfo-2	
	D_f={2}	D_f={2,3}	D_f={2}	D_f={2,3}
Overlap on D_r	0.9851	0.9967	0.9854	0.9995
Overlap on D_f	0.3499	0.6812	0.6657	0.7988
Overlap on Test Set	0.8270	0.8411	0.9062	0.9008
Activation Distance	0.4467	0.3597	0.1657	0.1555
Unlearning Score (φ)	0.8537	0.9272	0.9630	0.9723

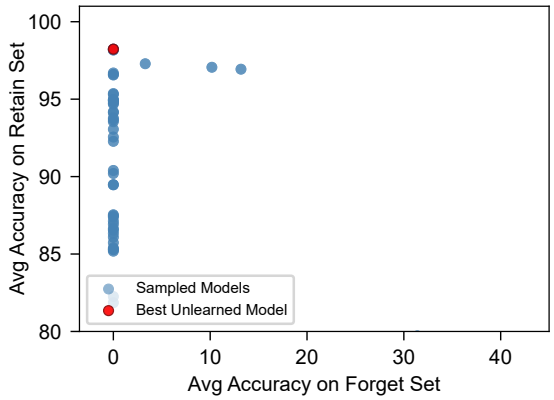
In the testing phase, increased scatter around the identity line for both varying and pivot classes suggests that while the model generalizes adequately, it faces challenges in managing multiple classes with conflicting loss objectives.

Overall, DiHyFo-2 successfully captures general loss trends but has difficulty controlling generation, especially when handling multiple classes. During inference, prompts may need adjustment to achieve the desired outcomes. These findings highlight the need for further refinement to improve the model's stability and precision in balancing forgetting and retention across different class configurations.

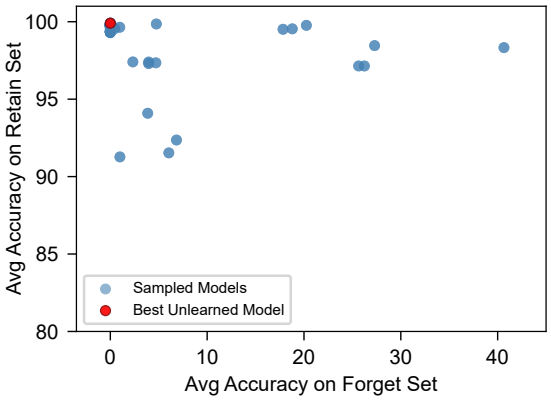
3.5 Unlearning Evaluation

To apply the DiHyFo models for unlearning, we sample multiple parameters prompting different high losses values for different forget sets simultaneously with low losses for the corresponding retain sets. The generated parameters are used to load an instance of the main network, which is then evaluated on a test set. We save the sampled model that obtains the lowest average accuracy on the forget set while obtaining the highest possible average accuracy on the retain set (best unlearned model). We use accuracy for the selection and evaluation as it is a more interpretable performance measure than loss.

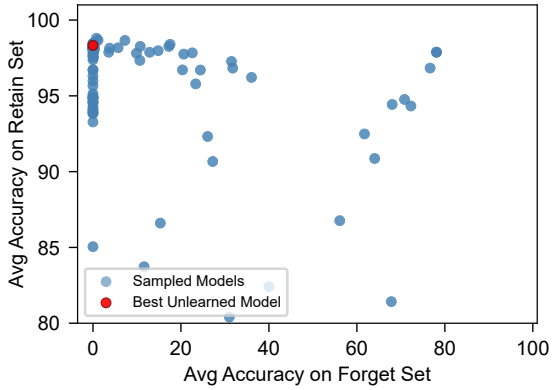
We sampled models that forget classes {2} and {2,3} for MNIST-4, and {2}, {2,3,4}, and {0,1,2,3,4} for MNIST. Figures 3.21 and 3.22 exemplifies this sampling and selection process.



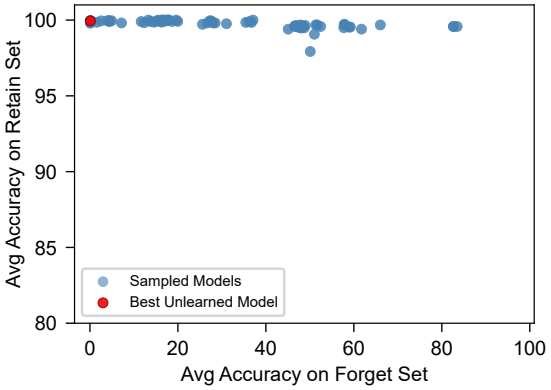
(a) DiHyFo-1 sampled models, $D_f = \{2,3\}$



(b) DiHyFo-1 sampled models, $D_f = \{2,3\}$

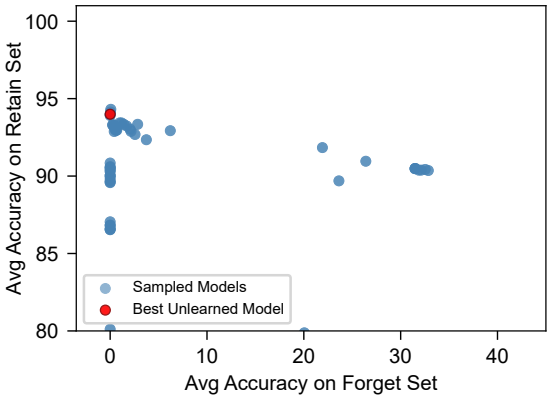


(c) DiHyFo-2 sampled models, $D_f = \{2\}$

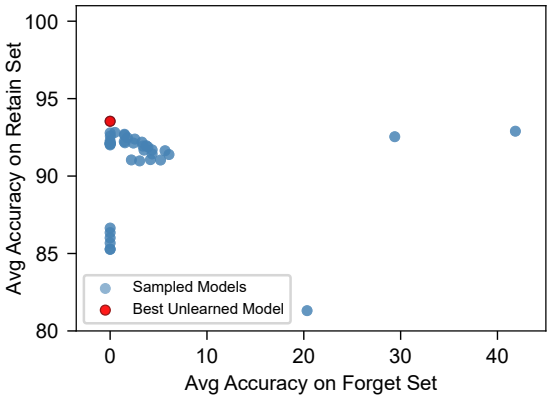


(d) DiHyFo-2 sampled models, $D_f = \{2,3\}$

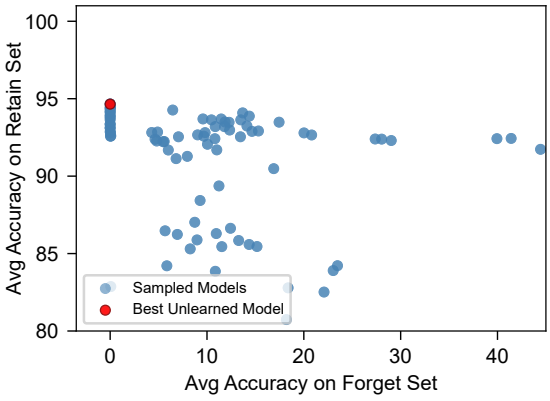
Fig. 3.21 Selection of sampled models using DiHyFo-1 and DiHyFo-2 on MNIST-4.



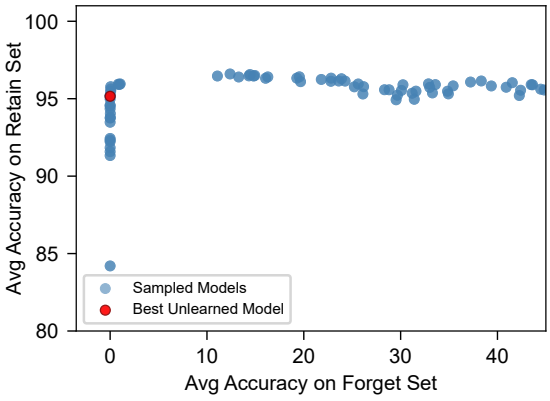
(a) DiHyFo-1 sampled models, $D_f = \{2,3,4\}$



(b) DiHyFo-1 sampled models, $D_f = \{0,1,2,3,4\}$



(c) DiHyFo-2 sampled models, $D_f = \{2,3,4\}$



(d) DiHyFo-2 sampled models, $D_f = \{0,1,2,3,4\}$

Fig. 3.22 Selection of sampled models using DiHyFo-1 and DiHyFo-2 on MNIST.

We also train an instance of the main network without the forget set to serve as the retrained model baseline. Notably, an instance of the main network needs to be retrained for each forget set considered, while the same DiHyFo can sample an unlearned model for all the forget sets considered.

Then, we proceed to the individual and pair comparisons as described in Section 3.3.3. The results of the computed metrics for the case of MNIST-4 are summarized in Tables 3.1 and 3.1, and for MNIST in Tables 3.3 and 3.3.

For all the forgetting tasks on both MNIST-4 and MNIST, the sampled unlearned models achieved zero accuracy on the forget set while maintaining high accuracy on the retain set, which indicates their robustness in preserving the performance on classes that are not subject to forgetting. Their individual performance on each set is comparable to the performance of the retrained model.Coupled with the obtained low MIA scores, the individual comparison shows achievement on the forgetting tasks and no significant signs of catastrophic forgetting.

Notably, in some cases the unlearned models obtained lower MIA scores than the retrained model, suggesting resistantance to membership inference attacks. This highlight the potential of the DiHyFo models and the HyperForget framework to synthesize models with effective unlearning capabilities.

The direct comparison of output spaces shows similar predictions on the retain and test sets, which aligns with individual performance metrics. However, noticeable differences are seen in predictions on the forget set, likely due to the models encoding different representations within their neurons, resulting in diverse classification outcomes. Despite this variability, the low concordance in the output space for the forget set is not a significant concern, as the unlearning does not lead to catastrophic outcomes. Combined with previous metrics, this suggests that the models are effectively unlearning the forget set, as they no longer rely on the data associated with those classes.

Taking as reference the predictions of the retrained model, we can compare the output spaces using the confusion matrix between the sampled models and the corresponding retrained model on the retain and complete test set. Figures and 3.24 present the results for DiHyFo-1 on MNIST-4 and DiHyFo-2 on MNIST (for additional plots see Appendix A). The forget set is not included in this comparison as it mainly contains matrices with zeros across all entries. This serves as a visual confirmation of commented findings using the evaluation metrics.

On the other hand, the activation distance is relatively low in most cases and the unlearning score is high, indicating that most of the time the unlearned models achieve a behavior close to the unlearning gold standard, with differences in how they treat the forget set.

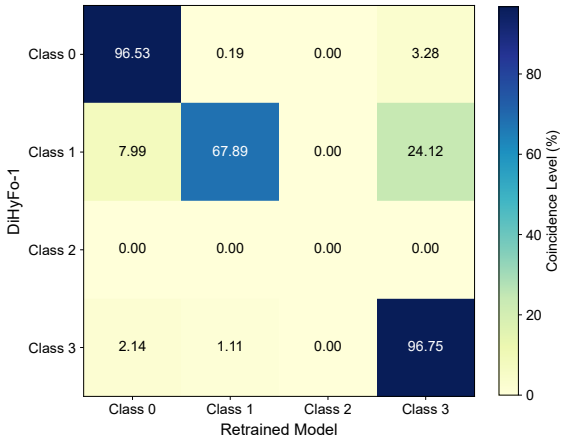
Table 3.3 Individual Unlearning Performance Metrics on MNIST.

MNIST	Sampled with DiHyfo-1			Sampled with DiHyfo-2			Retrained Model		
	D_f={2}	D_f={2,3,4}	D_f={0,1,2,3,4}	D_f={2}	D_f={2,3,4}	D_f={0,1,2,3,4}	D_f={2}	D_f={2,3,4}	D_f={0,1,2,3,4}
Accuracy on D_r	0.9110	0.9411	0.9362	0.7281	0.9460	0.9519	0.9193	0.9479	0.9504
Accuracy on D_f	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MIA	0.3376	0.3992	0.4379	0.3604	0.3241	0.3270	0.3562	0.3406	0.3798

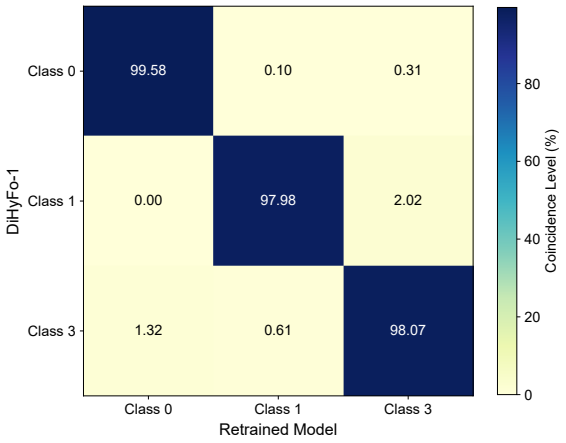
In summary, our evaluations demonstrate that both DiHyFo models can effectively sample unlearned models with behavior comparable to retraining from scratch without the forget set, in both output and parameter spaces. The models successfully unlearned the forget set while maintaining strong performance on the retain set, without significant signs of catastrophic unlearning or the Streisand effect. DiHyFo-1, in particular, produced more consistent results, especially with MNIST, where DiHyFo-2 struggled. The conditioning of DiHyFo-1 appears to make it more robust for learning and unlearning tasks. Based on these findings, and Section Section 2.1.4, we identify this approach as a potential Retrieval-Enhanced unlearning method that is model-intrinsic, dynamic, accuracy-preserving, possesses a good level of completeness, and provides guarantees of unlearning.

Table 3.4 Paired Unlearning Performance Metrics on MNIST.

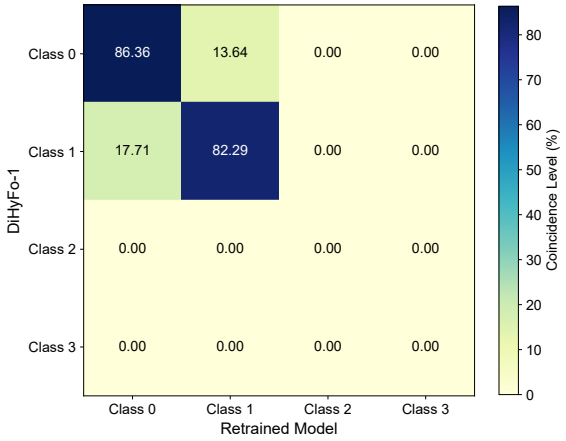
MNIST	Sampled with DiHyfo-1			Sampled with DiHyfo-2		
	D_f={2}	D_f={2,3,4}	D_f={0,1,2,3,4}	D_f={2}	D_f={2,3,4}	D_f={0,1,2,3,4}
Overlap on D_r	0.9160	0.9641	0.9560	0.7197	0.9386	0.9506
Overlap on D_f	0.5731	0.7946	0.7891	0.4035	0.7067	0.6331
Overlap on Test Set	0.8807	0.9128	0.8703	0.6868	0.8684	0.7875
Activation Distance	0.1760	0.1282	0.1715	0.9508	0.1878	0.2928
Unlearning Score (φ)	0.9876	0.9952	0.9943	0.4124	0.9852	0.9535



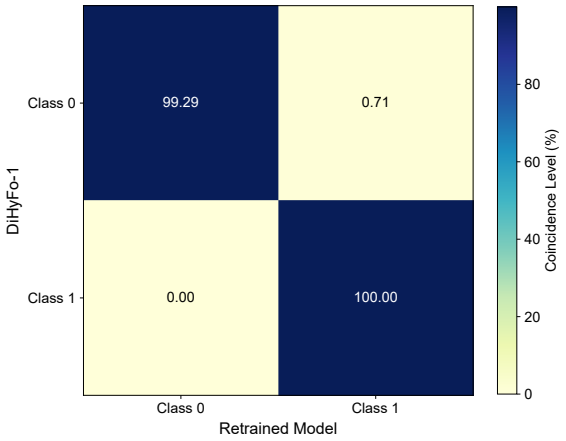
(a) Predictions on Test Set with $Df = \{2\}$



(b) Predictions on D_r with $Df = \{2\}$



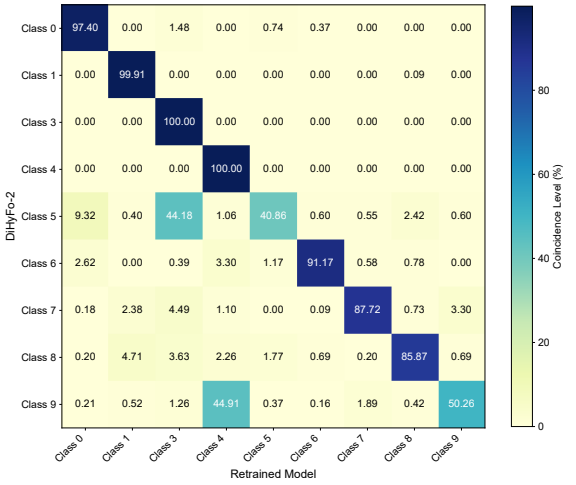
(c) Predictions on Test Set with $Df = \{2,3\}$



(d) Predictions on D_r with $Df = \{2,3\}$

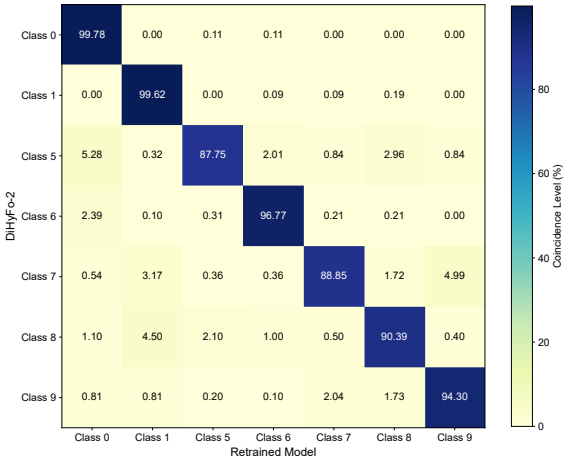
Fig. 3.23 Comparison of Predictions between DiHyFo-1 and Retrained Model on MNIST-4.

(a) Predictions on Test Set with $Df = \{2\}$



(b) Predictions on D_r with $Df = \{2\}$

(c) Predictions on Test Set with $Df = \{2,3,4\}$



(d) Predictions on D_r with $Df = \{2,3,4\}$

Fig. 3.24 Comparison of Predictions between DiHyFo-2 and Retrained Model on MNIST.

Chapter 4

Discussion

In this final section, we reflect on the insights gained from our experiments, highlighting the potential of diffusion hypernetworks for Machine Unlearning. We acknowledge both their strengths and limitations, offering suggestions for improving the current approach. We discuss the contributions of this work, draw key conclusions, and present perspectives for future research directions.

4.1 Model Limitations

While our POCs demonstrated the potential of the HyperForget framework and its associated models, DiHyFo-1 and DiHyFo-2, for addressing the Machine Unlearning problem, several critical limitations and potential drawbacks must be resolved before these approaches can be effectively applied in real-world scenarios. These concerns encompass scalability, unlearning policy compliance, generalization capabilities, generation control, and overall robustness.

The primary practical challenge is scalability. As the number of potential forget targets increases, so does the demand for data and computational resources. This can make the model impractical in scenarios with large datasets or complex models, where training and inference times may become prohibitively long. Moreover, the significant computational overhead associated with integrating diffusion models as hypernetworks further limits broader adoption, particularly for those lacking substantial computational resources.

Generalization capabilities present another concern. Although HyperForget performs well on the datasets used in this study, its ability to generalize to other datasets or more complex scenarios remains uncertain. Balancing multiple objectives—such as retaining performance on certain classes while forgetting others—increases the complexity and could affect the model's performance in diverse applications. Additionally, the model inherits challenges

from diffusion hypernetworks, including interpolation issues, limited extrapolation beyond trained loss ranges, and inconsistencies in generation capabilities.

Generative control is another issue. Although the generated parameters generally align with target losses, evaluation results reveal inconsistencies in achieving the desired loss outcomes. Often, to obtain a specific loss, a higher or lower loss must be requested, complicating control over generation outcomes. While this may not significantly impact unlearning objectives focused on high and low loss values, it poses challenges for broader applications with varied unlearning goals.

Finally, regarding Machine Unlearning and policy compliance, a critical risk involves the potential for performance recovery. The generative nature of HyperForget models allows for sampling parameters that could potentially restore performance on forgotten classes. This poses significant challenges for strict policy compliance scenarios and complete unlearning.

Addressing these challenges is essential to enable the HyperForget framework and DiHyFo models to be more broadly applicable and effectively used in Machine Unlearning tasks, ensuring they meet the rigorous demands of both industry and academia.

4.2 Conclusion and Future Work

This work introduced the HyperForget framework, a novel approach to Machine Unlearning that leverages hypernetworks to dynamically generate model parameters that selectively forget specific classes while retaining essential capabilities. Building on prior advancements, we integrated diffusion models into HyperForget, resulting in two distinct Diffusion HyperForget Networks (DiHyFo-1 and DiHyFo-2), each with unique strategies for conditioning parameter generation.

Our experiments, though limited in scope, showed that both DiHyFo models effectively achieved low accuracy on forget sets while maintaining high accuracy on retain sets, with low MIA scores indicating potential robustness against adversarial attacks. Notably, DiHyFo-1 demonstrated more consistent performance across tasks, offering assurances that the model closely mimics retraining without the need to start from scratch.

Although it is required to perform more evaluations and tests using more diverse datasets and scenarios, our POCs results give us reasons to believe that HyperForget and DiHyFo networks have potential benefits for dynamic forgetting scenarios with frequent or multiple unlearning request, potentially allowing for a more flexible and adaptable method for unlearning.

However, challenges remain. Scalability and computational complexity are significant barriers, as the framework's reliance on multiple checkpoints and diffusion models demands

high resources. This restricts HyperForget's practical application to scenarios requiring frequent dynamic forgetting or where the computational cost is justified. Additionally, the models' generalization capabilities and consistency in achieving desired loss outcomes need further refinement for broader reliability.

Concerns about the potential for knowledge recovery also pose risks for compliance with standards like GDPR. Additional safeguards and restrictions may be necessary depending on the unlearning task's context.

Addressing these limitations will be crucial to transform HyperForget into a more versatile and reliable tool for Machine Unlearning. Future work should focus on optimizing diffusion processes, exploring alternative architectures and checkpoint collection strategies, and expanding the framework's applicability beyond image classification. Additionally, as the fields of Machine Unlearning and Hypernetworks are in an early stage, there are numerous opportunities for further innovation. For instance, model-agnostic forgetting through the possibility of hypernetworks to sample parameters for different main networks architectures, new robust unlearning evaluation metrics, learning on latent spaces rather than directly on the parameter space, or improved strategies for checkpoint collection.

In conclusion, while the HyperForget framework has potential benefits for Machine Unlearning, further development is essential to overcome its current limitations. We hope this work serves as inspiration for future research that further expands the application of hypernetworks and Machine Unlearning, potentially leading to more adaptive solutions towards ethical and regulatory AI alignment.

References

Bao, F., Nie, S., Xue, K., Li, C., Pu, S., Wang, Y., Yue, G., Cao, Y., Su, H., and Zhu, J. (2023). One transformer fits all distributions in multi-modal diffusion at scale. In *International Conference on Machine Learning*, pages 1692–1717. PMLR.

Basden, B. H., Basden, D. R., and Gargano, G. J. (1993). Directed forgetting in implicit and explicit memory tests: A comparison of methods. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 19(3):603.

Baumhauer, T., Schöttle, P., and Zeppelzauer, M. (2022). Machine unlearning: Linear filtration for logit-based classifiers. *Machine Learning*, 111(9):3203–3226.

Becker, A. and Liebig, T. (2022). Evaluating machine unlearning via epistemic uncertainty. *arXiv preprint arXiv:2208.10836*.

Bessis, J. (2011). *Risk management in banking*. John Wiley & Sons.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2021). Machine unlearning. In *Proceedings - 2021 IEEE Symposium on Security and Privacy, SP 2021*, Proceedings - IEEE Symposium on Security and Privacy, pages 141–159. Institute of Electrical and Electronics Engineers Inc.

Brophy, J. and Lowd, D. (2021). Machine unlearning for random forests. In *International Conference on Machine Learning*, pages 1092–1104. PMLR.

Cao, Y. and Yang, J. (2015). Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE.

Carlini, N., Jagielski, M., Choquette-Choo, C. A., Paleka, D., Pearce, W., Anderson, H., Terzis, A., Thomas, K., and Tramèr, F. (2024). Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 175–175. IEEE Computer Society.

Chatterjee, A., Aryasomayajula, S. A., Chaudhari, R., Paul, S., and Singh, V. M. (2024). Remembering everything makes you vulnerable: A limelight on machine unlearning for personalized healthcare sector. *arXiv preprint arXiv:2407.04589*.

Chauhan, V. K., Zhou, J., Lu, P., Molaei, S., and Clifton, D. A. (2023). A brief review of hypernetworks in deep learning. *arXiv preprint arXiv:2306.06955*.

Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. (2021). When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 896–911.

Chen, R., Yang, J., Xiong, H., Bai, J., Hu, T., Hao, J., Feng, Y., Zhou, J. T., Wu, J., and Liu, Z. (2024). Fast model debias with machine unlearning. *Advances in Neural Information Processing Systems*, 36.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. (2023a). Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7210–7217.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. (2023b). Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*, 18:2345–2354.

Cohen, J. (1985). Trauma and repression. *Psychoanalytic Inquiry*, 5(1):163–189.

Costanzi, M., Cianfanelli, B., Santirocchi, A., Lasaponara, S., Spataro, P., Rossi-Arnaud, C., and Cestari, V. (2021). Forgetting unwanted memories: Active forgetting and implications for the development of psychological disorders. *Journal of Personalized Medicine*, 11(4):241.

Council of Europe (1950). European convention on human rights. Accessed: 2024-08-11.

Cuddy, L. J. and Jacoby, L. L. (1982). When forgetting helps memory: An analysis of repetition effects. *Journal of Verbal Learning and Verbal Behavior*, 21(4):451–467.

Davis, R. L. and Zhong, Y. (2017). The biology of forgetting—a perspective. *Neuron*, 95(3):490–503.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Dinsdale, N. K., Jenkinson, M., and Namburete, A. I. (2020). Unlearning scanner bias for mri harmonisation in medical image segmentation. In *Medical Image Understanding and Analysis: 24th Annual Conference, MIUA 2020, Oxford, UK, July 15-17, 2020, Proceedings 24*, pages 15–25. Springer.

Dinsdale, N. K., Jenkinson, M., and Namburete, A. I. (2021). Deep learning-based unlearning of dataset bias for mri harmonisation and confound removal. *NeuroImage*, 228:117689.

Dobrow, R. P. (2016). *Introduction to stochastic processes with R*. John Wiley & Sons.

El Ouadrhiri, A. and Abdelhadi, A. (2022). Differential privacy for deep and federated learning: A survey. *IEEE access*, 10:22359–22380.

Endress, A. D. and Johnson, S. P. (2021). When forgetting fosters learning: A neural network model for statistical learning. *Cognition*, 213:104621.

Erkoç, Z., Ma, F., Shan, Q., Nießner, M., and Dai, A. (2023). Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14300–14310.

European Commission (2024). Regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence, amending regulations and directives (artificial intelligence act). Accessed: 2024-08-11.

European Union (2000). Charter of fundamental rights of the european union. Accessed: 2024-08-11.

European Union (2016). Regulation (EU) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation). Accessed: 2024-08-11.

Fawcett, J. M. and Hulbert, J. C. (2020). The many faces of forgetting: Toward a constructive view of forgetting in everyday life. *Journal of Applied Research in Memory and Cognition*, 9(1):1–18.

Fei, Z., Fan, M., Yu, C., Li, D., and Huang, J. (2024). Scaling diffusion transformers to 16 billion parameters. *arXiv preprint arXiv:2407.11633*.

Feng, S., Miao, C., Zhang, Z., and Zhao, P. (2024). Latent diffusion transformer for probabilistic time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11979–11987.

Foster, J., Fogarty, K., Schoepf, S., Öztireli, C., and Brintrup, A. (2024a). An information theoretic approach to machine unlearning.

Foster, J., Schoepf, S., and Brintrup, A. (2024b). Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12043–12051.

Foster, J., Schoepf, S., and Brintrup, A. (2024c). Loss-free machine unlearning. *arXiv preprint arXiv:2402.19308*.

Freud, S. (1922). Repression. *The Psychoanalytic Review (1913-1957)*, 9:444.

Freud, S. (2014). *The psychical mechanism of forgetfulness*. Read Books Ltd.

Frise, M. (2018). Forgetting. In Michaelian, K., Debus, D., and Perrin, D., editors, *New Directions in the Philosophy of Memory*, pages 223–240. Routledge.

Gamboa, O. L., Garcia-Campayo, J., Müller, T., and Von Wegner, F. (2017). Suppress to forget: The effect of a mindfulness-based strategy during an emotional item-directed forgetting paradigm. *Frontiers in psychology*, 8:432.

Gatzert, N. and Wesker, H. (2012). A comparative assessment of basel ii/iii and solvency ii. *The Geneva Papers on Risk and Insurance-Issues and Practice*, 37:539–570.

Geraerts, E. and McNally, R. J. (2008). Forgetting unwanted memories: Directed forgetting and thought suppression methods. *Acta psychologica*, 127(3):614–622.

Ginart, A., Guan, M., Valiant, G., and Zou, J. Y. (2019). Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32.

Goel, S., Prabhu, A., and Kumaraguru, P. (2022a). Evaluating inexact unlearning requires revisiting forgetting. *CoRR abs/2201.06640*.

Goel, S., Prabhu, A., Sanyal, A., Lim, S.-N., Torr, P., and Kumaraguru, P. (2022b). Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*.

Goel, S., Prabhu, A., Torr, P., Kumaraguru, P., and Sanyal, A. (2024). Corrective machine unlearning.

Golatkar, A., Achille, A., and Soatto, S. (2020). Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312.

Graves, L., Nagisetty, V., and Ganesh, V. (2021). Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. (2019). Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*.

Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.

He, Y., Meng, G., Chen, K., He, J., and Hu, X. (2021). Deepobliviate: a powerful charm for erasing data residual memory in deep neural networks. *arXiv preprint arXiv:2105.06209*.

Heng, A. and Soh, H. (2024). Selective amnesia: A continual learning approach to forgetting in deep generative models. *Advances in Neural Information Processing Systems*, 36.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Hoofnagle, C. J., Van Der Sloot, B., and Borgesius, F. Z. (2019). The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98.

Huang, R., Huang, J., Yang, D., Ren, Y., Liu, L., Li, M., Ye, Z., Liu, J., Yin, X., and Zhao, Z. (2023). Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pages 13916–13932. PMLR.

Jagielski, M., Thakkar, O., Tramer, F., Ippolito, D., Lee, K., Carlini, N., Wallace, E., Song, S., Thakurta, A., Papernot, N., et al. (2022). Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*.

Jeon, D., Jeung, W., Kim, T., No, A., and Choi, J. (2024). An information theoretic metric for evaluating unlearning models. *arXiv preprint arXiv:2405.17878*.

Ji, Z., Lipton, Z. C., and Elkan, C. (2014). Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*.

Jung, C. G. (2014). *The collected works of CG Jung: Symbols of transformation (volume 5)*. Routledge.

King, P. and Tarbert, H. (2011). Basel iii: an overview. *Banking & financial services policy report*, 30(5):1–18.

Kingma, D. and Gao, R. (2024). Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2020). Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Kristiadi, A., Däubener, S., and Fischer, A. (2019). Predictive uncertainty quantification with compound density networks. *arXiv preprint arXiv:1902.01080*.

Kuhl, B. A., Dudukovic, N. M., Kahn, I., and Wagner, A. D. (2007). Decreased demands on cognitive control reveal the neural processing benefits of forgetting. *Nature neuroscience*, 10(7):908–914.

Kulikovskikh, I. and Prokhorov, S. (2018). Psychological perspectives on implicit regularization: a model of retrieval-induced forgetting (rif). In *Journal of Physics: Conference Series*, volume 1096, page 012079. IOP Publishing.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. (2024). Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36.

Li, Y., Chen, C., Zheng, X., Zhang, Y., Gong, B., Wang, J., and Chen, L. (2023). Selective and collaborative influence function for efficient recommendation unlearning. *Expert Systems with Applications*, 234:121025.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. (2022). Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Li, Y., Gu, S., Zhang, K., Van Gool, L., and Timofte, R. (2020). Dhp: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 608–624. Springer.

Li, Y., Li, W., Danelljan, M., Zhang, K., Gu, S., Van Gool, L., and Timofte, R. (2021). The heterogeneity hypothesis: Finding layer-wise differentiated network architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2144–2153.

Li, Z., Gao, L., and Wu, C. (2024). Text-to-model: Text-conditioned neural network diffusion for train-once-for-all personalization. *arXiv preprint arXiv:2405.14132*.

Liu, S., Yao, Y., Jia, J., Casper, S., Baracaldo, N., Hase, P., Xu, X., Yao, Y., Li, H., Varshney, K. R., et al. (2024a). Rethinking machine unlearning for large language models. *arXiv preprint arXiv:2402.08787*.

Liu, Y., Fan, M., Chen, C., Liu, X., Ma, Z., Wang, L., and Ma, J. (2022). Backdoor defense with machine unlearning. In *IEEE INFOCOM 2022-IEEE conference on computer communications*, pages 280–289. IEEE.

Liu, Z., Ye, H., Chen, C., and Lam, K.-Y. (2024b). Threats, attacks, and defenses in machine unlearning: A survey. *arXiv preprint arXiv:2403.13682*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Luo, C. (2022). Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*.

Ma, X., Wang, Y., Jia, G., Chen, X., Liu, Z., Li, Y.-F., Chen, C., and Qiao, Y. (2024). Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*.

Mahabadi, R. K., Ruder, S., Dehghani, M., and Henderson, J. (2021). Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*.

Marchant, N. G., Rubinstein, B. I., and Alfeld, S. (2022). Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7691–7700.

Marks, I., Lovell, K., Noshirvani, H., Livanou, M., and Thrasher, S. (1998). Treatment of posttraumatic stress disorder by exposure and/or cognitive restructuring: A controlled study. *Archives of general psychiatry*, 55(4):317–325.

Micaelli, P. and Storkey, A. J. (2019). Zero-shot knowledge transfer via adversarial belief matching. *Advances in Neural Information Processing Systems*, 32.

Moulin, C. J., Perfect, T. J., Conway, M. A., North, A. S., Jones, R. W., and James, N. (2002). Retrieval-induced forgetting in alzheimer's disease. *Neuropsychologia*, 40(7):862–867.

Murayama, K., Miyatsu, T., Buchli, D., and Storm, B. C. (2014). Forgetting as a consequence of retrieval: a meta-analytic review of retrieval-induced forgetting. *Psychological bulletin*, 140(5):1383.

Nasirigerdeh, R., Razmi, N., Schnabel, J. A., Rueckert, D., and Kaissis, G. (2024). Machine unlearning for medical imaging. *arXiv preprint arXiv:2407.07539*.

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A. F., Ippolito, D., Choquette-Choo, C. A., Wallace, E., Tramèr, F., and Lee, K. (2023). Scalable extraction of training data from (production) language models. *ArXiv*.

Nguyen, T. T., Huynh, T. T., Nguyen, P. L., Liew, A. W.-C., Yin, H., and Nguyen, Q. V. H. (2022). A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

Norman, K. A., Newman, E. L., and Detre, G. (2007). A neural network model of retrieval-induced forgetting. *Psychological review*, 114(4):887.

Oh, G. and Peng, H. (2022). Cvae-h: Conditionalizing variational autoencoders via hypernetworks and trajectory forecasting for autonomous driving. *arXiv preprint arXiv:2201.09874*.

Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

Peebles, W. S., Radosavovic, I., Brooks, T., Efros, A. A., and Malik, J. (2022). Learning to learn with generative models of neural network checkpoints. *ArXiv*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Ren, K., Zheng, T., Qin, Z., and Liu, X. (2020). Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*.

Roeder, G., Metz, L., and Kingma, D. (2021). On linear identifiability of learned representations. In *International Conference on Machine Learning*, pages 9030–9039. PMLR.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494.

Schoepf, S., Foster, J., and Brintrup, A. (2024). Potion: Towards poison unlearning.

Schürholt, K., Kostadinov, D., and Borth, D. (2021). Self-supervised representation learning on neural network weights for model characteristic prediction. *Advances in Neural Information Processing Systems*, 34:16481–16493.

Sekhari, A., Acharya, J., Kamath, G., and Suresh, A. T. (2021). Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086.

Shastri, S., Wasserman, M., and Chidambaram, V. (2019). The seven sins of Personal-Data processing systems under GDPR. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, Renton, WA. USENIX Association.

Shuang Sha, A., Pereira Nunes, B., and Haller, A. (2024). " forgetting" in machine learning and beyond: A survey. *arXiv e-prints*, pages arXiv–2405.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Soleymani, F., Paquet, E., Viktor, H. L., and Michalowski, W. (2024). Structure-based protein and small molecule generation using egnn and diffusion models: A comprehensive review. *Computational and Structural Biotechnology Journal*.

Sommer, D. M., Song, L., Wagh, S., and Mittal, P. (2020). Towards probabilistic verification of machine unlearning. *arXiv preprint arXiv:2003.04247*.

Song, Q., Tan, R., and Wang, J. (2023). Towards efficient personalized driver behavior modeling with machine unlearning. *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, pages 31–36.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Storm, B. C., Bjork, E. L., and Bjork, R. A. (2008). Accelerated relearning after retrieval-induced forgetting: the benefit of being forgotten. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 34(1):230.

Tarun, A. K., Chundawat, V. S., Mandal, M., and Kankanhalli, M. (2023a). Deep regression unlearning. In *International Conference on Machine Learning*, pages 33921–33939. PMLR.

Tarun, A. K., Chundawat, V. S., Mandal, M., and Kankanhalli, M. (2023b). Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*.

Thudi, A., Deza, G., Chandrasekaran, V., and Papernot, N. (2021). Unrolling sgd: Understanding factors influencing machine unlearning. *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319.

United Nations (1948). Universal declaration of human rights. Accessed: 2024-08-11.

Van Vugt, M. K. and Jha, A. P. (2011). Investigating the impact of mindfulness meditation training on working memory: A mathematical modeling approach. *Cognitive, Affective, & Behavioral Neuroscience*, 11:344–353.

Vaswani, A. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Voigt, P. and Von dem Bussche, A. (2017). The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555.

Volk, T., Ben-David, E., Amosy, O., Chechik, G., and Reichart, R. (2022). Example-based hypernetworks for out-of-distribution generalization. *arXiv preprint arXiv:2203.14276*.

Von Oswald, J., Henning, C., Grewe, B. F., and Sacramento, J. (2019). Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*.

Wang, K., Xu, Z., Zhou, Y., Zang, Z., Darrell, T., Liu, Z., and You, Y. (2024a). Neural network diffusion. *ArXiv*.

Wang, W., Tian, Z., and Yu, S. (2024b). Machine unlearning: A comprehensive survey. *arXiv preprint arXiv:2405.07406*.

Wixted, J. T. (2004). The psychology and neuroscience of forgetting. *Annu. Rev. Psychol.*, 55(1):235–269.

Wu, G., Hashemi, M., and Srinivasa, C. (2022). Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8675–8682.

Wu, J., Ji, W., Fu, H., Xu, M., Jin, Y., and Xu, Y. (2024). Medsegdiff-v2: Diffusion-based medical image segmentation with transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 6030–6038.

Xu, J., Wu, Z., Wang, C., and Jia, X. (2024). Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. (2022). Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*.

Yoon, Y., Nam, J., Yun, H., Lee, J., Kim, D., and Ok, J. (2022). Few-shot unlearning by model inversion. *arXiv preprint arXiv:2205.15567*.

Zhang, D., Finckenberg-Broman, P., Hoang, T., Pan, S., Xing, Z., Staples, M., and Xu, X. (2023a). Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *arXiv preprint arXiv:2307.03941*.

Zhang, D., Pan, S., Hoang, T., Xing, Z., Staples, M., Xu, X., Yao, L., Lu, Q., and Zhu, L. (2024). To be forgotten or to be fair: Unveiling fairness implications of machine unlearning methods. *AI and Ethics*, 4(1):83–93.

Zhang, H., Nakamura, T., Isohara, T., and Sakurai, K. (2023b). A review on machine unlearning. *SN Computer Science*, 4(4):337.

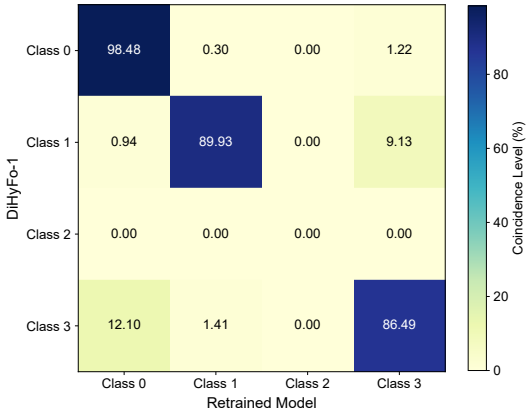
Zhu, L., Liu, X., Liu, X., Qian, R., Liu, Z., and Yu, L. (2023). Taming diffusion models for audio-driven co-speech gesture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10544–10553.

Zoellner, L. A., Feeny, N. C., Bittinger, J. N., Bedard-Gilligan, M. A., Slagle, D. M., Post, L. M., and Chen, J. A. (2011). Teaching trauma-focused exposure therapy for ptsd: Critical clinical lessons for novice exposure therapists. *Psychological Trauma: Theory, Research, Practice, and Policy*, 3(3):300.

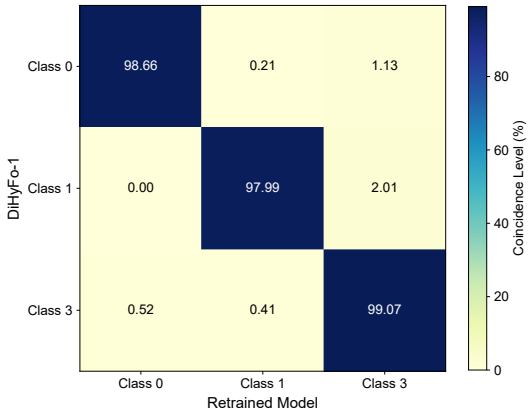
Appendix A

Additional Supporting Results

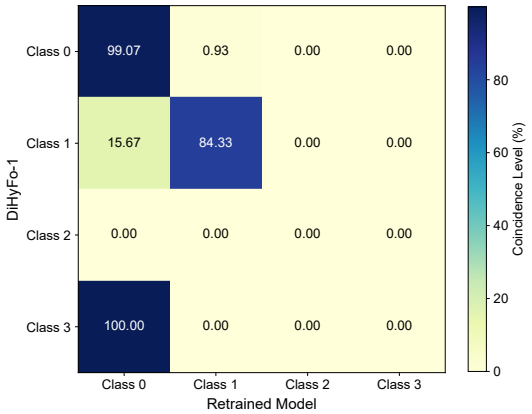
Additional Confusion Matrices for Unlearning Evaluation



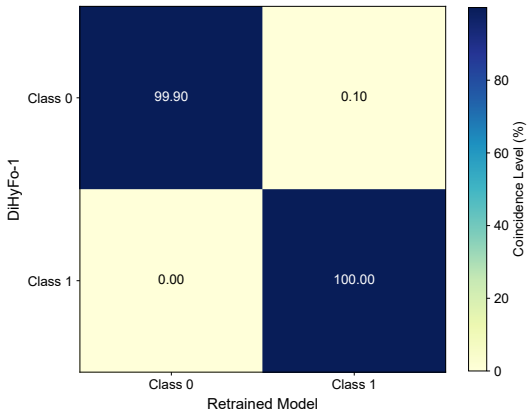
(a) Predictions on Test Set with $Df = \{2\}$



(b) Predictions on D_r with $Df = \{2\}$

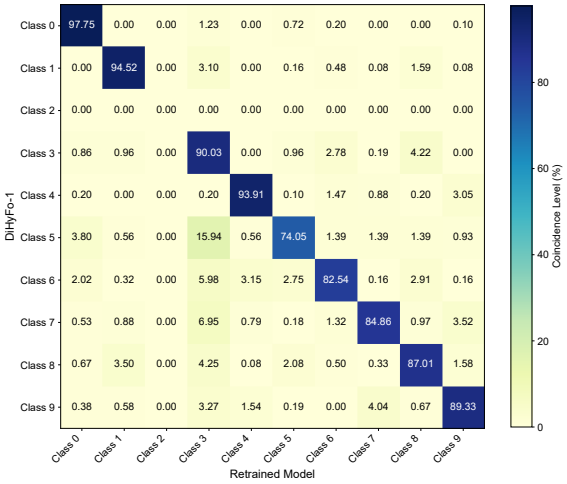


(c) Predictions on Test Set with $Df = \{2,3\}$

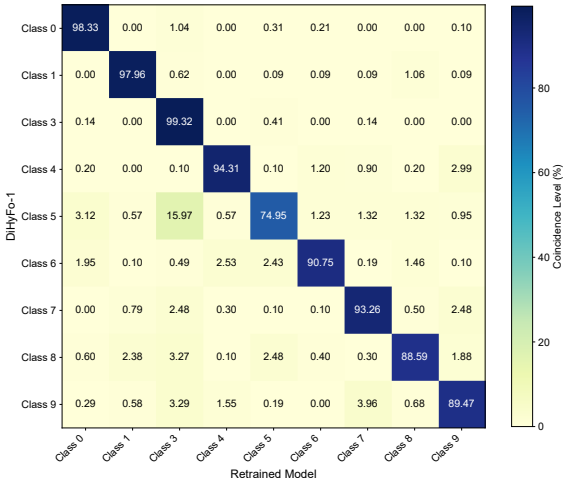


(d) Predictions on D_r with $Df = \{2,3\}$

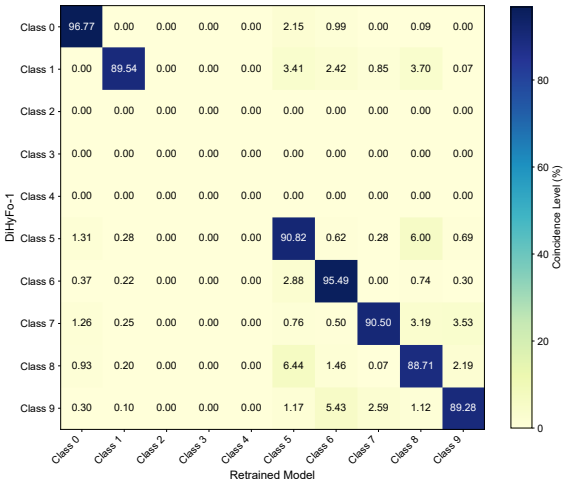
Fig. A.1 Comparison of Predictions between DiHyFo-2 and Retrained Model on MNIST-4.



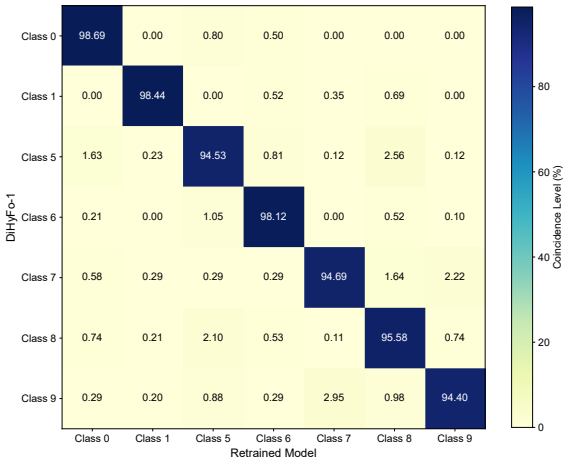
(a) Predictions on Test Set with $Df = \{2\}$



(b) Predictions on D_r with $Df = \{2\}$



(c) Predictions on Test Set with $Df = \{2,3,4\}$

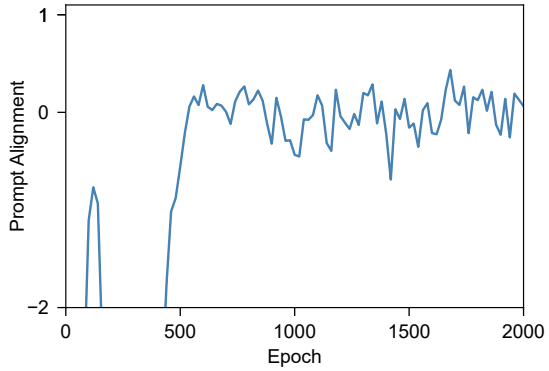


(d) Predictions on D_r with $Df = \{2,3,4\}$

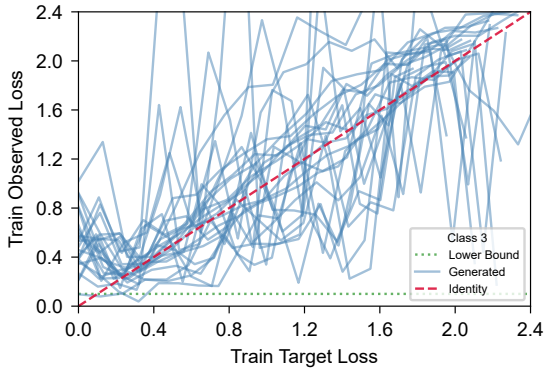
Fig. A.2 Comparison of Predictions between DiHyFo-1 and Retrained Model on MNIST.

Learning to Learn on Individual Classes

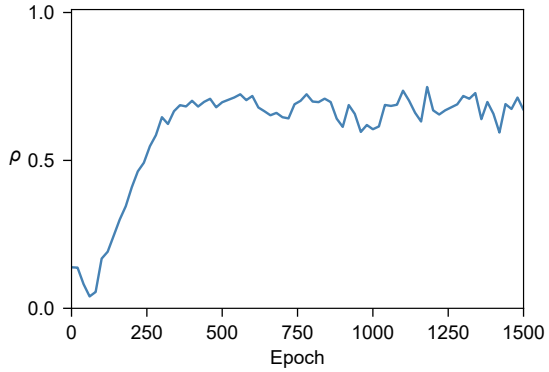
We present some results of G.pt reviewed in Section 2.3.2 when trained conditioned on an individual class loss. This was intended to evaluate the capabilities of G.pt to learn to generate parameters conditioned on individual class losses. The results showed that G.pt had potential to learn with this type of condition, still it needed modifications to be able to learn from multiple classes simultaneously and to learn to forget.



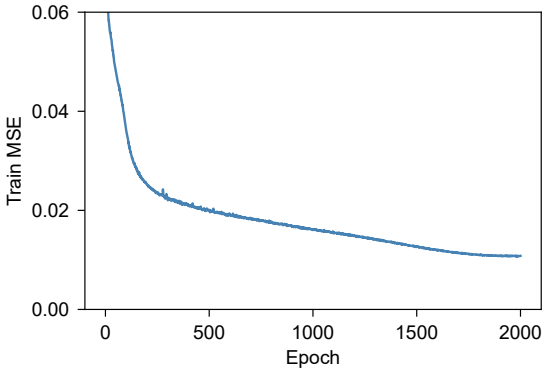
(a) Prompt alignment - Train.



(b) Loss comparison - Train.



(c) Correlation - Train.



(d) Training Learning Curve.

Fig. A.3 Behavior of G.pt when trained conditioned on one class loss.