Multi-label Lifelong Machine Learning using Deep Generative Replay

by

Mohammed Awal Kassim

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science,
Concentration in Applied Artificial Intelligence (AI)

Ottawa-Carleton Institute of Computer Science
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of ottawa
Ottawa, Canada

Abstract

Lifelong machine learning concerns the development of systems that continuously learn from a set of diverse tasks, incorporating new knowledge without forgetting the knowledge they have previously acquired. One of the main challenges within this paradigm is the stability-plasticity dilemma, which entails balancing a model's adaptability in terms of incorporating new knowledge, corresponding to new tasks, with its stability in terms of retaining previously acquired knowledge (known tasks). Multi-label classification is a supervised learning process in which each instance is assigned multiple non-exclusive labels. When faced with multi-label data, the lifelong learning challenge becomes even more pronounced, as it becomes essential to preserve relations between multiple labels across sequential tasks. This thesis begins with a scoping review that explores the intersection of lifelong learning and multi-label classification. Addressing a major knowledge gap we identified, we introduced Multi-label Deep Generative Replay (MLDGR), a framework for learning from multi-label image datasets across a sequence of tasks with minimal forgetting. As an initial task, the MLDGR approach assigns multiple labels to images. This is achieved by utilizing a convolutional neural network (CNN) for feature extraction from images, which are then input into a base classifier for multi-label classification. In the second stage of our algorithm, selected image samples are transformed into textual representations and stored in a buffer. When learning a new task in the lifelong learning pipeline, the stored textual representations (from prior tasks) are used to generate synthetic images which are then interleaved with real images from the current task. This novel process mitigates forgetting by reinforcing the model's retention of previously acquired knowledge, thereby preventing performance degradation on earlier tasks. Experiments on widely used sequentialized multi-label image datasets demonstrate the effectiveness of our approach and confirm that our algorithm outperforms the state-of-the-art.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Professors Herna L. Viktor and Wojtek Michalowski for their unwavering support, patience, and guidance throughout my studies. Your invaluable feedback has made me a better researcher, and I could not have done this without you.

Also, I would like to thank my family and friends for their love, support, prayers, and well wishes. Thank you all for believing in me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The idea of building systems that reason like humans and are capable of performing intellectual tasks has intrigued researchers since the advent of computing. The seminal work of [3] introduced the concept of Machine Learning (ML) and demonstrated its potential for playing checkers at a high level. This marked an early recognition that reliable, effective, and robust machine intelligence can be achieved by developing algorithms that are capable of learning and improving their performance over time as opposed to handcrafting rules for performing specific tasks. This stands in addition to several key milestones accelerated the growth in ML, and established a subfield of Artificial Intelligence (AI) that focuses on the methods for developing algorithms that can learn from data without requiring explicit programming.

A major drawback with existing ML algorithms is their inability to learn in a continual manner [1] rather than in isolation. This means they learn to perform a particular task and operate under the assumption that data encountered during deployment has the same characteristics as the training data and is independently distributed or sampled from a

static distribution. This is a very limiting assumption as it does not always holds in the real world and hence the performance of ML models on new data tend to degrade. To build systems capable of learning similarly to the way humans do requires algorithms that learn continually and have the ability to identify and learn new tasks. Algorithms that learn in a lifelong manner can help overcome the drawbacks of the current learning paradigm.

Lifelong machine learning is an ML paradigm that focuses on the design and development of algorithms and systems that learn continuously, accumulate knowledge, and are capable of identifying and learning new tasks [1]. Learning with previously accumulated knowledge can potentially eliminate the need for a large number of labeled training instances. Key among the challenges presented by this paradigm is the stability-plasticity dilemma, a trade-off between a model's plasticity for integrating new knowledge and stability to prevent the forgetting of previous knowledge [4].

One of the many problems ML is widely used to address is classification, where an algorithm learns to assign labels to test instances based on examples seen during training. In multi-label classification, labels are mutually inclusive; hence, any test instance may be associated with multiple labels simultaneously. Lifelong machine learning in a multi-label setting has interesting use-cases in the real world. In healthcare for example, a patient may suffer from diabetes, heart disease, and asthma at the same time. It is not only important to identify all three conditions but also to identify the presence of a complication or adverse interactions that have not previously been seen.

1.1 Motivation

The ability to learn from multiple tasks sequentially is essential in the real world, where systems are expected to adapt and evolve over time. Consider the example of a self-

driving car that learns to navigate roads and identify traffic signs simultaneously. As time goes by, it should also be able to learn new tasks, such as detecting potholes and avoiding wildlife, without forgetting its previous knowledge. This ability to accumulate knowledge is essential for developing truly intelligent systems that can operate in dynamic and complex environments [1]. The data that constitute these tasks in the real world are sometimes multi-label, meaning that each data point can belong to multiple classes simultaneously. For example, an image captured by a self-driving car's camera might contain both a pothole and a pedestrian, requiring the model to identify both objects. This is different from multi-class data, where each data point belongs to a single class. The majority of research in the field of lifelong machine learning has focused on multi-class data, often evaluating algorithms on image datasets such as MNIST and CIFAR-10. However the ability to handle multi-label data is crucial for systems operating in the real world where objects do not usually occur in isolation.

One of the widely used techniques in the field of lifelong machine learning rely on an approach that reuses previously encountered examples to refresh the model's memory in order to prevent the forgetting of previously learned information while continuously learning new tasks. This is known as exemplar rehearsal. However, while this technique is effective in reducing forgetting, it comes with a significant drawback: it demands a considerable amount of working memory [5]. This requirement can be a major problem, especially in situations where computational resources are limited. Moreover, exemplar rehearsal poses challenges in contexts where data privacy is important [2], as storing and reusing data might not be feasible.

Relatively few studies in the lifelong learning literature have explored the potential of deep generative replay in multi-class problems, with none in multi-label problems. This approach involves generating new training examples that simulate the original data, thereby

eliminating the need to store actual data samples directly. Studies have shown that deep generative replay reduces the demand on working memory and also outperforms other lifelong learning strategies in terms of effectiveness [2]. In this thesis, we propose a method based on this paradigm.

While current approaches that assign single labels to instances have demonstrated state-of-the-art performance in various applications, they are not suitable for many real-world scenarios where input data is inherently multi-faceted and cannot be adequately represented by a single label. Furthermore, rehearsal-based methods cannot be used in applications where data is no longer available after training or access to the data is limited. This thesis addresses this issue by introducing a method that generates representative synthetic images of prior tasks from stored textual descriptions. This ensures learning is done in a lifelong manner without requiring access to real images from earlier tasks.

1.2 Objective

The objective of this thesis is to address the existing research gap in the domain of multi-label lifelong machine learning. As stated above, the current research predominantly focuses on exemplar rehearsal and lacks substantial investigation into generative replay. As such, we aim to explore the potential and applicability of generative replay for learning from multi-label data across a sequence of tasks with minimal forgetting. To achieve this, we introduce MLDGR, a deep generative replay technique, tailored for the multi-label lifelong learning context. Deep generative replay fundamentally enables a model to generate new data instances based on previous learning, facilitating the retention of knowledge without the need to store actual data. MLDGR extends this concept by incorporating an image-to-text sub-module. This sub-module is tasked with converting selected image samples

from prior tasks into textual descriptions, which then serve as a basis for generating new images pertinent to the current learning task. By storing selected samples in the form of text and using this for generating replay samples, the approach ensures that the labels in the task are identified and stored efficiently. Capturing and retaining label information in a compact textual format is a novel contribution of this work. This approach has not been utilized in prior research, and it offers a unique way to preserve and utilize the knowledge of labels in multi-label learning scenarios involving images, enhancing the model's ability to generate relevant and representative replay samples.

In addition, recall that lifelong learning algorithms based on exemplar rehearsal rely on the availability of data from previous tasks when learning the current task, in order to mitigate forgetting. However, this is not feasible in scenarios where past data is not available due to privacy concerns. By creating synthetic images, MLDGR aims to learn new tasks without the need to access real images from previous tasks. This makes this technique particularly useful in scenarios where past image data is unavailable after training.

Further more, existing approaches for handling the multi-label problem in lifelong machine learning either focus more on other properties of the dataset such as its inherent imbalance or the specific application for which they are developed. As such, they fall short in properly addressing the multi-label classification problem. MLDGR incorporates a multi-label solver, which is designed to learn from tasks with data instances involving multiple labels. It uses a CNN to extract features relevant to the multiple labels associated with each image. These features are then fed to a classifier to make predictions for each label, taking into account the relation between labels. The classifier is configured with the appropriate loss and activation functions to ensure it learns each label separately, at the same time considering all the features extracted by the CNN.

In summary, the overall aim of this thesis is to narrow the research gap in multi-label

lifelong machine learning by demonstrating the efficacy of MLDGR. It seeks to prove that MLDGR can sustain the acquisition of new knowledge across multi-label tasks over time, while preventing the loss of previously learned information.

1.3 Thesis Organization

This thesis is structured around two components. The first is a comprehensive scoping review of multi-label lifelong machine learning, detailed in Chapter 2. Within this chapter, Sections 2.1 and 2.2 provide a foundational overview of multi-label classification and lifelong learning, respectively. We discuss key techniques used in each domain and highlight real-world applications of lifelong learning. In Section 2.3, the research criterion used in the review is explained, stating our research questions, findings, and knowledge gaps. Section 2.4 outlines challenges in multi-label lifelong learning and potential directions for future research and concludes this segment of the thesis.

The second portion of the thesis addresses a significant gap uncovered in the review. Chapter 3 introduces MLDGR, detailing the architecture's components and how they work together to achieve the objective of our research. Chapter 4 is dedicated to the experimental design, focusing on the selection of datasets and the specifications of the hardware and software utilized. Here, the outcomes of these experiments are presented, followed by an analysis and the extraction of insights. The thesis concludes in chapter 5, where we restate our contributions and identify future research directions stemming from this work.

Chapter 2

Background

The work presented in this chapter is based on our paper: *Multi-label Lifelong Machine Learning: A Scoping Review of Algorithms, Techniques, and Applications* [6] submitted for publication.

In this chapter, we begin by providing separate overviews of each learning paradigm, then proceed to consolidate our findings, highlighting the interplay between the two paradigms. This chapter is intended to help readers develop insights into the various algorithms, techniques, and applications of multi-label lifelong machine learning.

2.1 Overview of Multi-label Classification

In ML, classification is a fundamental task which involves categorizing input data into pre-defined classes or labels [7]. It plays an important role in various domains, including image recognition and text classification. Accurately predicting the class labels of new, previously unseen examples is the main goal of classification. Classification tasks can be

broadly categorized into two types: single label and multi-label classification. In single label classification tasks, each instance in a dataset is assigned to only one label. However, in many real-world applications instances may have multiple labels at the same time. Multi-label classification addresses this situation by allowing instances to be simultaneously associated with multiple labels [8], thus expanding the scope of classification and providing a realistic representation of real-world problems.

2.1.1 Formal Definition

Mathematically, a multi-label data instance is represented by $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i = (x_1, ..., x_d)$ is a d-dimensional vector of features and $y_i = (y_1, ..., y_L)$ is the associated set of labels, with each $y_l \in \{0, 1\}$. As formulated by Zhang and Zhou, the multi-label training set $D = \{(\mathbf{x}_i, y_i)|1 \leq i \leq m, \}$, where m is the total number of training examples; the task of multi-label classification is to create a classifier that learns from D a function f that maps the input space $\mathbf{X}$ to the binary exponential label space [9]. Formally:

$$f : \mathbf{X} \rightarrow 2^Y \tag{2.1}$$

For any unseen example, the classifier predicts the set of labels based on the learned relationship between the input space $\mathbf{X}$ and the label space 2^Y. Fig. 2.1 provides some examples of multi-label data across different modalities. The image in this example contains multiple objects (a bowl, bread, broccoli, and an orange). In such a recognition task, the goal is to accurately identify and label each of the distinct objects in the image. To accomplish this, the algorithm must learn to predict multiple labels for a single image. The text represents a single instance from a text classification dataset where a set of predefined categories are assigned to the given text. For the sensor data, each instance

is represented by a single row in the table. A combination of features (temperature, humidity, moisture, etc.) could yield different weather conditions (rainy, cloudy) that are not mutually exclusive. It is worth noting that the major difference between multi-class and multi-label classification is that each instance is allowed one— and only one — label in multi-class problems.

As is evident in (2.1), the size of the label sets increases exponentially as the number of labels increases. This is a major challenge associated with learning from multi-label data. For example, a dataset with five unique labels would have about 32 possible label sets. Increasing the number of labels to 10 would significantly increases the possible label sets to over a thousand (1024 to be precise). It is therefore important to take advantage of any label relations in order to address this challenge. Zhang and Zhang categorized the existing strategies for exploiting label relations into three families, based on the degree of associations considered by each strategy [10].

The first-order strategy addresses multi-label learning by treating each label independently, neglecting the co-existence of other labels. This approach decomposes the multi-label learning problem into separate binary classification tasks for each label [11–13], resulting in simplicity and efficiency but potentially also sub-optimal performance due to the disregard of label relations. The second-order strategy considers pairwise relations between labels, involving ranking labels as relevant or irrelevant [14–16] or interactions between pairs of labels [8, 17–19]. This strategy partially exploits label relations, leading to good generalization performance. Finally, the high-order strategy deals with high-order relations among labels. Some approaches assume the association of all other labels with each label [20–23], while others address connections among a random subset of labels [24–26].

Figure 2.1: Example of multi-label data across different modalities.

Due to the degree of relation modeled in high order strategies, the problem of learning

from a vast output space is alleviated. However, these strategies are more computationally expensive and less scaleable than first and second order strategies. The aforementioned categorization is solely based on the degree of association between labels in any given dataset. Throughout the literature, two main techniques have been developed to handle multi-label classification: problem transformation and algorithm adaptation. This taxonomy is shown in Fig. 2.2. Each of these techniques adopt one or more of the strategies introduced above. In problem transformation techniques, the original multi-label classification problem is transformed into one or more single label problems, while algorithm adaptation techniques modify existing algorithms to cater to the constraints of multi-label data.

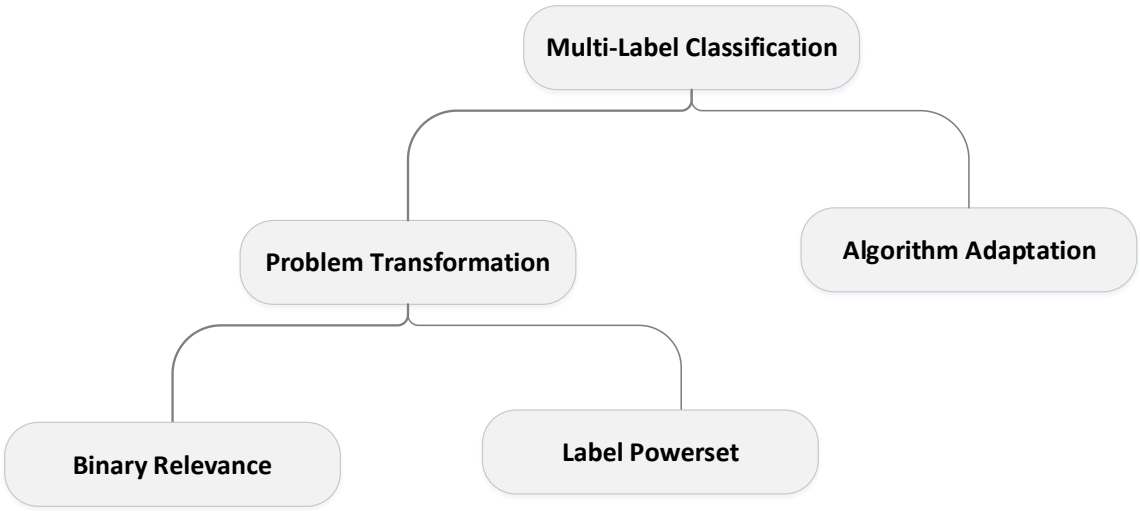


Figure 2.2: The taxonomy of multi-label classification techniques.

2.1.2 Problem Transformation

Problem transformation, as previously stated, involves converting the original multi-label problem into one or more binary or multi-class classification problems that can be solved using existing single-label classification algorithms. The fundamental idea is to divide the multi-label problem into a number of easier sub-problems, each of which focuses on determining whether a single label will be present or absent [27]. Binary Relevance (BR) is a first order strategy that is frequently employed in the problem transformation process. BR strategies approach the multi-label problem as a collection of independent binary classification problems, with a separate binary classifier trained for each label. Each binary classifier is trained utilizing the original feature space and the related binary labels for the specific label being predicted. At test time, each classifier independently predicts the presence or absence of its associated label, and the sum of these binary predictions yields the final multi-label prediction.

Label Powerset (LP) strategies transform the multi-label task into a multi-class classification problem, whereby a unique combination of labels forms a distinct class [9]. In essence, LP strategies treat every unique label combination as a different class and train a multi-class classifier to predict the correct label combination for every instance. This method explicitly models label relations while taking into account the joint distribution of labels. The LP approach, however, has scaling problems because the number of possible label combinations can increase exponentially as more labels are added.

Tsoumakas et al. worked on BR's adherence to the One-Versus-All (OVA) approach, in which a separate binary dataset is created for each label [28]. The Classifier Chains (CC) model [25] utilizes N binary classifiers that are interlinked, with each classifier incorporating the labels predicted by the previous classifiers as additional features. This

approach considers label relations randomly and exhibits linear complexity with respect to the number of labels. The Probabilistic Classifier Chains (PCC) model is a Bayes optimal approach to forming classifier chains that outperforms the original CC model at the expense of computational time [29] . Antenreiter et al. implement a BR strategy in two stages [30]. In the first stage, the model learns from the data, and in the second stage it carries out meta-learning.

The Vanilla LP method [11] constructs a single-label dataset, considering each possible combination of labels as a separate class. A multi-class algorithm is then employed for further processing. Pruned Problem Transformation (PPT) or Pruned Sets (PS) algorithms aim to reduce the complexity of LP by focusing on the most crucial label combinations [31]. They achieve this by pruning examples with less frequent label sets. Tsoumakas et al. introduce an ensemble-based algorithm which utilizes random projections of the label space that construct multiple LP classifiers, each trained with a random subset of k labels [26].

2.1.3 Algorithm Adaptation

Unlike problem transformation, which converts the multi-label problem into multiple single-label or multi-class classification tasks, algorithm adaptation focuses on modifying existing classification algorithms or developing new ones explicitly designed for multi-label data [27]. This approach recognizes the intrinsic complexity of multi-label problems and seeks to take advantage of label relations during model training and prediction.

An adaptation of the C4.5 algorithm [32] to the Multi-Label Learning (MLL) setting has been introduced [12]. The original C4.5 algorithm was developed to generate decision trees using the concept of entropy, and is capable of handling both continuous and discrete attributes. To adapt it for MLL, the algorithm was modified to enable multiple labels in

the leaves, and the definition of entropy was modified to consider both membership and non-membership of each class. The Multi-layer Multi-Perceptron (MMP) algorithm [33] associates each label with a separate perceptron, and the performance of the entire ensemble is taken into account when updating each perceptron, in contrast to BR (Binary Relevance). The study demonstrates that MMP exhibited superior performance compared to BR in text classification tasks. Predictive Clustering Tree (PCT) [34] is a flexible framework used to perform prediction tasks by defining a distance metric and prototype. It has been successfully applied to various tasks, including predicting tuples of variables and hierarchical multi-label classification, where each label represents a component of the target tuple. PCTs are generated top-down, with data partitioned into clusters to minimize intra-cluster variation at each node. The Multi-Label Paired Comparisons (ML-PC) method [35] utilizes two probabilistic binary classifiers to distinguish between each pair of overlapping classes. Wan and Xu define a set of linear classifiers optimized to minimize a measure that evaluates the average fraction of label pairs that are reversely ordered for each instance [36]. Multi-label k Nearest Neighbours (kNN) [37] uses lazy learning to determine the k nearest neighbors before computing a membership counting vector which indicates the number of neighbors belonging to each possible class. Using the statistical information thus derived from the label sets of the neighbors, the set of labels for the unseen instance is determined based on the maximum a posteriori (MAP) principle.

It must be mentioned, however, that these methods are not always used in isolation. Many studies use an ensemble of learners. Vateekul and Kubat, for example, employ an ensemble of decision trees to automate the categorization of multi-label text documents described by thousands of features [38]. Read et al. use an ensemble of pruned sets to identify the most relevant relations between labels [24]. This pruned sets approach operates by treating sets of labels as a single label in order to reduce the complexity of the output

space.

Given that this chapter is about exploring the interplay between multi-label classification and lifelong learning, we do not provide an exhaustive overview of all multi-label algorithms. Interested readers are referred to [9], [39], [40] and [41] for more comprehensive reviews of multi-label algorithms.

2.2 Fundamentals of Lifelong Learning

2.2.1 What is Lifelong Machine Learning (LML)?

The concept of lifelong machine learning emerged from the realization that ML algorithms often struggle to adapt to dynamic and evolving environments [1]. Lifelong machine learning approaches aim to develop algorithms that can continuously learn and adapt to new data, tasks and environments, without forgetting previously acquired knowledge. The term lifelong learning is often used synonymously with continual learning [42]. Although both these approaches focus on the idea of learning over time, they are distinct in scope and objectives. Lifelong learning encompasses a broader vision of learning across diverse tasks and domains over an extended period. In addition to knowledge retention, lifelong learning systems are capable of discovering new tasks and using accumulated past knowledge to help future learning [1]. Continual learning is a specific area within this broader context that only deals with the challenges of learning continuously from new data while preserving knowledge of past tasks.

According to Thrun's definition of lifelong machine learning [43], a learning system undergoes a sequential learning process where it accumulates knowledge from N previously encountered tasks. Subsequently, when presented with the $(N+1)th$ task, the system

leverages the knowledge it has acquired from the preceding N tasks to facilitate the learning process for the $(N+1)th$ task. This approach allows the system to benefit from prior experiences and effectively transfer learned knowledge to new tasks, thus promoting continuous learning and adaptation over time. This definition emphasizes the utilization of previously acquired knowledge as a means to enhance the system's performance on new tasks and enable effective knowledge transfer. However, this definition, while insightful, introduces some ambiguity regarding the precise interpretation of the terms "task" and "knowledge". That is, one must reflect on whether a task is defined by a specific problem instance, a distinct learning objective, or a combination of both. Additionally, the notion of "knowledge" remains somewhat vague. It is unclear whether the term knowledge refers to generalizable insights, or to the underlying model parameters acquired during the learning process. Due to the lack of explicit clarification of these fundamental concepts in Thrun's definition, there have been attempts in the literature to provide a more concise definition.

Chen and Liu define lifelong machine learning as a continuous learning process: "At any time point, the learner performed a sequence of N learning tasks, $T_1, T_2, ..., T_N$. These tasks can be of the same type or different types and from the same domain or different domains. When faced with the $(N+1)th$ task T_{N+1} (which is called the new or current task) with its data D_{N+1}, the learner can leverage past knowledge in the knowledge base to help learn T_{N+1}. The objective of LML is usually to optimize the performance on the new task T_{N+1}, but LML can optimize any task by treating the rest of the tasks as previous tasks. A Knowledge base is constructed to maintain the knowledge learned and accumulated from the previous task. When learning T_{N+1} is complete, the knowledge base is updated with the knowledge gained from learning T_{N+1}. The updating can involve inconsistency checking, reasoning, and meta-mining of additional higher-level knowledge" [1]. While this expanded definition has a broader scope and effectively enhances the conceptualization of a task, it

falls short of providing a clear definition of knowledge. Fig. 2.3 and Fig. 2.4 illustrate the difference between classical machine learning and the lifelong machine learning paradigm.

Parisi et al. highlight the importance of avoiding catastrophic forgetting and the need for efficient mechanisms to consolidate and transfer knowledge across different tasks and experiences [44]. Their comprehensive review offers valuable insights into the field of lifelong machine learning and provides a broader perspective on the challenges it involves, along with potential solutions to them.

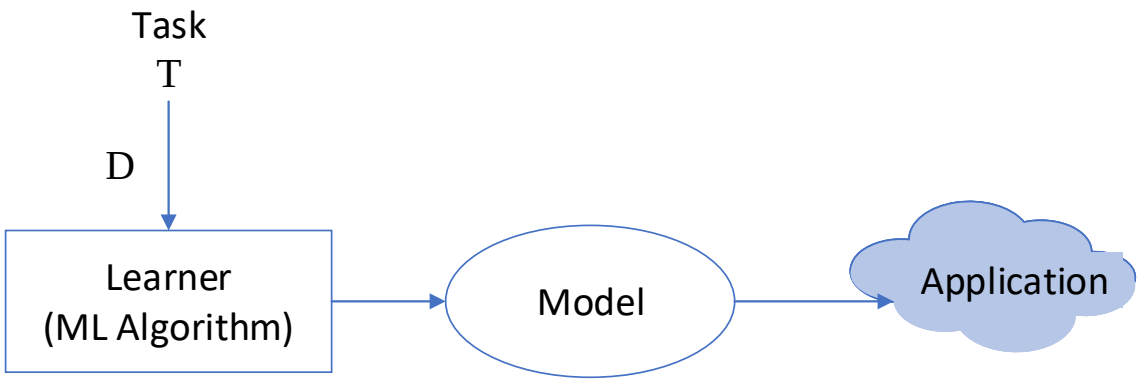


Figure 2.3: Classical machine learning architecture as depicted by [1]

2.2.2 Lifelong Learning Techniques and Algorithms

Learning in a continual manner presents some challenges, with catastrophic forgetting, as previously mentioned, being a particularly significant problem [43] [4]. As such, research in this field for the past three decades has been geared towards mitigating catastrophic forgetting. Existing work on lifelong learning can be grouped into three categories: regularization, rehearsal, and parameter isolation approaches (shown in Fig. 2.5. Our catego-

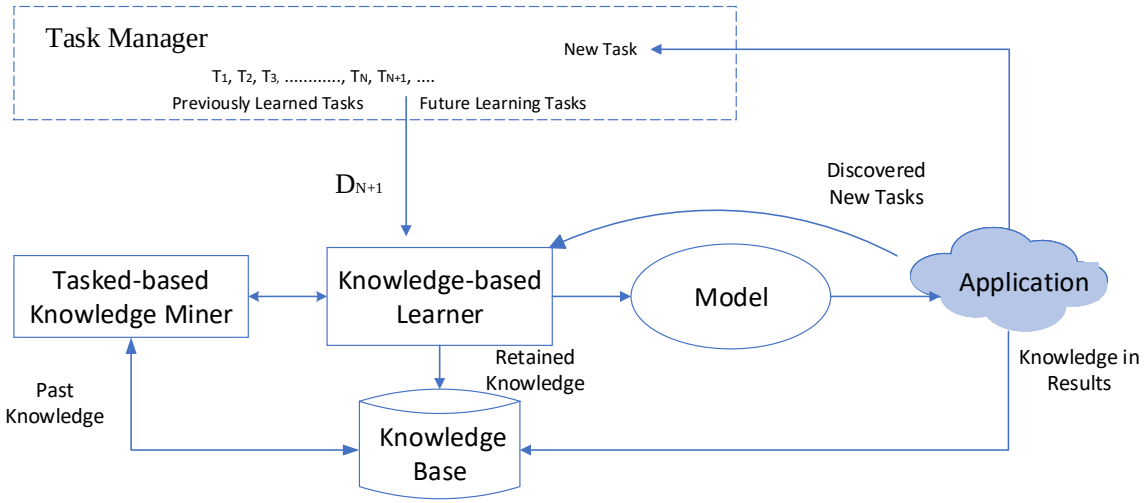


Figure 2.4: Lifelong learning paradigm [1]

rization is inspired by the work of De Lange et al. [45]. As with traditional ML techniques, a strict allocation of works into these three categories is not always practical, and hence some researchers provide a fourth category to represent works that combine multiple techniques [42] In this section, the effectiveness of these approaches are examined to provide insights into their practical implementation and limitations.

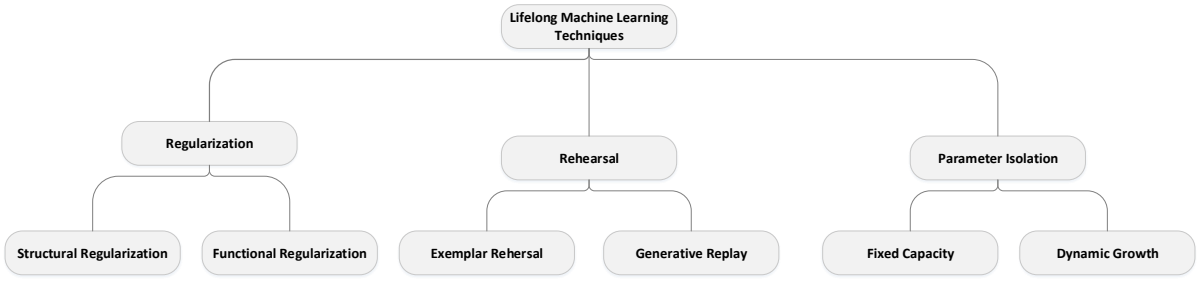


Figure 2.5: Lifelong learning techniques

2.2.2.1 Regularization

Regularization is a broad term used to refer to a set of techniques that can prevent overfitting in neural networks by imposing constraints on updating model parameters. In the lifelong learning context, regularization techniques aim to minimize forgetting by preventing the model from overfitting a new task while preserving representations learned from previous tasks.

Elastic Weight Consolidation (EWC) [46] alleviates catastrophic forgetting by imposing a quadratic penalty on the discrepancy between the parameters learned from old and new tasks, which aids in slowing down learning for task-relevant weights in order to preserve previously acquired knowledge. Synaptic Intelligence (SI) [47] enablesestimating the significance of individual synapses (parameters) for solving a learned task. This algorithm penalizes alterations to the most relevant synapses, facilitating the learning of new tasks with less forgetting. Chaudhry et al. generalize EWC and SI by creating an objective function that utilizes both Fisher information-based importance and an additional optimization-path-based importance score [48]. The latter perspective involves calculating distances within the induced Riemann manifold and optimizing the importance score based on the optimization trajectory. Aljundi et al. compute the importance of neural network parameters in an unsupervised and online manner, preventing important knowledge related to previous tasks from being overwritten when learning new tasks [49]. Learning without Forgetting (LwF) [50] involves the use of Convolutional Neural Networks (CNNs). The network with predictions from previously learned tasks is forced to be similar to the network handling the current task through knowledge distillation (the transfer of knowledge from a large, highly regularized model to a smaller one). The LwF algorithm optimizes a set of shared parameters across all tasks (θ_s) while optimizing the parameters of the new task

(θ_n). It also imposes an additional constraint to ensure that predictions on the samples of the new task using θ_s and the parameters of old tasks θ_o do not undergo significant shifts, in order to retain θ_o's memory.

Regularization methods offer a means of mitigating catastrophic forgetting under specific circumstances. Nevertheless, they introduce additional loss terms to safeguard consolidated knowledge. As highlighted in Parisi et al., this may result in a trade-off between performance on old and new tasks when neural network resources are limited [44].

2.2.2.2 Rehearsal

Rehearsal, also known as replay, has emerged as a promising approach to addressing catastrophic forgetting. It involves the selective reintroduction of past data samples during the training of a model on new tasks [51, 52]. These techniques aim to preserve learned knowledge by re-exposing the model to previous experiences. Replay techniques can be broadly classified into two categories, namely Exemplar Rehearsal and Generative Replay. Exemplar Rehearsal involves storing a buffer of selected past data samples that are representative of the previously observed distribution. This minimizes the amount of memory required to store all previous samples. Generative replay techniques leverage generative models, such as Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs), to generate synthetic data samples which resemble the distribution of previously encountered tasks. These synthetic samples are then combined with the current task's data during training. This provides a means of supplementing the training process with data that resemble past experiences. While these are the two primary sub-categories of rehearsal, they can be combined to form a hybrid method. As the name implies, hybrid approaches combine real and synthetic data samples for rehearsal. By utilizing a mixture of real past data samples and generated samples, such approaches aim to provide a diverse

and representative training experience for the model and to allow the model to leverage the benefits of both approaches.

Gradient Episodic Memory (GEM) [53] is a good example of exemplar replay. This approach facilitates the positive transfer of knowledge to previous tasks. GEM employs episodic memory to store a subset of observed examples from each task, which helps mitigate catastrophic forgetting. While minimizing the loss on the current task, it ensures that the losses on the episodic memories of previous tasks are treated as inequality constraints, allowing for their decrease without an increase. Building on the foundational work on GEM, significant improvements have been made in terms of the computational and memory costs involved in optimization under the constraints of gradient updating [54]. Deep Generative Replay (DGR) [2] is a typical example of the generative replay approach. It involves a dual-model architecture comprising a deep generative model and a task solver. This design allows training data from previously learned tasks to be sampled; these data can then be interleaved with data from new tasks. Consequently, there is no need to explicitly revise old training samples for experience replay, leading to reduced working memory requirements.

2.2.2.3 Parameter Isolation

Parameter isolation approaches to lifelong learning focus on modifying the underlying architecture to accommodate lifelong learning scenarios (this is also referred to as the architectural approach in the literature [44] [42]). This category can be sub-divided into two sub-categories: fixed capacity and dynamic growth. Fixed capacity approaches attempt to maintain a fixed parameter size throughout the learning process. Such methods have faced criticism due to their reliance on an overparametrized network [42]. On the other hand, dynamic growth architectures address this limitation by allowing the network to

dynamically expand its capacity to accommodate new tasks. In a neural network setting, these architectures can grow by adding new neurons, modules, or layers when learning new tasks. This expansion capability enables them to maintain their performance on previous tasks while efficiently incorporating new information.

Bayesian Neural Networks [55] are a classic example of methods that maintain a fixed capacity. They work by guiding task-specific information through the architecture. The key idea is to adjust the system so that a given maximum number of units in the neural network are highly active at any time. This reduces the overlap between activities performed for different tasks. Such methods also employ a measure of uncertainty to refine the model, creating specific binary masks for each task that help pinpoint the relevant parts of the model's complex weight distributions.

Progressive Neural Networks (PNN) [56] aim to preserve a network trained on previous knowledge while expanding the architecture with new sub-networks to accommodate new information. This is a good example of a method that uses dynamic growth architecture. A pool of pre-trained models is retained, one for each learned task. When a new task (T_{N+1}) is presented, a new neural network is created, and lateral connections with the existing tasks are learned to facilitate knowledge transfer. Mallya et al. use a mechanism that is similar to PNN, except it introduces a gating mechanism to automate the selection of a suitable model from an ensemble of learners [57].

In this section, we have provided an overview of lifelong learning techniques and highlighted notable algorithms that play a pivotal role in this domain. Understanding the foundational principles and algorithms of lifelong learning sets the stage for exploring their practical applications, which will be the primary focus of the upcoming section. For a more in-depth study of lifelong learning techniques, refer to [1], [44], and [42]

2.3 Findings

So far, we have provided an overview of multi-label classification and lifelong learning in isolation. Here, we explore how these areas come together, and share insights about multi-label lifelong learning. Drawing inspiration from the definition of lifelong learning established by Chen and Liu [1], we propose a tailored definition for this nascent sub-field. Formally, we define multi-label lifelong learning as an ongoing and continuous learning paradigm in which the learner engages in a sequence of N learning tasks, denoted as $T_1, T_2, ..., T_N$. In each task T_i, the learner is presented with a dataset D_i, whereby each instance of the dataset is associated with a label set y. The label set y for each instance is defined such that each label can take on values in the set {0, 1}, indicating the absence (0) or presence (1) of the corresponding label.

Recall that no comprehensive review of the literature on multi-label lifelong learning has yet been published. While Chen and Liu [1], Parisi et al. [44] and several others have published reviews which focus solely on lifelong learning, and Gibaja et al. [39] have focused on multi-label classification, there exists a growing body of work that explores the integration of lifelong learning in a multi-label setting. Our aim is to identify the extent and scope of this work and shed light on the challenges and opportunities presented by this emerging research area.

2.3.1 Research Methodology

The scoping review methodology employed in this study is grounded in the five-stage framework proposed by Arksey and O'Malley [58]. By adhering to this systematic and rigorous process, this study ensures complete transparency, facilitating replication of the search results and the reliability of our findings. Arksey and O'Malley's framework encompasses

five essential stages, each executed to explore and synthesize the existing literature on multi-label lifelong machine learning. These stages are described in greater detail below.

2.3.1.1 Stage 1: Research Questions

At this stage, the initial research questions establishing the foundation for the entire review are developed. Techniques that acknowledge the distinctive attributes of multi-label data and offer effective mechanisms to address issues of catastrophic forgetting in lifelong learning are analyzed, and measures of performance are evaluated. Thus, the research questions are:

1. Which lifelong learning algorithms have been proposed to deal with multi-label classification?
2. Which metrics and datasets have been used to evaluate the performance of these algorithms?

2.3.1.2 Stage 2: Identifying Relevant Studies

Here, a comprehensive search is undertaken to identify all relevant studies related to multi-label lifelong machine learning. We employed rigorous and exhaustive search techniques to gather a wide range of primary literature from diverse sources, consulting a bibliometric expert to help us craft key search terms. These terms were used in Boolean search queries to establish a search strategy (see Table 2.1). Table 2.1 is organized into three columns, representing different concepts that are logically connected by the operator 'AND'. This dictates that each search query must contain terms from Concept 1, Concept 2, and Concept 3 simultaneously to meet the criteria. Concept 1 includes the terms "multi-label" and

"multi-output". Concept 2 encompasses variations of the term lifelong learning, such as "lifelong", "life long", and "life-long", "continuous", and "continual". Concept 3 is consistent with the term "machine learning". Additionally, the rows are interconnected by the operator 'OR', indicating that any of the variations within a single concept can be used interchangeably. For example, a valid search query might combine "multi-label" (from Concept 1), "lifelong" (from Concept 2), and "machine learning" (from Concept 3). This formulation provided comprehensive and precise search queries, ensuring that the search process was both thorough and relevant search process.

Table 2.1: Development of Search Strategy

	Concept 1	AND Concept 2	AND Concept 3
	Multi-label	lifelong	machine learning
OR	Multi-output	life long	learning
OR		life-long	
OR		Continuous	
OR		Continual	

We utilized Google Scholar as the principal search engine for the retrieval of pertinent academic works. In addition, we retrieved articles from Scopus and Web of Science, but these were identified as duplicates of articles retrieved from Google Scholar, and were therefore excluded from the analysis. To focus our review on the most relevant articles, we restricted our selection to only the first five pages of the Google Scholar search results.

2.3.1.3 Stage 3: Study Selection

In this stage, we applied inclusion and exclusion criteria to filter the retrieved studies. Only those that met the predetermined criteria were retained for further examination. This screening process guaranteed that the selected studies were directly pertinent to the

research questions, eliminating any potential bias or ambiguity. Our primary focus was on papers addressing lifelong learning and multi-label classification in conjunction, in order to align with our main research questions. Table 2.2 shows the inclusion and exclusion criteria for sources in our review, which define the scope and boundaries of our work. The column titled 'Inclusion' highlights the criteria applied when selecting articles, which were mainly English academic sources focusing on multi-label lifelong learning. In contrast, the exclusion criteria removed articles that solely centered on lifelong learning in education, those discussing lifelong learning or multi-label classification independently, non-academic sources or those without a focus on machine learning, and sources which were not available in the English language.

Table 2.2: Inclusion and Exclusion Criteria

Criteria	Inclusion	Exclusion
Population	Papers focusing on multi-label lifelong machine learning	Papers focusing solely on lifelong learning in education
Context	Fusion of both domains	Papers focusing only on lifelong learning or multi-label classification independently
Types of Sources	Academic papers, conference proceedings, and journal articles	Non-academic sources or sources not focused on machine learning
Timeframe	No specific timeframe	Not Applicable
Language	English-language sources	Non-English-language sources

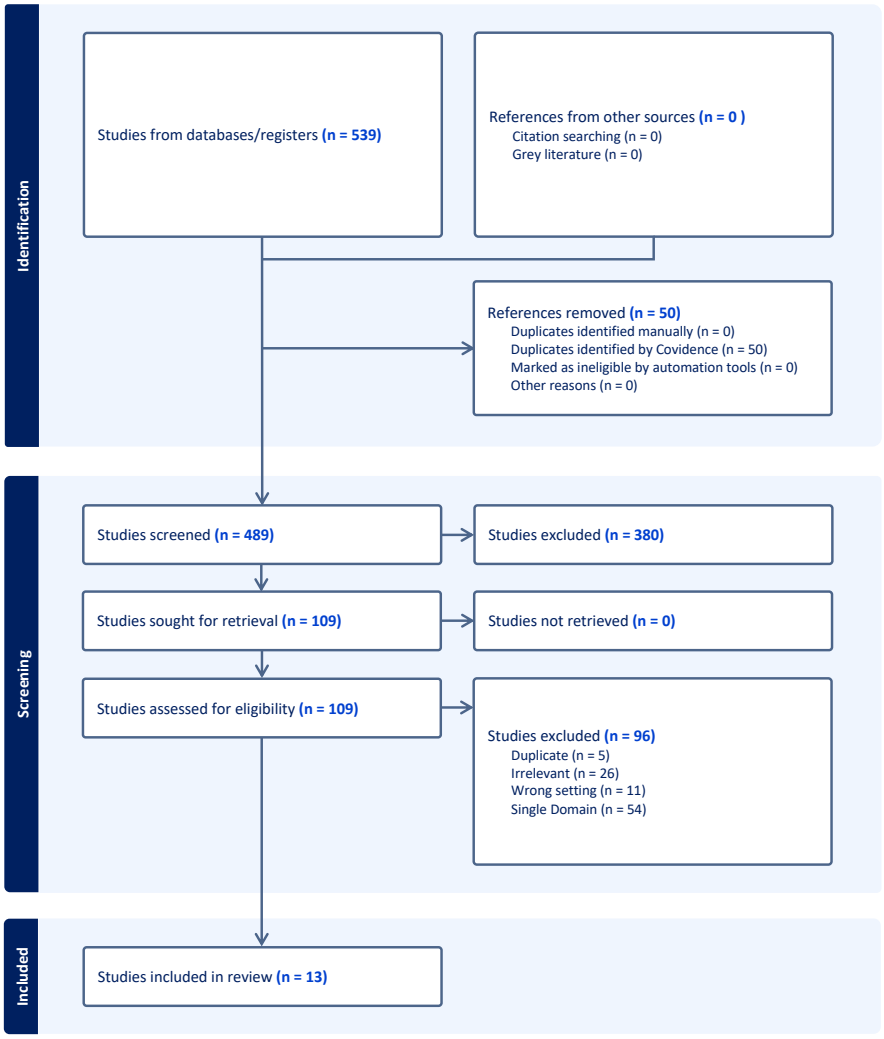


Figure 2.6: PRISMA diagram for scoping review

2.3.1.4 Stage 4: Data Charting and Collation

In this stage, summaries of the retrieved articles were created and analyzed. Table 2.3 provides a brief summary of the studies included in this process. It lists the different

approaches and their underlying algorithms. Each row refers to a specific approach, categorized by the lifelong learning technique employed (i.e Replay, Parameter Isolation, Regularization), the multi-label technique applied (mostly Algorithm Adaptation), the base algorithm used, and a brief description of how the method works.

Table 2.3: Summary of identified multi-label lifelong machine learning algorithms

Reference	Algorithm	LL Technique	Multi-label Technique	Base Algorithm	Brief Description
[59]	MLSAkNN	**	Algorithm Adaptation	KNN	Introduces a self-adapting approach for learning from multi-label drifting data streams
[60]	PRS	Replay	Algorithm Adaptation	Neural Network	Maintains a reservoir that represents a true random sample of the data stream
[61]	OCDM	Replay	Algorithm Adaptation	Neural Network	Treats the updating of its replay buffer as an optimization problem

[62]	MLCA	Parameter Isolation	Algorithm Adaptation	Gaussian Kernel Function	Synergizes Adaptive Resonance Theory (ART) and Bayesian techniques for MLC
[63]	LTM*	Replay	Problem Transformation (BR)	Multiple	lifelong topic modeling approach designed to uncover hidden topics in a text corpus
[64]	BAT-OCDM	Replay	Algorithm Adaptation	Neural Network	Modifies OCDM by introducing separate memory for each task
[65]	AGCN	Parameter Isolation	Algorithm Adaptation	Neural Network	Utilizes an Augmented Correlation Matrix (ACM) to dynamically capture label dependencies
[66]	DFSL	Parameter Isolation	Algorith Adaptation	Neural Network	Few-shot continual learning framework for audio classification

[67]	CPG	Parameter Isolation	Algorithm Adaptation	Neural Network	Deep lifelong learning for defect detection in manufacturing pipelines
[68]	KRT	Regularization	Algorithm Adaptation	Neural Network	Knowledge Restore and Transfer (KRT) tailored for multi-label class-incremental learning
[69]	CIFDM	Parameter Isolation	Algorithm Adaptation	Neural Network	A framework for multi-label stream learning
[70]	CDSH	Regularization	Algorithm Adaptation	Neural Network	Continual Deep Semantic Hashing (CDSH) for learning binary codes of multi-label images
[71]	DLFL	Replay	Algorithm Adaptation	Neural Network	a disentangled label feature learning (DLFL) framework for multi-label learning

** The MLSAkNN model operates by storing previously encountered examples, utilizing these instances during the inference phase. The base algorithm (kNN) is non-parametric, and 'learning' in the traditional sense does not occur. Consequently, categorizing MLSAkNN under any of the three commonly-recognized learning techniques would be a misrepresentation, given its distinct operational mechanism that diverges from standard parametric learning models.

2.3.2 Stage 5: Summary of Findings

In this section, we focus on articles that directly address our research questions. Of the 109 papers assessed for eligibility, 13 presented lifelong learning algorithms in a multi-label setting. Some of these benchmarked their models on openly available datasets. The PRISMA diagram for the review is shown in Fig. 2.6

2.3.2.1 Techniques and Algorithms

Here, as a direct answer to Research Question 1, we describe the techniques and algorithms used by papers' authors. Roseberry et al. introduce a self-adapting algorithm which learns from multi-label drifting data streams [59]. Their study focuses not only on learning continuously from multi-label data, but also on adapting to different concept drifts. As the authors establish, the problem of learning from multi-label data is further exacerbated when the statistical properties of the data change over time as a result of drift. Traditional learning methods tend to struggle to maintain accuracy and adapt to evolving data distributions. The proposed approach leverages the k Nearest Neighbors (kNN) algorithm, which is known for its simplicity and effectiveness in performing classification tasks. The

kNN algorithm uses the notion of proximity when making predictions by comparing the test instance to the k closest examples in the training set. However, unlike traditional kNN, the proposed method introduces a self-adjusting mechanism that dynamically adapts the value of k based on the properties of the incoming data. This adaptive nature allows the model to respond effectively to concept drift and changing label distributions, enabling continuous learning in dynamic environments. Moreover, the conventional k Nearest Neighbors (kNN) algorithm typically employs a criterion such as majority voting to assign a class to a test instance. However, this method encounters limitations in the context of multi-label scenarios, as individual neighbors may possess distinct label sets. To determine which labels to assign to a given test instance, the algorithm tallies the frequency of each label's occurrence within the closest neighbors identified from the training dataset. Employing this count, it proceeds to compute the likelihood and posterior probability associated with each label, utilizing Bayesian principles. Label presence or absence is determined by these probabilities.

Masuyama et al. combine the principles of Adaptive Resonance Theory (ART) [72] and the Bayesian technique for label probability computation to effectively group instances with similar label patterns into clusters, thereby identifying relations within the data and reducing the size of the output space to be learnt [62]. ART is a theory of cognitive information processing that underpins the development of some neural networks. ART-based algorithms attempt to solve the stability-plasticity dilemma by being competitive and self-organizing. By leveraging ART, the approach devised by Masuyama et al. adaptively and continually generates prototype nodes corresponding to the given data, and the generated nodes are used as classifiers. Meanwhile, a Bayesian approach is used to independently track label occurrences for each label and compute corresponding probabilities, in a similar manner to [59]. As a result, each classifier outputs a set of probabilities that

is used to determine label occurrence and hence to simultaneously assign multiple labels to a given instance. The authors use a probability threshold of 0.5, with values lower than this threshold considered to indicate the absence of a label. This approach enables the effective handling of a growing number of labels, ensuring the algorithm's adaptability and scalability in multi-label scenarios. The proposed algorithm, called MLCA, can learn continuously and exhibits competitive classification performance on synthetic and real-world multi-label datasets.

The Augmented Graph Convolutional Network (AGCN) [65,73] approach has been proposed to solve the problem of catastrophic forgetting of old classes when training a model on data with different partial labels in image recognition problems. The proposed method builds an Augmented Correlation Matrix (ACM) across sequential partial-label tasks and captures label dependencies with a dynamic augmented structure to yield effective label representations. This facilitates knowledge transfer and adaptation across different image recognition tasks in a lifelong learning setting. By incorporating graph convolutional networks with a relationship-preserving loss function, AGCN effectively captures the relationships and dependencies among labels and images, enabling accurate and efficient multi-label recognition.

Wang et al. introduce a few-shot continual learning framework for audio classification [66]. Few-shot learning allows for the recognition of novel classes based on only a few labeled data at the time of inference. By efficiently utilizing small amounts of labeled data during inference, this approach enables fast and efficient model updates. The approach is an adaptation of the Dynamic Few-Shot Learning (DFSL) framework, which uses a CNN classifier to extract features and an attention mechanism to exploit past knowledge. To train a model capable of predicting multiple concurrent classes, the categorical cross entropy loss in DFSL is replaced with binary cross entropy. This shift to binary cross

entropy loss is crucial, as it allows the model to evaluate each class label as an independent binary classification problem. In essence, rather than predicting a single label from a set of mutually exclusive labels, the model assesses the presence or absence of each class label independently.

Kim et al. focus on alleviating catastrophic forgetting in neural networks when learning from imbalanced datasets by proposing a technique called Partition Reservoir Sampling (PRS) [60]. This approach is a modified version of the Reservoir Sampling technique, which is widely used to efficiently store a stream of data and sample it. Reservoir Sampling maintains a reservoir of elements that represent a true random sample of the data seen so far in the stream. As the stream flows, each element in the reservoir has an equal probability of being replaced. In PRS, the authors extend this method to create balanced training partitions that ensure each mini-batch contains a representative sample of both majority and minority classes before it is rehearsed. It achieves this by caching a running statistic of all observed examples and uses label frequencies to set the target proportion of classes in memory. This technique allows the model to learn from imbalanced data effectively and mitigates the impact of class imbalance during lifelong learning.

Optimizing Class Distribution in Memory (OCDM) [61] is a memory-based technique that dynamically maintains a representative distribution of classes in memory, in a similar manner to [60]. As this is a rehearsal based technique, a small amount of previously seen data is stored in a replay buffer. Unlike [60], the proposed OCDM method formulates the memory update mechanism as an optimization problem. All the observed mini-batches of data are first added directly into memory until it is full. When a new observation is introduced, the memory is updated such that the new distribution of samples is closest to a given target distribution. This is achieved by minimizing the distance between the two distributions (the target distribution and the distribution of the selected dataset),

measured using a metric such as KL divergence.

Pham et al. propose a lifelong topic modeling approach to facilitate the discovery of hidden topics in a text corpus by exploiting prior domain knowledge [63]. This approach uses a probability-based close domain metric to select valuable knowledge that a model has learnt from the past. This is used to produce more topics associated with the current domain. The proposed metric measures the closeness of two domain datasets, which is then used to select data that enhance the current task's learning. Knowledge of hidden topics is derived from the closeness of the domains. The approach enables the use of this knowledge to enrich features for a multi-label text classifier. The multi-label learning algorithm proposed by Zhang et al. [74] is used. This comprises two main steps. First, for each label in the dataset, the algorithm conducts a clustering analysis on both the positive and negative instances associated with that label. Through this analysis, it constructs features that are specific to each label, ensuring that the characteristics unique to each label are captured. In the second step, it uses these label-specific features to develop a set of binary classifiers, each tailored to the distinct attributes of a different label.

BAT-OCDM [64] investigates Domain Incremental Learning and presents a scenario for lifelong learning whereby a model adapts to handle a stream of machines with distribution shifts. Domain Incremental Learning refers to a lifelong learning scenario whereby the set of labels in the output do not change across tasks. Instead, changes occur in the distribution of the input data from one task to the other [64]. The proposed approach modifies the OCDM algorithm [61] by using a separate memory for each task. This is done to ensure balance on both labels and tasks, as OCDM does not guarantee the retention of all previously seen tasks. Tests on real data sourced from the packaging industry are conducted to demonstrate the feasibility of addressing this significant problem. The goal is to predict a list of distinct and mutually inclusive alarms that are likely to occur in the

future. This is modeled as a multi-label classification problem.

Chen et al. propose the use of deep lifelong learning (learning with densely connected neural networks) for defect detection in manufacturing pipelines [67]. The approach allows the model to learn to detect new defect types while maintaining its ability to detect old defect types without retraining on previous data. Each task is formulated as a binary classification problem for each defect type. When a new defect type is discovered in the manufacturing process, a new binary classification task is formulated by collecting the dataset for the defect. The proposed Compact, Picking and Growing (CPG) algorithm then learns the new task. The CPG algorithm's learning process involves identifying crucial weights within the pre-existing deep neural network model learned from past tasks, compacting the model to free up weights for upcoming tasks, and enlarging the network's size if the performance target has not been met.

Knowledge Restore and Transfer (KRT) [68] is a framework tailored for multi-label class-incremental learning (MLCIL). This framework incorporates two key modules: the Dynamic Pseudo-Label (DPL) module, which restores knowledge from old labels, and the Incremental Cross-Attention (ICA) module, which is designed to preserve task-specific knowledge and transfer old knowledge to the new model. Through the application of this proposed method, the authors report significant improvements in recognition performance and effectively mitigate the issue of forgetting in multi-label class-incremental learning tasks.

Continual and interactive feature distillation for multi-label stream learning (CIFDM) [69] is a proposed framework in which new labels emerge continuously in changing environments and are assigned to previous data. The framework utilizes knowledge from previous tasks to learn new knowledge and avoid catastrophic forgetting, and consists of three components: an interactive knowledge compression function, a knowledge bank, and a pioneer

module. The compression function compresses and transfers new knowledge to the bank. The knowledge bank stores the compressed knowledge along with its associated label set. When a new task with novel labels is encountered, the knowledge base comes into play, initializing a pioneer module which is intended to learn the new information from incoming examples.

Song et al. propose a method called Continual Deep Semantic Hashing (CDSH) for learning the binary codes of multi-label images with increasing labels [70]. This method consists of two hashing networks, one for hashing the increasing semantics of data into semantic codes and the other for mapping images to the corresponding semantic code. CDSH incorporates empirically verified loss, and a special regularization design to ensure that old labels remain unchanged during encoding. The authors also theoretically demonstrate that their method improves the probability of the old data's code remaining unchanged after the model is updated.

Jia et al. propose a Disentangled Label Feature Learning (DLFL) framework to learn a disentangled representation for each label [71]. The framework introduces the One-Specific-Feature-for-One-Label (OFOL) mechanism to address the limitations of the One-Shared-Feature-for-Multiple-Labels (OFML) mechanism commonly used in multi-label classification. The framework includes a feature disentanglement module, which contains learnable semantic queries and a Semantic Spatial Cross-Attention (SSCA) sub-module. The SSCA sub-module localizes the label-related spatial regions and aggregates located region features into the corresponding label feature to achieve feature disentanglement.

While all these approaches have been proposed to handle multi-label data, some of the mechanisms they use to identify interrelationships among labels and assign multiple labels per instance are not presented transparently. For example, Kim et al. [60] and Liang et al. [61] emphasize sample selection and dealing with imbalance, but fall short of providing

practical insights into how the multi-label problem is solved. It is crucial to note that both methods are rooted in neural network architectures, as the choice of activation and loss functions within these networks is key. These functions are tailored to optimize distinct objectives, and their selection can substantially influence outcomes — potentially leading to results that could be misinterpreted if not contextualized correctly. In addition, it is obvious from Table 2.3 that the majority of the multi-label lifelong learning algorithms (ten out of thirteen) use neural networks as their base algorithm. For these algorithms, it is reasonable to expect that the output size matches the number of labels, with each neuron generating a prediction for each label. In terms of the activation function, a common approach is to produce separate probabilities for each neuron; one example of this is the coupling of the sigmoid function with binary cross entropy loss, as employed by Wang et al. [66]. Finally, although some of the methods discussed have been used for specific applications such as equipment monitoring [64], audio classification [66] and fault detection [67], it is important to note that the core algorithm in each case is not limited to any specific domain. Each of these core algorithms can be applied to any dataset with the same modality as those described in their respective studies.

2.3.2.2 Evaluation Metrics and Datasets

To answer our second research question, we identified the metrics and datasets used to evaluate the algorithms discussed earlier. None of the works we have reviewed propose new metrics specifically for the evaluation of the methods under consideration. Existing assessment metrics encompass either lifelong learning metrics, multi-label metrics, or a fusion of both. Consequently, we now proceed to examine these metrics.

Multi-label Metrics

To accurately assess a multi-label classification model's performance, it is essential to understand its capabilities and limitations. To achieve this, a diverse range of evaluation metrics are used, each serving a specific purpose in evaluating the model's overall effectiveness in handling multiple labels.

Tsoumakas et al. propose two distinct categories of metrics for evaluating multi-label learning methods: example-based metrics and label-based metrics [28] . Example-based metrics are calculated for each test example and then averaged across the entire test set. In the label-based approach, individual metrics are computed for each label based on true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), and then averaged to obtain an overall value. This averaging can be performed using either the macro method (shown in (2.2)) whereby metrics are calculated for each label and then averaged across all categories, or the micro method (shown in (2.3)), which considers predictions for all instances together and calculates the measure across all labels by aggregating TP, TN, FP, and FN values.

$$\text{Macro } B = \frac{1}{N}\sum_{i=1}^{N} B(TP_i, TN_i, FP_i, FN_i) \tag{2.2}$$

$$\text{Micro } B = B\left(\sum_{i=1}^{N} TP_i, \sum_{i=1}^{N} TN_i, \sum_{i=1}^{N} FP_i, \sum_{i=1}^{N} FN_i\right) \tag{2.3}$$

where,

$$N = \text{Number of labels}$$

$$B = \text{Any binary evaluation measure}$$

From the above formulas, a micro and macro average can be calculated for metrics such as Recall, Precision, Receiver Operating Characteristic (ROC), Area Under Curve (AUC), Accuracy, and F1 score. In addition to these metrics, in Table 2.4 we provide brief definitions of other evaluation metrics discussed in the literature [9] [39] and [40]. These metrics are included here in order to provide readers with an understanding of how the performance of such algorithms is quantified and compared.

1. **Exact Match Ratio** computes the proportion of instances in which all predicted labels match the true labels exactly. It provides a more stringent evaluation metric than subset accuracy as it requires all labels to be predicted correctly for an instance to be considered correctly classified.

2. **Hamming Loss** measures the fraction of misclassified labels per instance by comparing predicted labels with true labels.

3. **Ranking Loss** evaluates the ranking quality of the predicted labels compared to the true labels.

4. **Jaccard Index** measures the similarity between predicted and true label sets using the ratio of their intersection to their union. This assesses the model's ability to handle label sets with varying degrees of overlap.

5. **Subset Accuracy** evaluates the ratio of the total number of instances where all predicted labels are a subset of the true labels to the number of instances in the dataset. It is a less strict metric than the Exact Match Ratio.

Table 2.4: Multi-label evaluation metrics

Metric	Brief Description	Equation
Exact Match Ratio	Proportion of instances in which all predicted labels match the true labels exactly	$\frac{1}{n}\sum_{i=1}^{n} Z_i = Y_i$
Hamming Loss	Fraction of misclassified labels per instance	$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{q}\left\|Z_i \Delta Y_i\right\|$
Ranking Loss	Ranking quality of the predicted labels compared to the true labels	$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{\left\|Y_i\right\\|\overline{Y_i}\right\|}\|E\|$
Jaccard Index	Measures the similarity between predicted and true label sets	$\frac{1}{n}\sum_{i=1}^{n} \frac{\|Y_i \bigcap Z_i\|}{\|Y_i \bigcup Z_i\|}$
Subset Accuracy	Ratio of instances where all predicted labels are a subset of the true labels to total number of instances	$\frac{1}{n}\sum_{i=1}^{n} Z_i \subset Y_i$

Notation for Multi-label Evaluation Metrics
n: Number of instances in the dataset.
Z_i: Set of labels predicted for the i-th instance.
Y_i: Set of true labels for the i-th instance.
q: Total number of possible labels.
$|\cdot|$: Cardinality of a set.
Δ: Symmetric difference between two sets.
$|Y_i \parallel \overline{Y_i}|$: Total number of label pairs where one label is in Y_i and the other is not, for the i-th instance.
E: Set of incorrectly ordered label pairs.
$\bigcap$: Intersection of two sets.
$\bigcup$: Union of two sets.
$\subset$: Subset relation.

The Exact Match Ratio has an all-or-nothing approach, which is useful in scenarios where label accuracy is paramount. However, its strict nature might not be suitable for appli-

cations in which partial matches are still informative. It fails to acknowledge the partial correctness of the predictions, which can be useful in many real-world scenarios. While subset accuracy and Hamming loss are less strict, critical misclassifications may be overlooked. In cases where missing certain labels is a more critical problem than misclassifying others (i.e., where there are important labels that should not be missed), these metrics are not particularly helpful.

Lifelong Learning Metrics

The more frequently used metrics [44] for assessing the quality of lifelong learning methods are as follows:

1. **Forgetting** measures the extent to which an algorithm loses its performance on previously learned tasks while learning new ones. It evaluates the preservation of knowledge acquired during earlier tasks and indicates the potential interference between new and old knowledge during model updates.

2. **Forward Transfer** measures the impact of previously learned tasks on the performance of new tasks. It measures whether knowledge learned from earlier tasks has positive effects on learning new tasks.

3. **Backward Transfer** assesses the influence of new tasks on the performance of previously learned tasks. It measures whether learning a new task improves or degrades performance on earlier tasks.

4. **Interference** measures the extent to which the acquisition of new knowledge hinders or interferes with the retention and adaptation of existing knowledge, leading to a degradation of performance on earlier tasks.

Some methods, including [71], [68], [61], [60], and [65], are evaluated using both micro and macro averaging techniques in addition to mean average precision (mAP). Specifically, the overall precision (OP), overall recall (OR) and overall F1 (OF1) micro averages are determined. For the macro averages, per-class precision (PC), per-class recall (CR), and per-class F1 (CF1) are used. No attempts are made to assess the model's ability to learn new tasks without forgetting, or to transfer past knowledge to current tasks. Some researchers [59, 66, 67, 69] further limit the scope of evaluation by using either a macro or micro averaging technique, rather than both. Dalle et al. combine both multi-label metrics and lifelong metrics [64]. While such a combination offers valuable insights into certain aspects of trained models, it still falls short of comprehensively capturing the intricate dynamics associated with learning multiple labels across sequential tasks.

Datasets

Various datasets have been used to assess the effectiveness of the algorithms reviewed in this chapter. Table 2.5 lists the datasets most commonly used in the retrieved articles. These include MSCOCO, NUS-WIDE, Yeast, PASCAL-VOC, MIR-FLICKR25K, DAGM, ESC-50, AudioSet, and ALPI, each with a distinct label count and modality (image, tabular, or audio). MSCOCO and PASCAL-VOC, for example, are image datasets used in object detection and scene segmentation, while Yeast is a tabular dataset derived from gene expression data in yeast cells. The table also includes audio datasets like ESC-50 and AudioSet, which focus on environmental sounds and a wide range of real-world audio clips, respectively. The usage proportion of the identified datasets, as well as their modalities, are also shown in Fig. 2.7 and Fig. 2.8 respectively. The bar chart shows the count of different data modalities used across the reviewed literature. We observe that imagery is the most frequent data modality with a count of 5, indicating that images are the most

commonly used type of data in this context. Audio follows with a count of 3, suggesting that it is also significant but less frequently used than images. Tabular data has a count of 1, showing that it is present in the literature but not as prominently utilized as image or audio data. Lastly, the 'others' category also has a count of 1; this category includes any data modalities that do not fall into the first three categories.

Table 2.5: Summary of datasets widely used for multi-label classification

Dataset	Label Count	Modality	Brief Description
MSCOCO	80	Image	MS COCO (Microsoft Common Objects in Context) is a widely used large-scale dataset for object detection, segmentation, and captioning tasks in computer vision. It contains a diverse collection of images, each annotated with multiple object instances and their corresponding labels
NUS-WIDE	81	Image	A multi-label dataset that consists of a diverse collection of images, each associated with multiple relevant labels. The dataset covers a wide range of object categories and scenes, making it suitable for various real-world applications

Yeast	14	Tabular	This dataset is derived from gene expression data and contains information about the presence or absence of some functional classes in yeast cells. Each data instance corresponds to a yeast gene, and the labels represent the functions associated with the gene
PASCAL-VOC	20	Image	The PASCAL-VOC (Visual Object Classes) dataset is a widely used benchmark dataset, just like COCO. It consists of images from various real-world scenes
MIR-FLICKR25K	24	Image	This multi-label dataset is mostly used in the field of Multimedia Information Retrieval (MIR). It presents a challenging and diverse set of images with a wide range of visual content and semantic labels
DAGM	10	Image	Algorithm Adaptation
ESC-50*	50	Audio	This dataset consists of environmental audio recordings comprising animal sounds, human sounds, and background noise

AudioSet	527	Audio	A large collection of human-labeled sound clips that provides comprehensive coverage of real-world sounds
ALPI	154	Audio	A sequence of alarms logged by packaging equipment in an industrial environment

Note: The actual datasets used in the multi-label lifelong experiments are sequentialized versions of the original.

*Originally single labeled, and then converted to multi-label for comparative analysis [66]

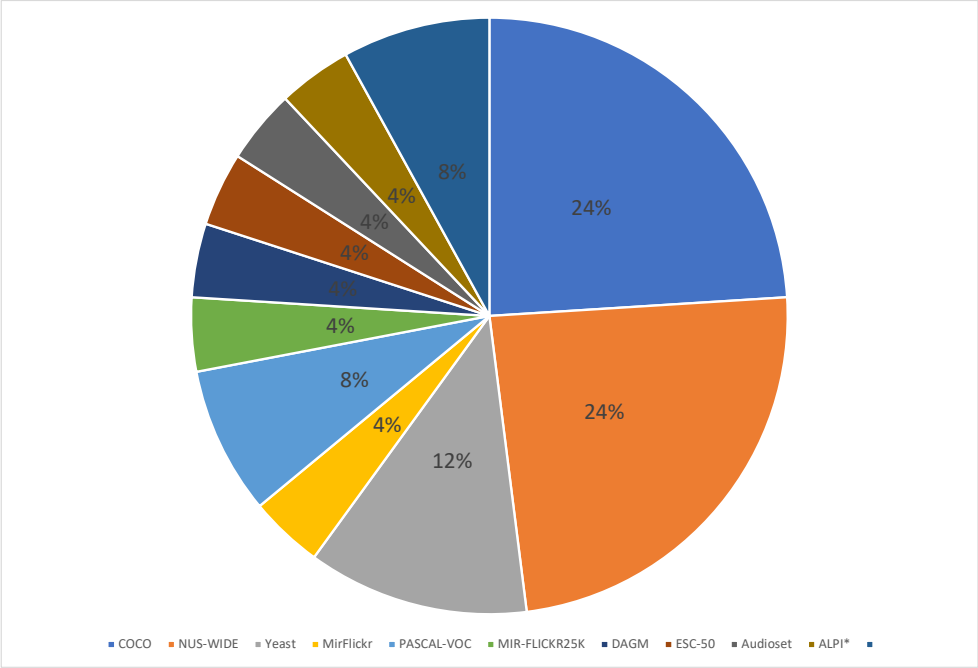


Figure 2.7: Proportion of datasets and their usage in the reviewed literature

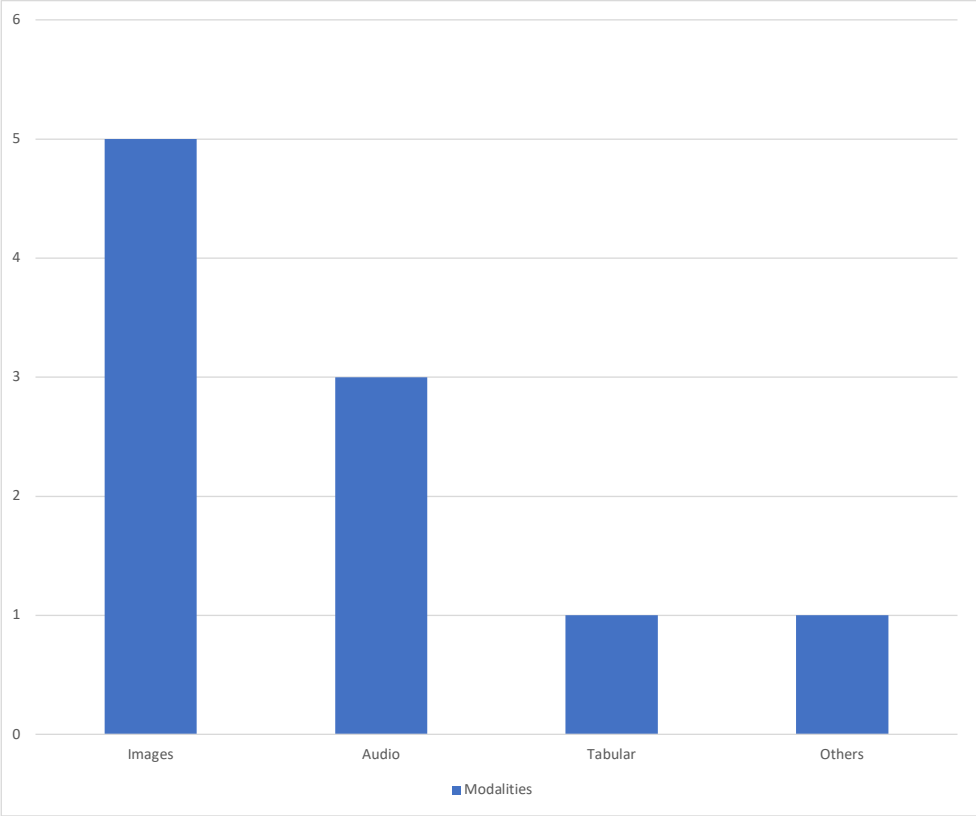


Figure 2.8: Modalities of the multi-label datasets. The vertical axis represents the number of datasets with the given modality

2.3.2.3 Applications and Use-Cases

Multi-label lifelong learning has a wide range of practical applications that capitalize on its ability to continuously adapt to evolving data and label distributions. Chen et al. present an interesting approach for detecting defects in manufacturing pipelines [67]. In traditional defect detection methods, models are trained to identify specific types of defects. However, as the manufacturing process develops, new types of defects may emerge that are not mutually exclusive with older defects. Models trained on older defect types struggle

to identify these new types, leading to inefficiencies. Multi-label lifelong learning allows classifiers to identify old types of defects while incrementally learning to detect new ones, even when they occur simultaneously. In large-scale image retrieval, images are converted into compact binary codes known as 'hashes' to facilitate efficient storage and searching. Song et al. present a useful approach for the lifelong hashing and retrieval of multi-label images [70]. The applications presented by Dalle et al. [64] and Wang et al. [66] can be broadly categorized under audio classification. The former focuses on monitoring equipment in a manufacturing setting, specifically on keeping track of alarms emitted by various machines. Multi-label LML appears to be a viable method for addressing challenges in the following established domains:

1. **Medical diagnostics and healthcare** In the medical field, patient health data can be dynamic and multi-dimensional, and often requires multiple labels for accurate diagnosis and treatment planning. Multi-label lifelong learning can be leveraged to build robust medical diagnosis systems that adapt to changing patient profiles, integrate new medical knowledge, and make accurate predictions across various medical conditions.

2. **Environmental monitoring** Environmental monitoring involves the analysis of diverse parameters such as air quality, water pollution, and climatic conditions. Multi-label lifelong learning can be applied to build predictive models that continuously update to capture environmental changes, making it possible to provide real-time warnings which facilitate proactive measures to address ecological challenges.

3. **Finance and trading** Multi-label lifelong learning enables trading algorithms to incorporate new patterns and market trends without compromising knowledge of historic market dynamics.

4. **Robotics and autonomous systems** Robotic systems can continuously learn from real world experiences, allowing them to operate in dynamic and changing environments.

5. **Education** Educational platforms can progressively tailor contents based on the individual learner's needs.

6. **Smart homes** Lifelong learning techniques can enable IoT devices in smart homes to continuously learn and adapt to the resident's preferences.

7. **Cybersecurity and fraud detection** Lifelong learning can facilitate the learning of new attack patterns, identification of emerging threats, and the updating of defense mechanisms to enhance protection against evolving cyber threats.

8. **Transportation** Intelligent Transportation Systems and autonomous vehicles can continuously learn from traffic patterns, adapt to changing road conditions, and improve navigation.

It is important to note that this list of domains and applications, while not exhaustive, provides examples which highlight the versatility and potential impact of multi-label lifelong learning across different sectors.

2.3.3 Discussion

Multi-label lifelong learning algorithms emerged from the multi-label classification and lifelong learning domains. Leveraging the strengths of both fields, these algorithms effectively navigate the complexities of handling multi-label data while accommodating lifelong learning scenarios.

The research landscape in the field of lifelong learning has predominantly emphasized continual learning, with a major focus on addressing catastrophic forgetting. However, lifelong learning encompasses a broader spectrum of challenges and scenarios, including cumulative learning across multiple tasks and domains. This highlights the need for a more balanced exploration of the various aspects of lifelong learning, considering such aspects as task identification and knowledge transfer over extended periods — features often associated with real-world lifelong learning applications.

Intuitively, providing solutions to real-world problems must involve both data and algorithms. As such, the methods presented here are not always used in isolation. An end-to-end solution would typically involve one or more of the two broad categories of problem transformation and algorithm adaptation. With regard to algorithm adaptation, traditional ML algorithms require substantive modification to work well with multi-label datasets. Neural networks, on the other hand, are easy to modify and adapt for multi-label problems. However, they fall short when it comes to explainability, which is a key requirement in many applications. Finally, we learnt that imbalance is an inherent property of most multi-label datasets, and as such most multi-label classification algorithms tend to include mechanisms for handling imbalance.

Lifelong machine learning represents an exciting research paradigm that is essential for the creation of AI systems that can adapt and learn continuously in dynamic environments. The lessons learned from this research shed light on the importance of mitigating catastrophic forgetting through the employment of memory-augmented models or the adaptation of architectures to enhance lifelong learning performance. As the field continues to advance, addressing scalability challenges and exploring real-world applications will be critical for unlocking the full potential of lifelong machine learning in various domains.

2.4 Conclusions and Future Challenges

This chapter presents a scoping review of the algorithms, methods, and metrics used in lifelong learning within the context of multi-label classification. Through a scoping review framework, we identified areas of overlap and common ground between these two domains. By leveraging Arksey and O'Malley's scoping review methodology, we have systematically gathered a wide range of relevant literature, enabling us to gain new insights into the state of research in this field. Our review uncovers several knowledge gaps in the existing literature. First and foremost, with regard to application, we noticed an under-exploration of the potential of generative replay techniques for handling multi-label data. Generative replay methods that use generative models such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have demonstrated their effectiveness in alleviating catastrophic forgetting in sequential learning tasks [2] by synthesizing and replaying past data samples during model updates. However, the current body of research predominantly focuses on exemplar replay (refer to Table 2.3), which necessitates memory buffers, leaving a substantial gap in terms of investigation of the applicability and potential challenges of generative replay in multi-label learning scenarios. The integration of generative replay into the multi-label context remains largely unexplored. In addition, existing multi-label lifelong learning algorithms often rely on modifying current standalone datasets to simulate various scenarios, such as class incremental or domain incremental learning. However, this approach may not fully capture the complexities and nuances of real-world environments. The effectiveness and generalizability of these algorithms in practical, dynamic settings remains uncertain. Furthermore, the absence of specific evaluation metrics tailored for multi-label lifelong learning has led to the utilization of standalone metrics in the existing literature. This remains a significant research challenge.

These identified gaps present exciting opportunities for future research to significantly advance the field and drive the development of more effective lifelong learning algorithms in multi-label settings. A promising future direction for multi-label lifelong learning research lies in the exploration and adaptation of generative replay techniques specifically for multi-label data, as will be discussed in the next chapter. With regard to datasets, it is essential to develop approaches that enable lifelong learning models to interact and adapt to real world scenarios directly, as opposed to using modified handcrafted static datasets. This includes devising strategies to incorporate real-world data streams, dynamic changes, and varying contexts into the learning process. This is a promising research direction with the potential to help us better assess the performance, robustness, and applicability of these algorithms in ever-evolving scenarios. Another promising research direction is the development of dedicated evaluation metrics that can effectively address the unique challenges and requirements posed by lifelong learning in a multi-label context.

2.5 Summary

In this chapter, we present our scoping review conducted to explore multi-label lifelong machine learning, identifying key algorithms, techniques, and applications. The review highlighted a significant gap: the limited use of deep generative models in addressing catastrophic forgetting. To fill this gap, we introduce MLDGR (Multi-Label Deep Generative Replay), an approach for learning from multi-label image datasets in a lifelong manner, that extends generative replay. The details of MLDGR will be discussed in the subsequent chapters.

Chapter 3

Multi-label Deep Generative Replay (MLDGR)

While exemplar rehearsal techniques have demonstrated state-of-the-art performance in lifelong learning tasks, several challenges still exist in their practical application. One notable concern is related to memory constraints, as the accumulation of past experiences for replay can become impractical in resource-limited environments [5]. Additionally, privacy issues arise in scenarios where sensitive information is embedded within the training data [2]. The preservation and utilization of such data during replay may compromise privacy.

Drawing inspiration from the generative capabilities observed in the hippocampus, a short-term memory system in primate brains, and addressing the privacy issues associated with exemplar rehearsal, Shin et al. introduced Deep Generative Replay [2]. This framework features a cooperative dual model architecture known as the scholar model, comprising a generator and a task-solving model known as the solver. Our work explores

learning from multi-label image datasets by extending the scholar model. Our proposed framework, which we call MLDGR comprises a solver designed to learn from multiple labels and a generator tasked with generating text descriptions from input images during training. In the course of learning a new task, the model utilizes the text descriptions from preceding tasks to generate synthetic image samples, which are seamlessly interleaved with the actual data pertaining to the current task. This interleaved training approach is aimed at enhancing the model's ability to adapt to new tasks while simultaneously retaining the knowledge gained from prior learning experiences.

To fully understand how the proposed method works, it's essential to break it down into its basic parts and understand what each part does before looking at how they work together. Each piece, like the multi-label solver and the generator that transforms images into text and back to images, has its own complexities that need a careful look. Examining these parts separately helps us understand how they work and where they might have limitations. We define these components as follows.

Definition 3.1 (The Scholar Model)**.** *The scholar model is a dual neural network architecture comprising a task solving model known as the solver and another neural network for generating synthetic data referred to as the generator [2].*

Definition 3.2 (Solver)**.** *The solver is the component of the scholar model responsible for learning from data across various tasks and assigning labels to test examples [2].*

Definition 3.3 (Generator)**.** *The generator is a key component of the scholar model that creates synthetic data samples representative of past tasks [2]. These samples are then integrated with the current task's data to train the solver.*

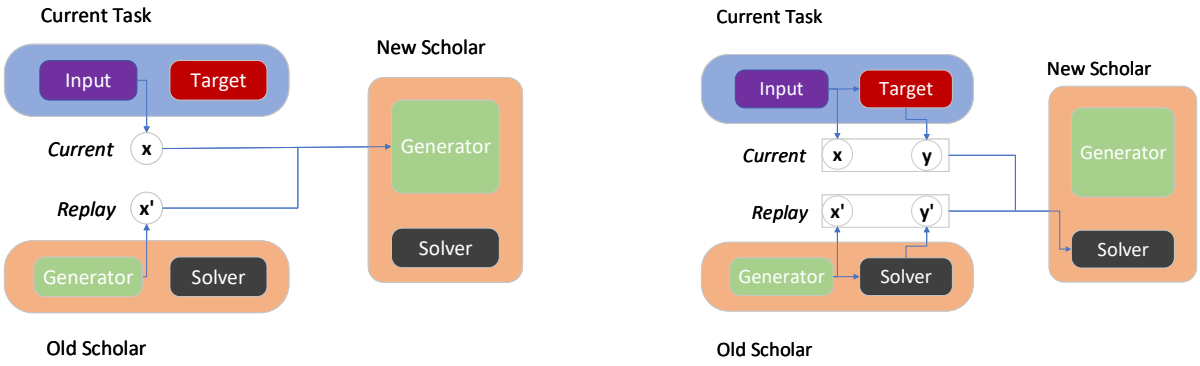
This chapter is organized as follows. In Section 3.1, we provide a brief overview of the scholar model and explain how we extend this model to accommodate multi-label lifelong

learning tasks. In Section 3.2 we explore the architecture of our multi-label solver in detail, covering aspects such as feature extraction from input images, activation functions, loss functions, as well as training and optimization strategies. Section 3.3 is dedicated to discussing various generator architectures for creating synthetic data samples, highlighting their respective advantages and disadvantages, and explaining our rationale behind the selection of a particular generator for our model. The chapter concludes in Section 3.4 with a summary of the work presented in this chapter.

3.1 The Scholar Model

The method introduced by Shin et al. is known as the "scholar model" because of its remarkable capability to acquire new knowledge without forgetting previous tasks and to teach other models by generating input-output pairs as examples [2]. This enables learning in a sequential manner without relying on past data. Instead of referring to historical data, this framework uses a generator trained to create synthetic data that resembles data from previous tasks as shown in Figure 3.1. The generated data is then paired with the corresponding labels assigned by the solver. When faced with a new task, the scholar model combines the synthetic data and the corresponding labels with the new data to update the generator and solver.

While we use this strategy as the foundation for lifelong learning, we have made notable adjustments. Firstly, we used various generator designs and added a feature to retain information about past tasks in the form of text. Additionally, we've developed a solver that can assign multiple labels to each example with state-of-the-art precision. We will delve into these changes in more detail in the upcoming sections. The workflow for our improved model is shown in Figure 3.2



(a) Training Generator (b) Training Solver

Figure 3.1: Training the scholar model [2]

In MLDGR, learning starts with the introduction of the first task in the lifelong learning sequence. The system trains a multi-label solver, designed to handle various outcomes simultaneously. Once this initial training is complete, the system retains the information pertinent to this first task as a foundational knowledge base. This is achieved by generating descriptions from select images which are stored in a buffer. When subsequent tasks in the lifelong learning pipeline are presented, the system takes a different approach. Instead of starting from scratch, it generates samples that represent the tasks it has previously learned and assigns appropriate labels to these samples. These synthetic image samples are generated using the stored image descriptions from the previous task. This helps in creating a more robust dataset by augmenting the new task data (images) with these representative samples. The system then incorporates this augmented data into its learning model. By doing so, it reinforces what it has already learned, thus mitigating the risk of forgetting older information as new tasks are introduced. The learning cycle continues in this manner, with the system checking for the completion of tasks and, if unfinished, looping back to continue the learning process. The process comes to a halt only when all tasks have been learned.

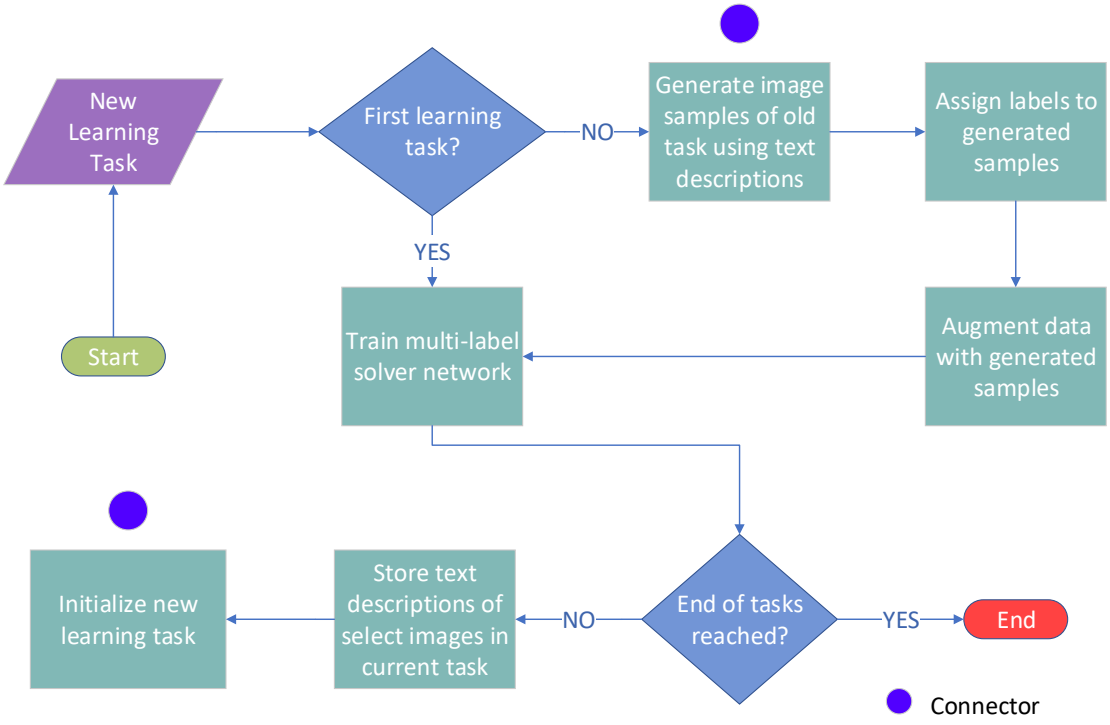


Figure 3.2: Workflow for multi-label deep generative replay

In our work, we focus our attention on image datasets. The novelty of our work is that unlike the approach proposed in [2], our method incorporates a component for storing information on the previous tasks (images) in the form of text. This is because through experiments, we observed that given the size and dimensions of our image datasets (the previously introduced MSCOCO and NUS-WIDE), the output of the generator was not always representative of samples in the past tasks. Figure 3.3 shows a sample image, stored text description, and the generated image based on the stored description. It can be seen that MLDGR identifies the main labels in the image, generate a textual description, and then use this textual description to construct a new representative image. That is, our

novel approach generates a description of the image, noting that it contains a dog and cat standing in a field. Next, the generator uses this description to introduce a different image of another dog and cat standing in a different field. As such, the textual description enables us to introduce novel generated data into the learning process. Details of this generative process is discussed in Section 3.3. Although MLDGR does not have a dedicated component for addressing privacy concerns, its design intrinsically ensures that privacy is maintained by generating synthetic images when learning new tasks, eliminating the need to access real images from previous tasks.

Figure 3.3: Sample stored image, textual description, and a generated image

In the next section, we discuss the architecture of the multi-label solver used in MLDGR. We explain the feature extraction process, and how the activation and loss function of the classifier is configured to enable learning multiple labels simultaneously.

3.2 Multi-label Solver

Recall that, the solver is the component of the proposed framework responsible for learning from the data. Due to the type of data used in this experiment (images) we make use of a solver architecture built with Convolutional Neural Networks (CNNs) [75]. Traditional neural networks, known as feedforward neural networks, process data in a straightforward manner. They take in input, perform a series of mathematical operations, and produce output. They're good for simple tasks but can struggle with complex data like images. Images are made of thousands of pixels and carry a lot of information in the form of colors, shapes, and textures. CNNs are specifically designed to handle this complexity. In addition, images have statistical properties that remain unchanged when translated [76]. CNNs leverage this trait by utilizing the same parameters at various positions within the image.

Since this is a neural network based approach, the key to assigning multiple labels simultaneously is to select a loss function that account for the accuracy of each label. For instance, if a picture contains a cat and a dog, and the model only recognizes the cat, the loss function should reflect the missed dog label. Another important component is an activaition function that treats each output independently.

3.2.1 Architectural Overview

The architecture of the multi-label solver is shown in Figure 3.4. As mentioned earlier, we chose a CNN based architecture due to the ability of CNNs to detect and learn important features from images as well as recognize objects regardless of where they are located in the image (translation invariance). As seen in the Figure, the base of the solver uses MobileNetV2 [77], a CNN based architecture notable for speed and training efficiency.

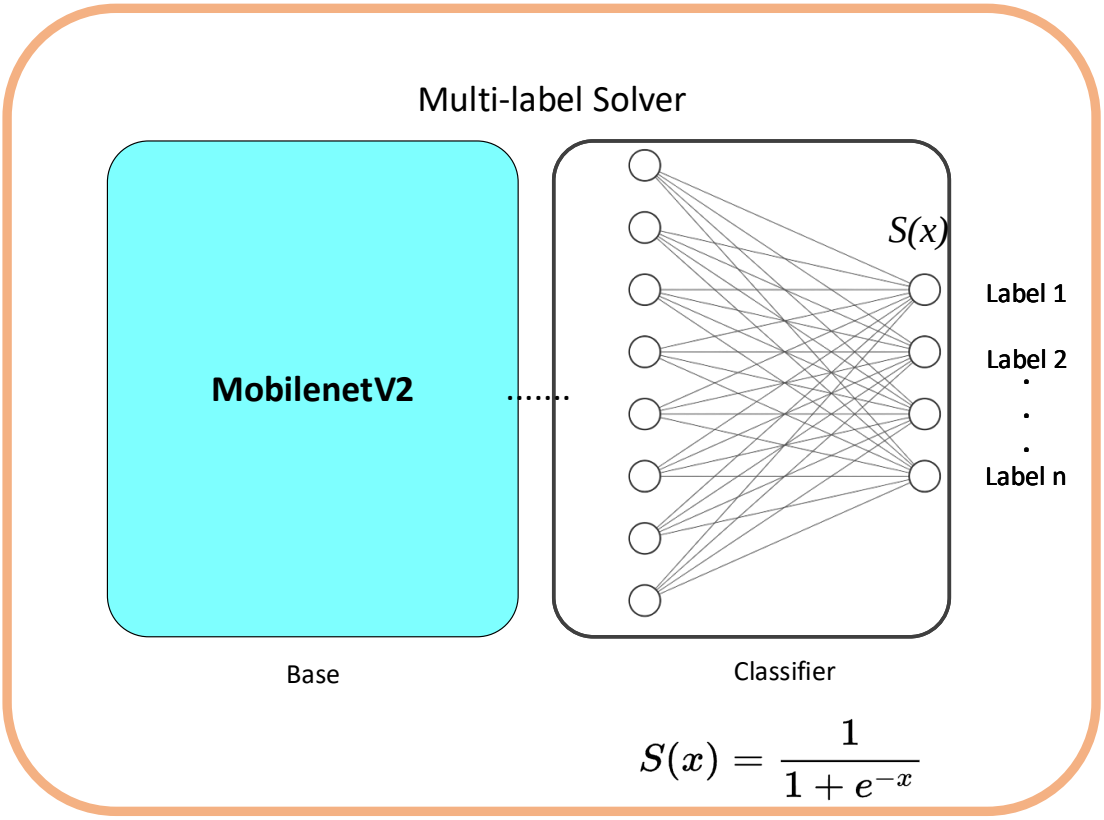


Figure 3.4: Architecture of the multi-label solver

When learning a task, a batch of images is presented at the input layer of the base network. Each image undergoes a transformation from a visual object that we see into a numerical format that the computer can process (see Figure 3.5). The image is divided into a grid of tiny dots called pixels. Each pixel represents the smallest unit of the image and has values corresponding to color channels, typically Red, Green, and Blue (RGB)for colored images. Pixel values range from 0 to 255. A value of 0 means that color is not present at all, while 255 represents the highest intensity of that color. The learning system sees this array of pixel values (a matrix for grayscale images and a 3D array for color images) as numerical

data. Each pixel's color intensity is represented by numbers, and these numbers are what the algorithm works with. The diagram in Figure 3.5 shows the numerical representation of images for processing. The input image is taken from the MSCOCO dataset, but the pixel values are only for illustration purposes and do not represent the actual values stored in the image.

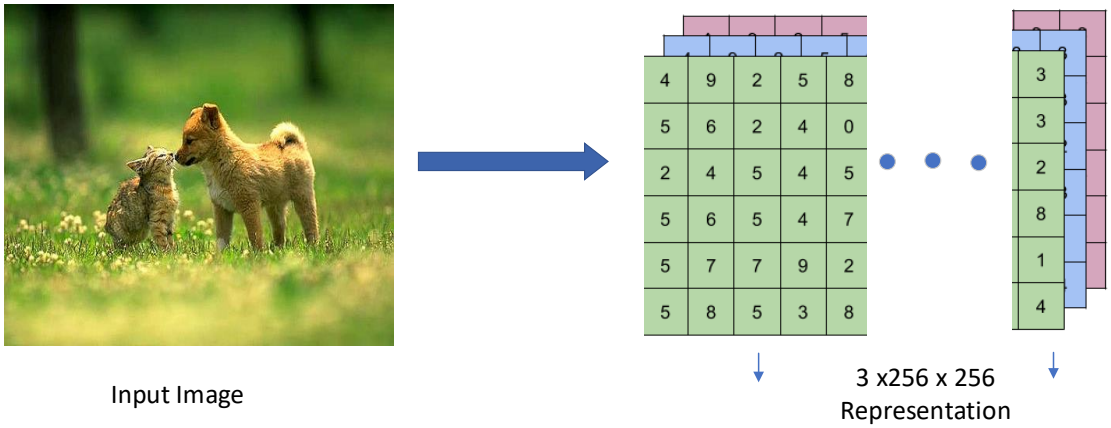


Figure 3.5: Numerical representation of images for processing

When learning from images, traditional feed forward networks (FFNs) consider the pixel values separately, hence neurons in the input layer are connected to every single pixel in the input image [78], essentially flattening the image into a 1D array. This process does not consider the spatial relationships between pixels in the image. These spatial relationships are important as they hold significant information about the structure and features of the image. For instance, consider the image in Figure 3.5. The pixels that make up the dog's ears are not just randomly scattered throughout the image; they are connected in a specific pattern that resembles the shape of an ear. When you flatten the image, the information about which pixels are adjacent or close to each other in a 2D

space (like the pixels forming the contours of the dog's ears or its nose) is lost. This could make it challenging for the network to learn and recognize complex patterns. Unlike FFNs, CNNs maintain the spatial hierarchy of the pixels. They use filters (also called kernels) to perform convolution operations that preserve the relationship between pixels [79]. These filters systematically traverse the image and analyze small chunks of pixels at a time, capturing the local dependencies within the image (like the specific arrangement of pixels that form the dog's ears). By doing so, CNNs are able to recognize and learn patterns with a much higher level of sophistication. They can identify features like edges, textures, and eventually more complex patterns, making them effective for tasks that involve image recognition.

More broadly, convolution is a mathematical operation that expresses the amount of overlap between two real-valued functions [76]. The formula for continuous convolution of two functions $f(t)$ and $g(t)$ is given by:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) \cdot g(t - \tau) d\tau \tag{3.1}$$

This formula essentially says that the convolution of f and g is the integral of the product of f and a reversed and shifted version of g. For discrete values, which are commonly encountered in digital signal processing and image analysis, convolution is defined as:

$$(f * g)[n] = \sum_{m=-\infty}^{+\infty} f[m] \cdot g[n - m] \tag{3.2}$$

This represents the sum over all possible positions of the product of $f[m]$ and a reversed and shifted version of g, just like the continuous case but adapted for discrete data. In the context of a two-dimensional image I and a two-dimensional filter (or kernel) K, the

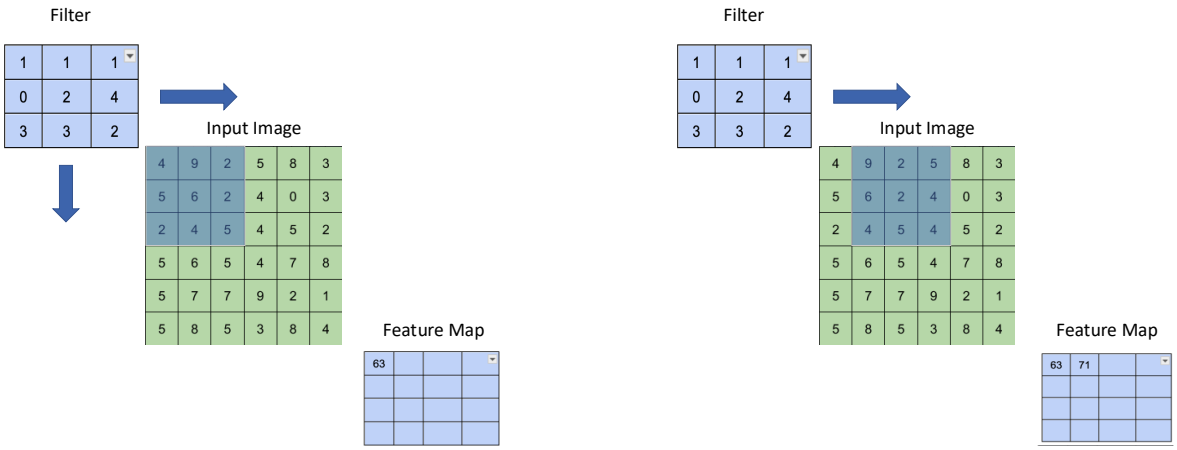
convolution operation is used to apply the filter to the image. This operation allows the extraction of useful features from the image. According to Goodfellow et al., the convolution of an image I with a filter K is expressed as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n) \tag{3.3}$$

In this formula, $S(i,j)$ represents the output of the convolution at position (i,j). The filter K is applied over the image I such that for each position (i,j), you calculate the sum of the element-wise product of the filter K and the part of the image it overlaps with. This operation is performed for all positions (i,j) in the image, effectively sliding the filter over the entire image and computing a weighted sum of the filter's values and the underlying image values. The result, S is a new matrix representing the filtered image, where specific features have been highlighted or extracted, depending on the nature of the filter K.

Let's apply this operation to the green channel of the image representation in Figure 3.5. Note that in the actual implementation, the pixel values are normalized between 0 and 1 to have a consistent input range, speed up learning, and ensure compatibility with activation functions.

The filter is a fundamental component that enables the network to automatically and adaptively learn spatial hierarchies of features from the input image. As shown in Figure 3.6(a), the filter is convolved across the width and height of the input image, computing the dot product between the entries of the filter and the input to produce a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input. The number of pixels by which the filter is moved across the image when performing the convolution operation is called the stride. A stride of 1 means the filter is moved one pixel at a time, ensuring a



(a) Applying a 3x3 filter to the image

(b) Shifting the filter to the next segment

Figure 3.6: 2D convolution with a stride of 1

detailed and fine-grained analysis of the input image. After the feature map is generated, it goes through a pooling layer. Pooling in CNNs is a downsampling technique used to reduce the dimensionality of feature maps [79]. It involves summarizing the presence of features in patches of the feature map. Two common types are max pooling, which takes the maximum value from the patch of the feature map, and average pooling, which computes the average. Pooling helps to make the detection of features invariant to scale and orientation changes and also reduces the computational load for processing data through the network.

Mobilenet utilized depthwise convolution, a type of convolution operation where each filter is applied to only one input channel. It's part of depthwise separable convolutions, which break the convolution into two layers: depthwise and pointwise convolutions. This approach reduces computational complexity and the number of parameters, making the network lighter and faster without compromising performance significantly [79]. The output of the base model, which represent the extracted features is fed to the fully connected layers in the classifier.

3.2.2 Activation and Loss

Activation functions and loss functions are fundamental components in the architecture of neural networks, each serving a distinct but important role in the learning process. The activation functions used in individual neurons (or nodes) of a neural network are used for the purpose of introducing non-linearity and to allow the network to learn complex patterns [78]. The ones used at the output layer are used to control the range of values in the output. On the other hand, loss functions, also known as cost functions, are used to evaluate how well the neural network performs. They guide the training of the network by providing a reference to update weights and biases to improve the performance of the model. We use a sigmoid activation function at the output and binary cross entropy loss.

Binary Cross-Entropy (BCE) is a loss function used mainly for binary classification tasks, where the goal is to categorize data into one of two classes [78]. It quantifies the difference between the predicted probabilities and the actual labels. For a single data point, the BCE is given by:

$$BCE = -(y \log(p) + (1 - y) \log(1 - p)) \tag{3.4}$$

In this equation, y is the actual label, and has one of two values (0 or 1). p is the predicted probability of the data point being in class 1.

The sigmoid activation function is used to squash the output of the neural network to values between 0 and 1, making it particularly useful for models where we need to predict probabilities as outputs. The equation of the sigmoid function is given as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.5}$$

Let's examine how the loss and activation functions work together to achieve our goal. Let's say our output values (extracted features fed to the fully connected layer) are [2.0,-1.0,0.5,-2.5], and the true labels are [1, 0, 1, 0]. After applying the sigmoid function to these values, they are squashed to approximately [0.88, 0.27, 0.62, 0.08]. The sigmoid function has a range between 0 and 1, and the output probabilities reflect the model's prediction about the likelihood of the input belonging to class 1. The binary cross-entropy loss then measures how well these probabilities align with the true labels, providing a value that quantifies the model's performance. A lower BCE value indicates better performance. It quantifies the error between the network's predictions and the actual labels. Let's consider the second data point for example. The true label is 0 while the predicted value is equal to 0.27. Specifically, for this data point, the network's prediction was higher than the actual label, meaning the predicted probability for the positive class was too high. To correct this, the network's weights will be updated based on the calculated loss value with the help of an optimizer. By iteratively updating the weights in this way, the network learns to reduce the loss, improving its predictions to more closely match the true labels. Over time, this process should lead to a decrease in the loss across the entire dataset, indicating improved performance of the model.

3.2.3 Training and Optimization

Thus far, we have explained the forward pass and how the predictions from the forward pass are compared with the true labels, and a loss (or cost) is computed using a loss function. Once the loss is calculated, the network undergoes a backward pass, known as backpropagation. In this step, the gradient of the loss function is calculated with respect to each weight in the network by applying the chain rule [76]. This process effectively

measures the impact of each weight on the loss. The calculated gradients are then used to update the weights of the network. In the simplest form, each weight is updated by subtracting the product of the learning rate and the gradient from it. The learning rate is a hyperparameter that determines the size of the steps taken during the optimization. The goal is to move the weights in the direction that minimally reduces the loss.

The specific manner in which the weights are updated can vary based on the optimizer used. Optimizers refine the gradient descent process by altering the amount and direction of the weight update [78]. Table 3.1 summarizes the common optimizers and their characteristics:

Optimizer	**Pros**	**Cons**
Gradient Descent	Simple and easy to understand	Can be slow; may get stuck in local minima
SGD (Stochastic Gradient Descent)	Faster convergence and works well with large datasets	May oscillate around the minimum
Momentum	Accelerates convergence and reduces oscillation	Additional hyperparameter (momentum) to tune
Adagrad	Adapts learning rate for each parameter	Learning rate continuously decreases; can become too small
RMSprop	Solves Adagrad's diminishing learning rates	Still sensitive to the initial learning rate setting
Adam (Adaptive Moment Estimation)	Combines advantages of Adagrad and RMSprop; good default choice	More complex; more hyperparameters to tune

Table 3.1: Overview of Optimizers and Their Characteristics

The process outlined thus far enable our solver to effectively learn from multi-label data. However, it doesn't inherently solve the forgetting problem, where learning new tasks can lead to loss of information about previous ones. To understand how we achieve this, we would further explore the generator and how the two components work together.

3.3 Overview of the Generator

The generator is the component of the scholar model which creates samples that closely resemble real data (belonging to past tasks). It is trained to maximize the likelihood that the samples it generates will be indistinguishable from a given real data distribution [2]. Practically, this component is implemented as a generative model. Unlike discriminative models that focus on learning the decision boundaries in a given labeled dataset, generative models aim to capture the underlying distribution of the data itself. This understanding allow them to create new data points that appear to be sampled from the original data. Throughout this work, we consider generative models based on neural networks due to their ability to generate complex data samples such as images. We experimented with variational autoencoders [80] and probabilistic diffusion models [81]. Generative adversarial networks [82] are also capable of generating realistic data samples and have been used by Shin et al. [2]. However, they are harder to train and do not always converge. Let's explore these models and examine how they create data points which are combined with actual data to help train the solver.

3.3.1 Variational Auto Encoders (VAEs)

Variational autoencoders were first introduced by Kingma and Welling as an example of efficient approximate inference with a neural network based recognition model. It combines the principles of autoencoders with probabilistic modeling, where an encoder network learns to map input data to a probability distribution in a latent space, and a decoder network reconstructs the input data from samples of this distribution [80]. This structure enables the VAE to not only compress data efficiently like a standard autoencoder but also to generate new, similar data by sampling from the learned distribution.

Algorithm 3.1 Multi-label Deep Generative Replay (MLDGR) with VAE

Input: Sequence of tasks $T_1, T_2, \ldots, T_N$ with datasets $D_1, D_2, \ldots, D_N$
Output: Trained model
Initialize model $\mathcal{M}$ for the first task T_1
for each task T_i in $T_1, T_2, \ldots, T_N$ **do**
    Train $\mathcal{M}$ on D_i using gradient descent and backpropagation
    if $T_i == T_N$ **then**
        End learning process
    else
        Initialize generator $\mathcal{G}$
        for each epoch in number of epochs **do**
            for each minibatch b in D_i **do**
                Forward pass b through $\mathcal{G}$'s encoder to compress data to latent space z
                Decode latent space z using $\mathcal{G}$'s decoder to reconstruct $\tilde{b}$
                Measure MSE + KL divergence between b and $\tilde{b}$
                Backpropagate the total loss
            end for
        end for
        Initialize new model $\mathcal{M}_{\text{new}}$ for task T_{i+1}
        for each minibatch b in D_{i+1} **do**
            Sample noise vector z from Gaussian distribution
            Decode z using $\mathcal{G}$ to generate synthetic data $\tilde{b}$
            Use $\mathcal{M}$ to assign multi-labels $y_1, y_2, \ldots, y_n$ to $\tilde{b}$, each label $y_j \in \{0, 1\}$
            Augment b with $(\tilde{b}, \text{multi-labels})$ to create augmented batch $\hat{b}$
            Train $\mathcal{M}_{\text{new}}$ on augmented batch $\hat{b}$
        end for
        $\mathcal{M} \leftarrow \mathcal{M}_{\text{new}}$
        Train generator $\mathcal{G}$ on D_{i+1}
    end if
end for

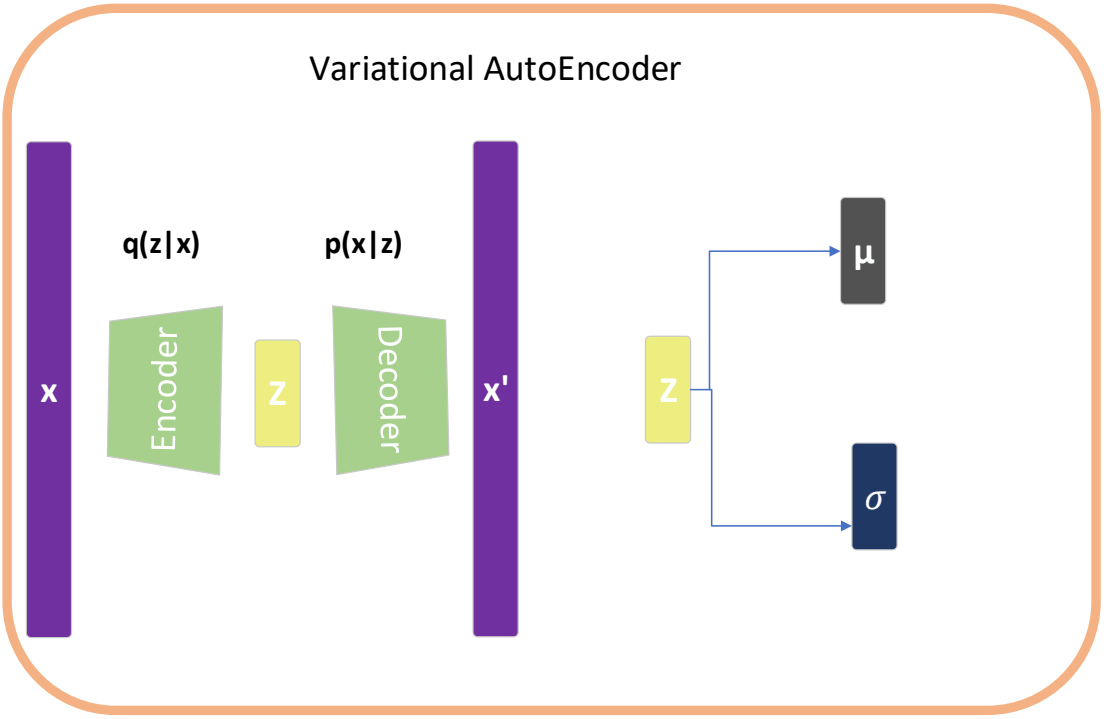


Figure 3.7: Architecture of the variational autoencoder

In Figure 3.7, the VAE is depicted as consisting of three primary components, apart from the input (x) and the reconstructed output (x'): the Encoder, Latent Space (Z), and Decoder. Imagine we have a dataset $X = \{x^{(i)}\}_{i=1}^{N}$ with N samples of a variable x, which could be anything like images or sensor readings. These samples are independent and identically distributed (i.i.d), meaning each sample is drawn randomly and independently from the same underlying process. There are two fundamental assumptions that underpin the operation of the VAE. As formulated by Kingma and Welling, first, there is a hidden value $z^{(i)}$ chosen from a prior distribution $p_{\theta^*}(z)$. z is an unseen variable that influences the creation of this data but is not directly observed. Second, based on this hidden value, the actual data point $x^{(i)}$ is generated from another distribution $p_{\theta^*}(x|z)$ [80]

z is split into μ (mean) and (standard deviation) to describe a distribution for generating latent variables. The reparametrization trick involves creating z by sampling from a standard normal distribution and then scaling and shifting it using μ and δ. This allows the model to backpropagate gradients from the decoder to the encoder, facilitating the optimization of the parameters of the distribution in a way that's compatible with stochastic gradient descent. Without this, backpropagating through the random nodes is not feasible because it involves taking gradients through stochastic nodes, leading to high variance and unstable gradients. The reparametrization trick solves this by expressing each random variable in the latent space as a deterministic function of the model's parameters [76].

The Encoder

The dimensionality of the components used in our implementation is shown in Table 3.2. The encoder is built using a series of convolutional layers. Each layer is encapsulated in a sequential block, ensuring that operations within a block are executed sequentially. A layer in the encoder comprises 2D convolutional structures for spatial downsampling, and units for batch normalization. The convolutional layer reduces the spatial dimensions of the input, effectively compressing the data while increasing the depth as per the specified number of output channels. Batch normalization stabilizes the learning process by normalizing the input of each layer to have a mean of zero and a variance of one, which accelerates training and mitigates the issue of internal covariate shift. We use a LeakyReLU activation function to introduce non-linearity into the encoder, allowing it to learn more complex patterns. Unlike the standard ReLU, LeakyReLU allows a small, positive gradient when the unit is not active, preventing neurons from completely going off during training.

Once the input data is encoded into a lower-dimensional space, the latent space representation is obtained using two separate linear layers. These layers are responsible for producing the mean and log variance of the latent space distribution, necessary for the

reparameterization trick used in VAEs.

The Decoder

The decoder part of the VAE aims to reconstruct the input image from the latent representation. It starts with a linear layer which projects the latent representation back into the feature space. The subsequent layers gradually upscale the feature representation while reducing the depth, essentially reversing the operation performed by the encoder. 2D transposed convolutions are performed to increase the spatial dimensions of the input feature maps. The use of batch normalization layers and LeakyReLU in the decoder mirrors their function in the encoder, ensuring feature normalization and introducing non-linearity, respectively.

Table 3.2: Sizes and Dimensionality in VAE Architecture

Component	**Function**	**Size**
Input Channels	Initial Data Input	3 (RGB images)
Latent Dimension	Size of Latent Space	[128,], [256,]
Hidden Dimensions	Layers in Encoder/Decoder	[32, 64, 128, 256, 512]
Encoder Convolutional Layers	Data Transformation in Encoder	Input: Input Channels, Output: follows Hidden Dimensions list
Encoder BatchNorm Layers	Normalization in Encoder	Matches size of respective Convolutional Layer's output
Encoder Activation Functions	Non-linearity in Encoder	Matches size of respective Convolutional Layer's output

Latent Space Mean	Mean of Latent Distribution	Output: Latent Dimension
Latent Space Log Variance	Variance of Latent Distribution	Output: Latent Dimension
Decoder Input Layer	Mapping Latent to Feature Space	Input: Latent Dimension
Decoder ConvTranspose Layers	Data Transformation in Decoder	Input: follows reversed Hidden Dimensions list, Output: decreases per layer
Decoder BatchNorm Layers	Normalization in Decoder	Matches size of respective ConvTranspose Layer's output
Decoder Activation Functions	Non-linearity in Decoder	Matches size of respective ConvTranspose Layer's output
Final Layer in Decoder	Output Generation	Output: Input Channels

Training the VAE

To train the VAE, there has to be a mechanism to quantify the discrepancy between the generated images and the actual images. The loss function consists of two components. The initial component is the standard reconstruction loss, which encourages the VAE to accurately recreate its input data [78]. The second component is the latent loss, which constraints the network to generate codings that resemble samples drawn from a simple Gaussian distribution. To achieve this, we utilize the Kullback Leibler (KL) divergence to

measure the discrepancy between the desired distribution (the Gaussian distribution) and the actual distribution of the codings [78].

The reconstruction loss is measured using the Mean Squared Error (MSE). MSE measures the average of the squares of the differences between the original and reconstructed data [78]. Mathematically, the reconstruction loss can be expressed as:

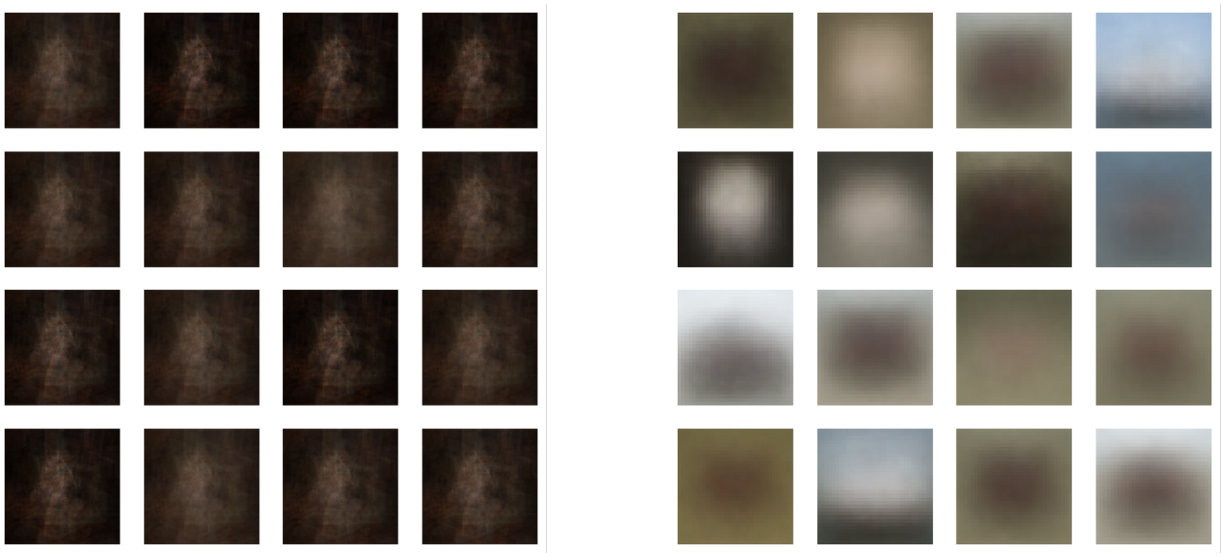
$$L_{\text{reconstruction}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{3.6}$$

where x is the original input, $\hat{x}$ is the reconstructed input, and n is the number of data points. The latent loss, as mentioned earlier is computed using the KL divergence, a measure of how one probability distribution diverges from a second expected probability distribution [76]. In this context, the KL divergence measures how much the learned distribution (of the latent variables) deviates from the desired distribution (Gaussian distribution). Mathematically, given two distributions P and Q, the KL divergence of Q from P is calculated as:

$$D_{KL}(P\|Q) = \sum_{x\in\mathcal{X}} p(x)\log\left(\frac{p(x)}{q(x)}\right) \tag{3.7}$$

$p(x)$ is the probability mass function (PMF) of distribution P at point x and $q(x)$ is the PMF of distribution Q at point x. This regularization ensures the latent space is well-organized and continuous, meaning that similar data points result in similar latent codes, and any point sampled from the latent space can be decoded into a realistic output. Without the KL divergence, the VAE might learn a disjoint, unstructured latent space, hindering its ability to generate new data points effectively.

VAEs are powerful for modeling complex data distributions, but they are not without



(a) Generated samples - 1D latent space

(b) Generated samples - 2D latent space

Figure 3.8: Generated samples with VAE on COCO dataset - Model fails to capture image distribution

challenges. One prominent issue is the generation of blurry images. This often happens because VAEs tend to average features during the reconstruction phase, leading to a loss of sharp, distinctive details in the generated images. This is not a problem in our implementation as long as the solver is capable of assigning labels. However as can be seen in Figures 3.8(a) and 3.8(b), this is not the case. Visualizing the generated samples, we noticed it is impossible for the solver to identify objects in images as they were not properly reconstructed. The complexity and variability of images in our dataset makes it difficult for the VAE to capture and accurately reproduce details in images. This is a much more simpler task when using datasets with a lower complexity and more uniform structure like MNIST.

Another challenge in working with VAEs is determining the appropriate size for the latent space. The latent space is a compressed representation of the input data, and

its dimensionality is a critical parameter. If the latent space is too small, it may not have enough capacity to capture the essential features of the data, leading to a poor reconstruction of inputs. On the other hand, if it's too large, the model may become prone to overfitting, where it learns the noise in the training data rather than the underlying distribution. Finding the right balance for the latent space size is a delicate task and often requires careful tuning and experimentation.

3.3.2 Generative Adversarial Networks (GANs)

GANs stand out as some of the most effective generative models available, particularly noted for their ability to create high-quality, realistic images. They have found successful application across numerous tasks, primarily within research environments. However, GANs offer distinct challenges due to their foundation in game theory, which sets them apart from other generative modeling techniques that typically revolve around optimization [82].

Game theory is a framework for analyzing situations where the outcome for each participant or player in a game depends not just on their own decisions but also on the decisions made by others [83]. In a two-player scenario, each player strategizes to maximize their own benefit, considering the potential choices of the other. The best outcome, often termed as the Nash Equilibrium, is where neither player can improve their situation by unilaterally changing their strategy, while the worst outcome is typically a situation where players fail to cooperate, leading to suboptimal results for all involved.

Taking inspiration from game theory, the generator network known as the actor creates samples while its counterpart, the discriminator network, tries to differentiate between samples obtained from the training data and those produced by the generator. The discrim-

inator provides a probability score which reflects the probability of the generated sample being an actual example from the training set as opposed to a fake sample [76].

Training difficulty with GANs is a well-known issue, primarily due to their architecture. As stated earlier, GANs consist of two competing networks: a generator and a discriminator. This adversarial setup can lead to training instability; the networks might oscillate without finding a stable solution, or the generator could start producing only a subset of possible outputs, a problem known as mode collapse. Furthermore, GANs require careful tuning of the learning rate and other hyperparameters. If the balance between the generator and discriminator is off, the training might not converge, leading to either the generator or discriminator overpowering the other. Lastly, Shin et al. used GANs for their experiments, as such, we opted for VAEs as this could provide a fresh perspective and new insights.

3.3.3 Diffusion Probabilistic Models

Diffusion is an approach to generative modeling that gradually break down the structure in a data distribution via an iterative forward diffusion process [81]. Learning then involves a reverse diffusion process that reconstructs the data's structure, resulting in a highly adaptable generative model of the data. This method enables us to swiftly learn from, generate samples of, and calculate probabilities in deep generative models. It also allows for the computation of conditional and posterior probabilities within the scope of the trained model [81]

In the iterative forward pass, we start with an image, and then add a tiny bit of noise to it. Then the process is repeated many times, each time adding a bit more noise [84]. Mathematically, this can be represented by:

Algorithm 3.2 Multi-label Deep Generative Replay (MLDGR) with Diffusion and BLIP

Input: Sequence of tasks $T_1, T_2, \ldots, T_N$ with datasets $D_1, D_2, \ldots, D_N$
Output: Trained model
Initialize model $\mathcal{M}$ for the first task T_1
Initialize buffer $\mathcal{B}$ for storing image captions
for each task T_i in $T_1, T_2, \ldots, T_N$ **do**
    Train $\mathcal{M}$ on D_i using gradient descent and backpropagation
    if $T_i == T_N$ **then**
        End learning process
    else
        for each epoch in number of epochs **do**
            for each minibatch b in D_i **do**
                Randomly select image x from b
                Generate caption c for x using BLIP model
                Store (x, c) in buffer $\mathcal{B}$
            end for
        end for
        Initialize new model $\mathcal{M}_{\text{new}}$ for task T_{i+1}
        Initialize generator $\mathcal{G}$
        Load content from buffer $\mathcal{B}$
        for each minibatch b in D_{i+1} **do**
            Use $\mathcal{G}$ and text from $\mathcal{B}$ to generate samples $\tilde{x}$ representing previous tasks
            Augment b with $\tilde{x}$ to create augmented batch $\hat{b}$
            Train $\mathcal{M}_{\text{new}}$ on augmented batch $\hat{b}$
        end for
        $\mathcal{M} \leftarrow \mathcal{M}_{\text{new}}$
    end if
end for

$$x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t}\epsilon \tag{3.8}$$

where x_t is the data at step t (the image in this case), β_t is a small noise scale at step t, and ϵ is random noise from a normal distribution. After many steps, the data turns into pure noise. The model then learns to reverse the noising process to reconstruct the original data. This is done by training a neural network to predict the noise that was added at each step during the forward process [84]. Then, this predicted noise is subtracted from the noisy data. The reverse process can also be mathematically represented as:

$$x_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\beta_t}} \epsilon_\theta(x_t, t) \right) \tag{3.9}$$

Here, x_{t-1} is the reconstructed data at step $t-1$ and $\epsilon_\theta(x_t, t)$ is the noise predicted by the neural network at step t. During training, the model is shown pairs of noisy data and the corresponding noise. Its job is to get good at predicting the noise that was added to the data. This is typically done using a loss function that measures the difference between the predicted noise and the actual noise. These equations briefly represent the core of how diffusion probabilistic models operate, transitioning data from a noisy state back to its original form through learned denoising steps.

In our implementation, we leverage the open-source Stable Diffusion model proposed by Rombach et al., a sophisticated framework that operates by gradually transforming random noise into detailed images through a controlled denoising process [85]. This model starts with a canvas of pure noise and, step by step, refines it into a coherent image, guided by the input text. This technique ensures that each phase of noise reduction is not just a move towards clarity but a step towards realizing the specific image dictated by the text prompt. MLDGR stores knowledge of past tasks in the form of text, which is then used

to guide the generation of samples to be interleaved with data from the current task. To accurately identify labels in images with minimal error, we leverage a pre-trained model, BLIP (Bootstrapping Language-Image Pre-training) [86], which excels in understanding and generating content. Figure 3.9 shows samples from our dataset and their associated generated captions. It is important to note that the main labels in this image (person and dog) have been captured by the caption. This is not always the case for all samples. In Figure 3.10 we show reconstructed samples from the image caption. Although the first caption has identified an 'owner', the generated image does not include a person. This means some labels could get missing during the reconstruction stage. However, given the number of samples generated, this is less likely to happen, although it is worth looking into. Employing this methodology, we achieved state-of-the-art results in learning from sequential tasks with minimal catastrophic forgetting.

Figure 3.9: Sample captions generated from training dataset - BLIP base

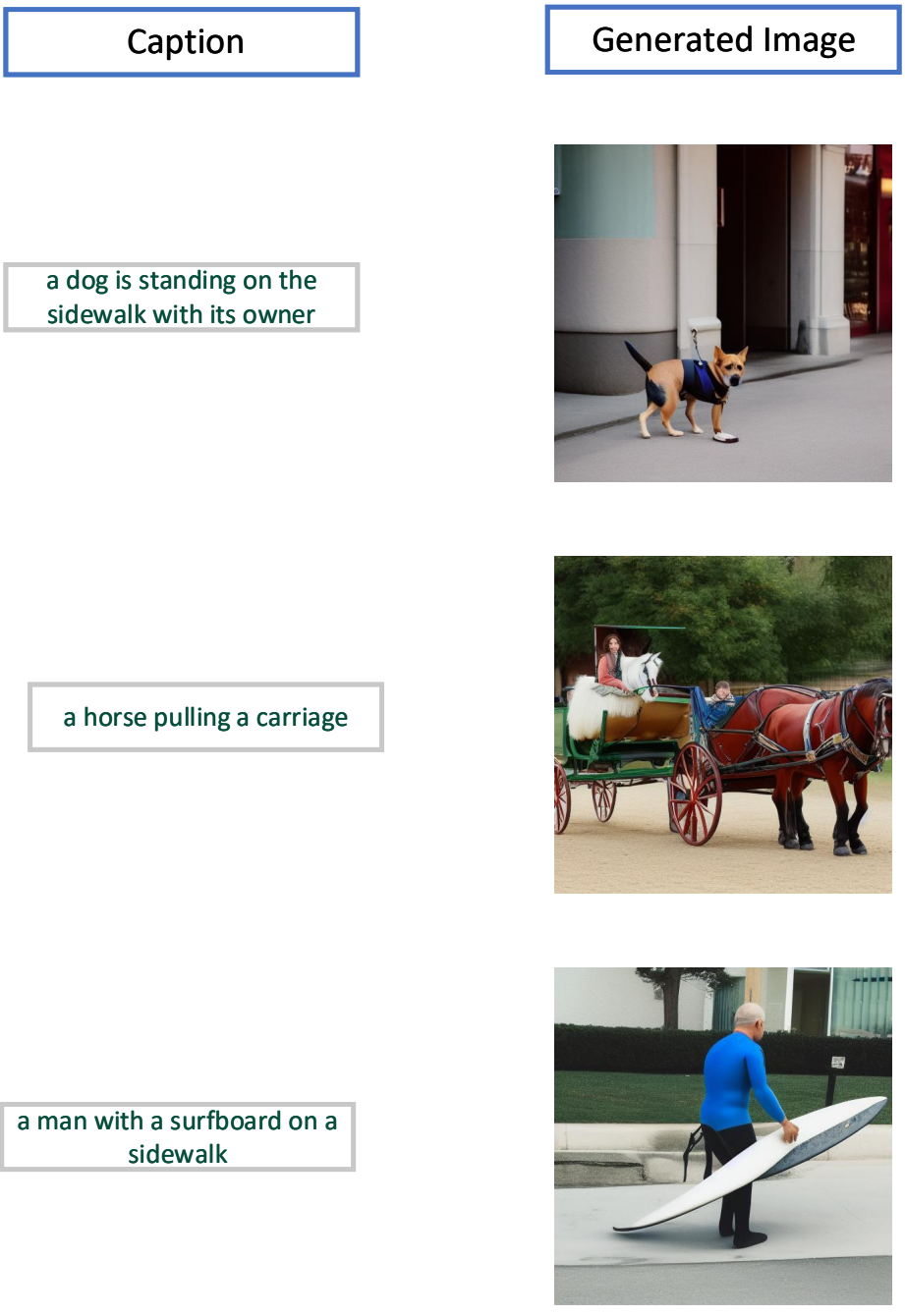


Figure 3.10: Sample images generated from stored captions - Stable Diffusion base

3.4 Summary

In this chapter, we introduced MLDGR, an extension of deep generative replay tailored for multi-label data. The chapter began by detailing the solver's mechanism, highlighting how it effectively learns from multi-label image datasets. Next, we delved into various generator architectures, discussing their respective roles in MLDGR. We explored how these generators are pivotal in generating synthetic samples from previous tasks, by generating textual descriptions from images and then generating new images based on these textual descriptions. This process aids the solver to retain and reinforce knowledge from past learning experiences (tasks) in the lifelong learning pipeline. This approach is instrumental in addressing the challenge of catastrophic forgetting, ensuring that the learning of new tasks does not overshadow the knowledge acquired from older ones, through the generation of synthetic examples. The next chapter discusses our experimental design and evaluation.

Chapter 4

Experimental Evaluation

This chapter outlines the framework utilized to evaluate the MLDGR approach. It details the hardware and software specifications, ensuring a transparent and replicable environment for the conducted experiments. The datasets employed in the experiments are described, providing insights into their sources, characteristics, and the rationale behind their selection. Furthermore, this chapter also highlights the performance metrics adopted to assess the outcomes, offering a clear understanding of the criteria used to measure success. The results are presented and analyzed. Lastly, the chapter discusses the statistical tests employed to validate the findings, ensuring the rigor and reliability of the conclusions drawn from the experimental results.

4.1 Hardware and Software Specifications

In this part of the thesis, we will describe the hardware and software that are essential for our research. We have chosen specific libraries that make it fast and easy to develop

and test our algorithms as well as ensure reproducibility. All of our experiments are conducted using Compute Canada's resources. Compute Canada provides a national-level high-performance computing (HPC) infrastructure. This system offers enhanced computing power and storage capabilities, which allows for more efficient processing of large-scale data analyses compared to typical desktop computers. The hardware resources we requested for our experimental work are detailed as follows.

1. **GPUs** We requested a node equipped with four NVIDIA Tesla V100 GPUs, the node delivers exceptional acceleration capabilities. These GPUs are at the forefront of deep learning hardware, providing the necessary computational power to process large datasets and complex neural network architectures efficiently.

2. **CPUs** Each job request is allocated four CPUs. This multi-core CPU allocation is useful in handling the parallel processing requirements of our experimental tasks that run on the CPU. Example of such tasks include data preprocessing and loading.

3. **Memory** A total of 64 GB of RAM ensures that memory-intensive operations, such as batch processing of large datasets and in-memory data manipulation, are executed seamlessly. This memory allocation is critical for reducing computational bottlenecks that could otherwise impede the training and inference stages of deep learning models.

When submitting a job, the programmer has the flexibility to request the specific resources and values that match their computational needs. Depending on the complexity of the task, you can scale these resource requirements up or down. However, it's worth noting that asking for more resources may increase the wait time before the job begins.

Complementing our hardware, we utilized important software and libraries shown in Table 4.1. The table presents a curated list of eight essential libraries utilized for the com-

putational and analytical tasks within this thesis, highlighting their respective versions and core functionalities. It includes the numerical library NumPy, along with PyTorch and its affiliated libraries—torchvision, torchmetrics, and tensorboard—which provide a comprehensive ecosystem for developing and visualizing deep learning models. For data manipulation, the widely-used library pandas is used. Matplotlib is specified for its robust plotting capabilities, enabling the creation of a variety of figures for data representation. Additionally, scikit-learn is featured for its extensive suite of tools for machine learning tasks. Together, these libraries form the backbone of our implementation and analysis, facilitating a range of operations from data preprocessing to model evaluation and visualization.

Table 4.1: Library Specifications and Descriptions

Library	**Version**	**Description**
Avalanche [87]	0.3.1	A library for continual learning research, built on top of PyTorch.
torch [88]	1.13.1	The core PyTorch library, providing tensor computation with GPU acceleration.
torchvision [89]	0.14.1	A PyTorch package that provides access to popular datasets and models for computer vision.
torchmetrics [90]	0.11.4	A PyTorch-based library providing metrics for model evaluation.
numpy [91]	1.24.2	A fundamental package for scientific computing with Python.
pandas [92]	1.5.3	A Package for data wrangling and manipulation
matplotlib [93]	3.7.0	A Python 2D plotting library which produces publication-quality figures.
scikit-learn [94]	1.2.1	A machine learning library in Python.

We adhered to best practices in computational studies, particularly the importance of creating a reproducible and controlled computing environment. By establishing a dedi-

cated virtual environment, we ensured that each library and dependency, such as PyTorch, Avalanche, and their associated packages, were locked at specific versions. This approach guards against potential discrepancies arising from updates or in the software dependencies [95].

4.2 Datasets Utilized in Experiments

In our experiments, we employed two prominent datasets, namely MSCOCO [96] and NUS-WIDE [97], as introduced in chapter 2. MSCOCO, an acronym for Microsoft Common Objects in Context, is a large-scale dataset widely used for object detection, segmentation, and captioning tasks. It consists of images featuring diverse object categories, making it a standard benchmark in computer vision research. NUS-WIDE, on the other hand, is a dataset created by the National University of Singapore, designed for web image research and encompasses a broad spectrum of images tagged with multiple labels, making it suitable for tasks like image retrieval and multi-label classification. To align with our experimental framework, we utilized sequentialized versions of these datasets, as presented by Du et al. [65].

4.2.1 COCO - Common Objects in Context

Table 4.2 presents detailed information about the MSCOCO dataset. As can be seen in the table, the dataset consists of a total of 328,000 images. Of these, 205,000 are annotated with rich details, including object instances and captions, while the remaining 123,000 are unannotated. In total, the dataset encompasses 1.5 million object instances across 80 distinct common object categories such as car, bus, cat, dog, and surfboard. Each

annotated image is accompanied by 5 descriptive captions, providing a diverse range of context and descriptions.

Table 4.2: Overview of the MSCOCO Dataset

Attribute	**Detail**
Total Dataset Size	328,000 images
Number of Annotated Images	205,000 images
Number of Unannotated Images	123,000 images
Total Number of Object Instances	1.5 million instances
Number of Object Categories	80 categories
Captions per Image	5 captions

Du et al. utilized 40 most common categories from the 80 classes in MS-COCO to create Split-COCO [65]. This subset consists of 65,082 training examples and 27,173 validation examples. These 40 categories are divided into 10 separate tasks, with each task comprising 4 distinct labels. We utilized 5 tasks the first five tasks for our experiments, and the category of objects that constitute each task is shown in Table 4.3

Table 4.3: Task Groupings for MSCOCO

Task	**Labels**
1	Person, Bicycle, Car, Motorcycle
2	Airplane, Bus, Train, Truck
3	Light, Bench, Bird, Cat
4	Dog, Backpack, Umbrella, Handbag
5	Tie, Ball, Skateboard, Surfboard

4.2.2 NUS-WIDE - Web Image Database

Table 4.4 provides an overview of the NUS-WIDE dataset. It comprises 269,648 images, each annotated to reflect one or more of 81 distinct concepts, including concepts such as

animal, flowers, plants, mountain, and road. The dataset's extensive size makes it widely used for tasks such as object recognition, tagging, and semantic categorization.

Table 4.4: Overview of the NUS-WIDE Dataset

Attribute	**Detail**
Annotation Type	Manually Annotated
Total Number of Images	269,648 images
Number of Concepts	81 concepts

The sequentialized version uses 21 most prevalent concepts out of the 81 categories in NUS-WIDE [65]. This subset includes 144,858 training examples and 41,146 validation examples. These 21 concepts are further divided into 7 tasks, each containing 3 unique labels. The first five tasks used in our experiments are shown in Table 4.5

Table 4.5: Task Groupings for NUS-WIDE

Task	**Labels**
1	Animal, Beach, Buildings
2	Clouds, Flowers, Grass
3	Lake, Mountain, Ocean
4	Person, Plants, Reflection
5	Road, Rocks, Sky

We focused on two main preprocessing steps: resizing images and normalizing pixel values. We resized all images in our dataset to a uniform dimension of 256x256 pixels. This standardization is important for batch processing and to ensure that the CNN receives input of a consistent size. Unlike cropping which involves removing portions of an image, potentially losing important data, resizing maintains the entire image content and merely adjusts its scale. This is particularly beneficial in multi-label classification tasks where multiple objects may be present in a single image, and losing any part of the image could mean missing out on critical information.

After resizing, we normalized the pixel values of each image to a range between 0 and 1. This normalization step is a common practice in image processing, as it scales the pixel intensity values to a standard range, facilitating model convergence during training. By converting pixel values from the usual range of 0 to 255 to a normalized range, the model can train more efficiently. This is because normalized data has a consistent scale, leading to a smoother and more stable gradient descent process.

4.3 Experimental Setup and Results

The experiments were structured to ensure reliability in the results. We conducted a total of five runs, each initialized with a distinct seed to mitigate the influence of random initialization on the outcomes. In each run, we recorded several key performance metrics: the mean Average Precision (mAP), overall F1-score, class-wise F1-score (CF1), and the measure of forgetting. The F-score is particularly important due to the inherent imbalance present in multi-label datasets. These metrics collectively provide a comprehensive insight into the model's performance, capturing aspects of precision, recall, and the model's ability to retain knowledge over time.

We compared MLDGR with a Naive Baseline and three state-of-the-art models. The Naive Baseline serves as a benchmark for evaluating the effectiveness of other algorithms in mitigating forgetting. It utilizes the same underlying model architecture as the other algorithms but lacks any specific mechanism to alleviate forgetting. The parameters used by the underlying model is summarized in Table 4.6 The model is trained sequentially on tasks as they arrive, without any explicit strategy to retain previously learned knowledge. As a result, the Naive Baseline's performance on earlier tasks deteriorates as it learns new tasks.

In Chapter 2, we presented several lifelong learning techniques, key among them EWC, LwF and SI. We employ these three state-of-the-art algorithms [65] [47] in our comparative evaluation. Recall that, LwF uses knowledge distillation to enforce similarity between the predictions of the network on previous tasks and the current task [50]. The algorithm optimizes a set of shared parameters across all tasks (θ_s) and the parameters specific to the new task (θ_n). Additionally, it imposes a constraint to ensure that the predictions on the samples of the new task using θ_s and the parameters of old tasks (θ_o) remain consistent, thereby preserving the memory of θ_o. In EWC, a quadratic penalty is imposed on the changes in parameters that are important for previous tasks. This penalty slows down the learning of task-relevant weights, helping to preserve knowledge acquired from old tasks [46]. EWC effectively balances the need to learn new tasks while retaining information from past tasks. Lastly, SI, similar to EWC penalizes changes to the most crucial synapses (parameters), allowing the model to learn new tasks with minimal forgetting. However, it does this by estimating the importance of individual synapses for solving a particular learned task [47].

Table 4.6: Experimental Setup and Parameters

Parameter	**Naïve**	**EWC**	**LwF**	**SI**	**MLDGR**
Optimizer	SGD	✓	✓	✓	✓
Learning Rate	0.001	✓	✓	✓	✓
Momentum	0.9	✓	✓	✓	✓
Criterion	BCELoss	✓	✓	✓	✓
Batch Size (Train)	64	✓	✓	✓	✓
Batch Size (Eval)	64	✓	✓	✓	✓
Epochs	5	✓	✓	✓	✓
EWC Lambda	-	0.1	-	-	-
LwF Alpha	-	-	0.1	-	-
LwF Temperature	-	-	0.1	-	-
SI Lambda	-	-	-	0.001	-
SI Epsilon	-	-	-	0.1	-
Number of Runs	5	✓	✓	✓	✓
Random Number Seeds	[15, 25, 40, 60, 85]	✓	✓	✓	✓

The aggregated results of the experiments conducted are presented in Tables 4.7 and 4.8. For a more granular view, including the results from each individual run, refer to the Appendix

Table 4.7: Experimental Results - MSCOCO

Method	**mAP**	**OF1**	**CF1**
Naïve Baseline	62.26	6.83	7.14
EWC	77.54	62.90	61.32
LWF	76.71	60.88	58.74
MLDGR (Ours)	**89.05**	**78.69**	**78.11**
SI	79.34	67.67	66.44

Table 4.8: Experimental Results - NUSWIDE

Algorithm	**mAP**	**OF1**	**CF1**
Naïve Baseline	65.81	5.21	5.08
EWC	79.25	66.23	64.59
LWF	77.26	61.92	59.64
MLDGR (Ours)	**87.93**	**77.19**	**77.13**
SI	79.91	68.44	67.05

Table 4.9: Forgetting Metrics for MSCOCO and NUS-WIDE Datasets

Dataset	**Naïve Baseline**	**EWC**	**LWF**	**MLDGR (Ours)**	**SI**
MSCOCO	0.078525	0.03745	0.039875	0.000575	0.032275
NUS-WIDE	0.1126	0.05725	0.053975	0.02145	0.051525

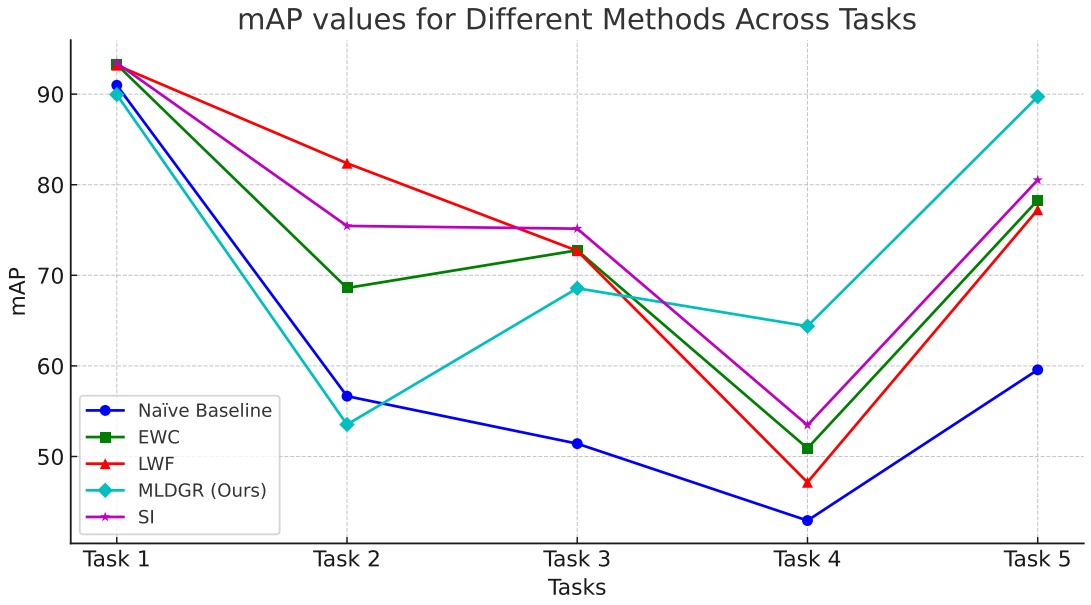


Figure 4.1: mean average precision (mAP) - MSCOCO

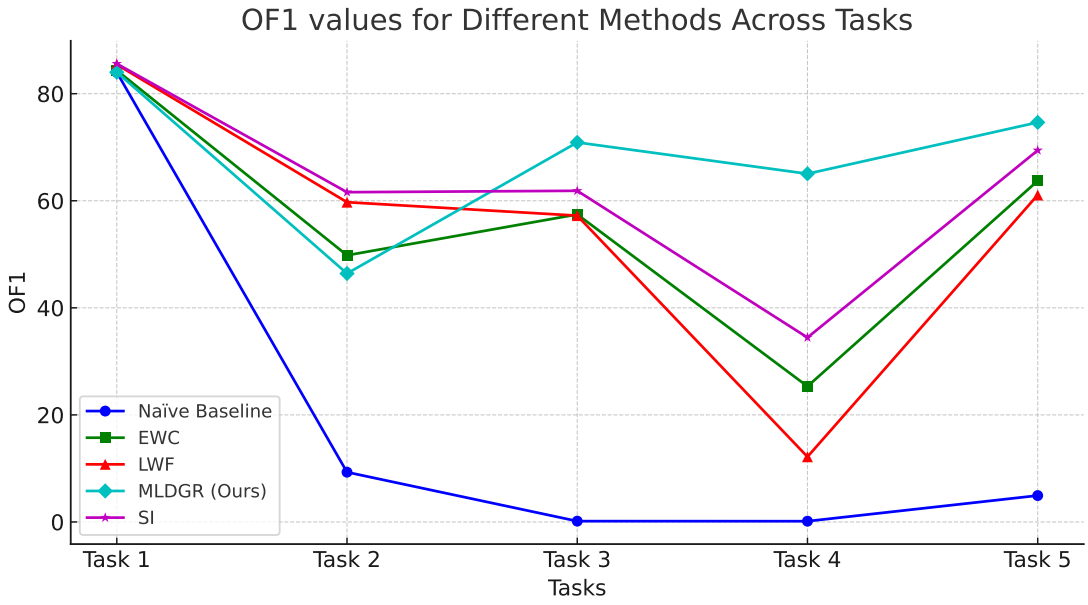


Figure 4.2: Overall F1 score - MSCOCO

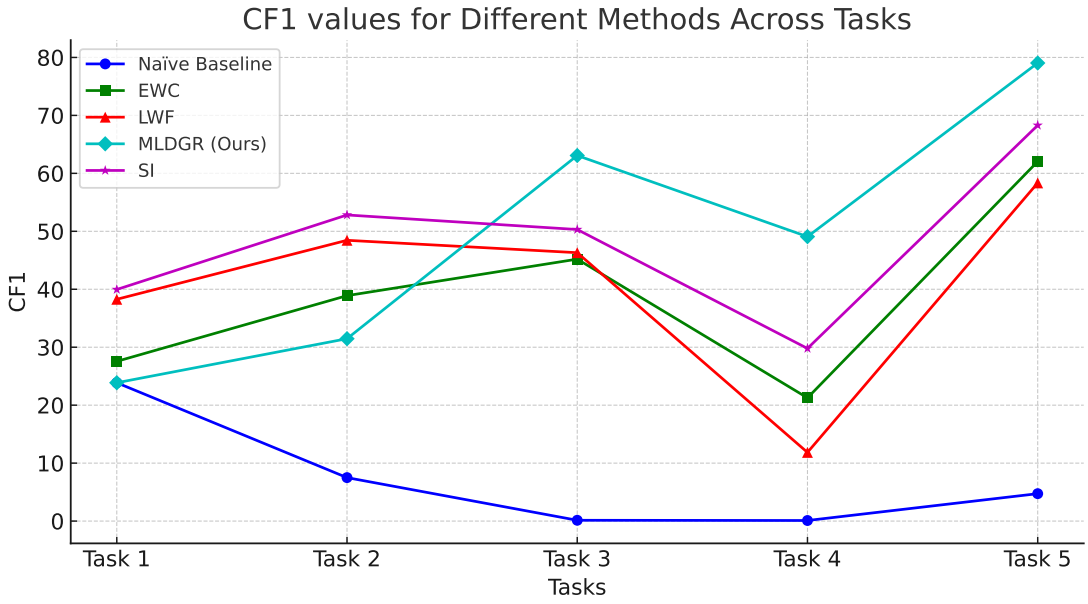


Figure 4.3: Class-wise F1 score - MSCOCO

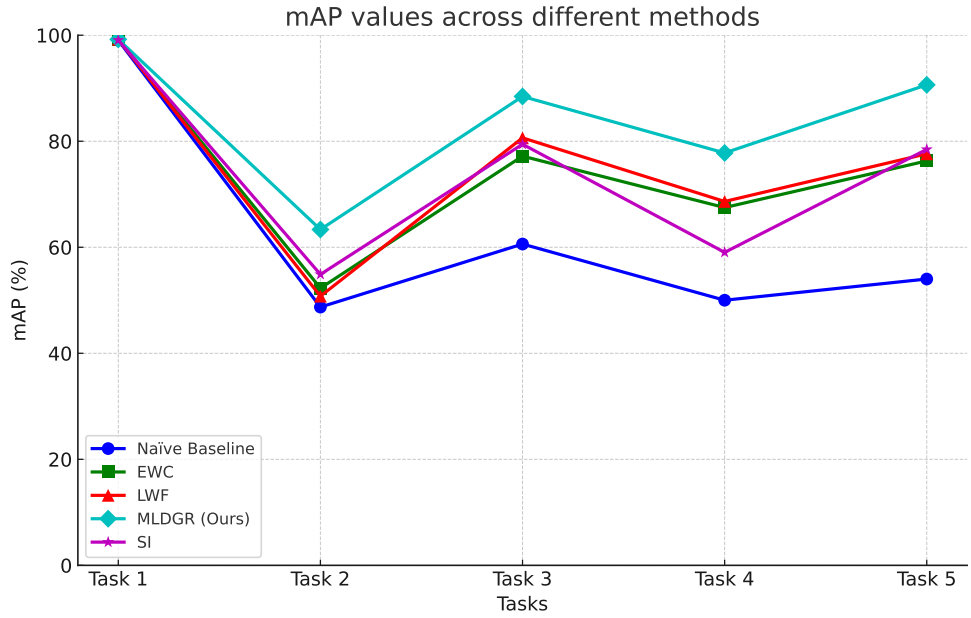


Figure 4.4: mean average precision (mAP) - NUSWIDE

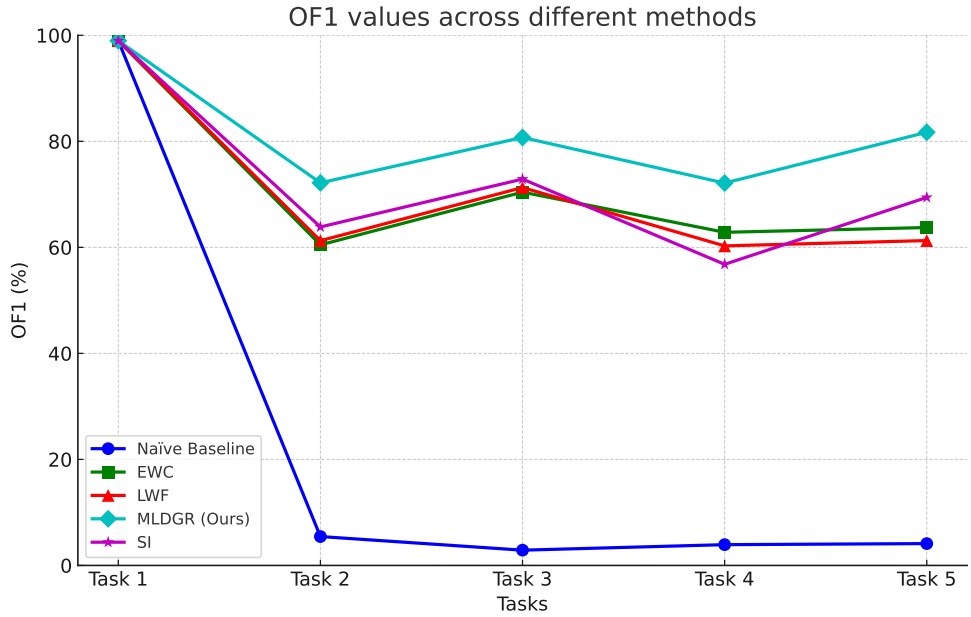


Figure 4.5: Overall F1 score - NUSWIDE

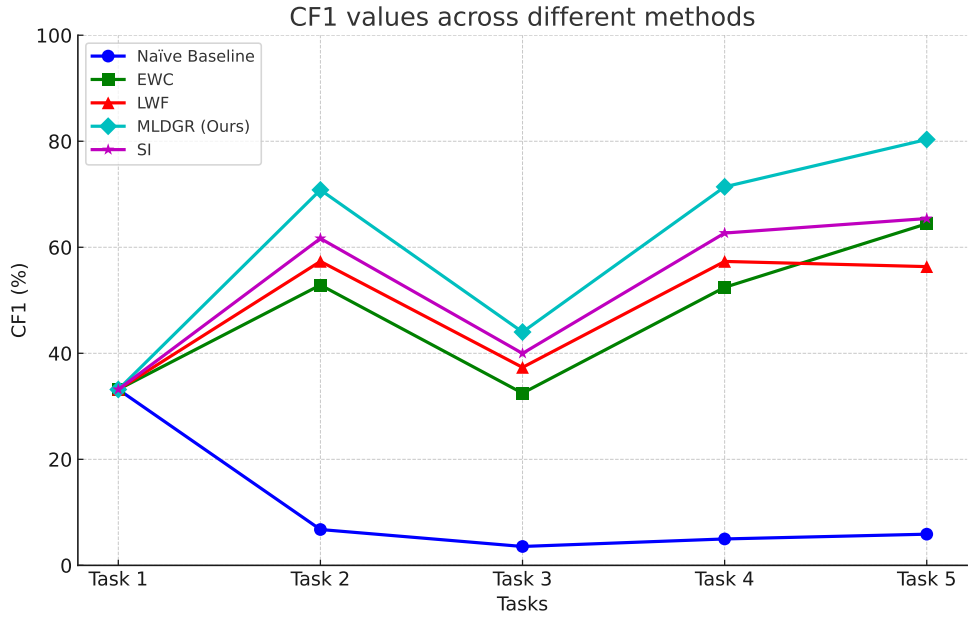


Figure 4.6: Class-wise F1 score - NUSWIDE

The results presented in Tables 4.7, 4.8 and 4.9 are elaborated in the discussion and analysis section. Here, we analyze the performance of the algorithms as new tasks are learnt. Figures 4.1, 4.2, and 4.3 show the mAP, OF1, and CF1 scores respectively as new tasks are learnt on the MSCOCO dataset. Naïve Baseline, as expected, generally has the lowest performance, which is typical as it usually represents a basic comparison point without advanced features. EWC and SI show strong performance across the tasks, with relatively high mAP values in most cases. This indicates robustness and general effectiveness in handling different tasks. LWF also performs well, particularly notable in task 2, where it has the highest mAP (82.37) among all algorithms. However, it has a somewhat lower performance in task 4. MLDGR (Ours) shows a unique pattern with a decline in performance, like all the other algorithms when learning the second task, but excels as more and more tasks are introduced, even achieving the highest mAP, OF1 and

CF1 score by the 5th task. There is a notable increase in performance in the later tasks, which indicate a specific adaptation to cumulative learning scenarios. A similar trend is observed in Figures 4.4, 4.5, and 4.6 as new tasks are learnt on the NUS-WIDE dataset.

4.4 Statistical Significance

To draw reliable conclusions and to ensure the validity of the experimental results, conducting a statistical significance test is a necessary step in machine learning experiments. Statistical tests are used to determine if the difference in performance between classifiers is real or just happened by chance. They do this by looking at how consistently one classifier outperforms another. This can be checked across many different test scenarios or by repeating tests several times on the same set of data. If a classifier consistently does better, it suggests that its superior performance is likely not just a chance [98]

Various statistical tests have been designed to help us understand how algorithms perform. For example, if we have two algorithms and want to determine which one is more effective at a specific task using a single dataset, we might use a paired t-test [98]. This test is based on the idea that by comparing the performance of both algorithms on the same data, we can control for any variations caused by the dataset itself. Also, statistical tests can be used to compare multiple algorithms on a single dataset and multiple algorithms on multiple datasets.

To assess how well multiple algorithms perform on multiple datasets, there are a few established methods to choose from. In this study, we utilize the Friedman test [98] to achieve this purpose. The Friedman test ranks algorithms on each dataset, from best performing to worst performing. It analyzes ranks of each classifier on each dataset and not on the actual values of the performance measure [98]. In our case, there are five

algorithms with only two datasets. However, we have different performance measures to choose from. Let's consider the mean average precision from the results shown earlier. Under the null hypothesis that all algorithms perform equally, the ranks for all algorithms should be the same. Let's proceed to test this hypothesis by assigning ranks as shown in Table 4.10. We then calculate the mean rank for each algorithm. This is done by summing the ranks for each algorithm across all datasets and then dividing by the number of datasets. Mathematically:

$$R_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij} \tag{4.1}$$

where R_j is the mean rank for the $j-th$ algorithm, n is the number of datasets, and r_{ij} is the rank of the $j-th$ algorithm in the $i-th$ dataset

Table 4.10: mAP values with ranks and average ranks for MSCOCO and NUSWIDE datasets

Dataset	**Naïve Baseline**	**EWC**	**LWF**	**MLDGR (Ours)**	**SI**
MSCOCO	62.26 (5)	77.54 (3)	76.71 (4)	89.05 (1)	79.34 (2)
NUSWIDE	65.81 (5)	79.25 (3)	77.26 (4)	87.93 (1)	79.91 (2)
Sum	10.0	6.0	8.0	2.0	4.0
Average Rank	5.0	3.0	4.0	1.0	2.0

Next, we calculate the overall mean rank, which is the average of all the ranks as follows:

$$\bar{R} = \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n} r_{ij} \tag{4.2}$$

Where m is the number of algorithms and all variables have their usual meaning as introduced earlier. This value is 3.0 in our case. The total sum of squares and sum of

squares error is then computed to quantify the variability within the algorithm ranks and error variability respectively. This is determined as:

$$SS_{total} = n \sum_{j=1}^{m} \sum_{i=1}^{n} (r_j - \bar{R})^2 \tag{4.3}$$

$$SS_{Error} = \frac{1}{m(n-1)} \sum_{j=1}^{m} \sum_{i=1}^{n} (r_{ij} - \bar{R})^2 \tag{4.4}$$

The Friedman statistic is expressed as a ratio of SS_{total} and SS_{error}. The distribution of the test statistic is approximated by a chi-squared distribution with $m - 1$ degrees of freedom. This is usually true for a larger dataset ($n > 15$) and $m > 5$. For small values on n, the approximation is imprecise [98] . The statistic is simplified to:

$$Q = \frac{12}{n \times m(m+1)} \sum_{j=1}^{m} (R_j - \bar{R})^2 - 3 \times n \times (m+1) \tag{4.5}$$

For $n = 2$ and $m = 5$, the value of this statistic is 8.0. At a 0.05 significance level, the critical value for $n = 2$ and $m = 5$ is 7.60 (refer to 1). This outcome provides substantial evidence to reject the null hypothesis, which posits that the performance of the algorithms across the two datasets are identical. The surpassing of the critical value by the test statistic indicates that there are statistically significant differences in the performance of at least some of the algorithms when evaluated on the MSCOCO and NUSWIDE datasets. This finding means we can carry out further investigation through post hoc analysis to identify the specific pairs of algorithms that exhibit significant differences in performance.

In our analysis, we've observed inherent limitations associated with the Friedman test, particularly evident due to the small number of datasets in our study (2). The test's

methodology, which centers around the use of ranks, overlooks the actual values of the data, focusing instead on their ordinal relationships. This approach, while mitigating the influence of outliers and non-normal distributions, simultaneously disregards the true magnitudes of differences between observations. Consequently, substantial variations in performance, are condensed into a uniform ranking structure, potentially masking meaningful insights. Furthermore, the Friedman test does not account for the variance within individual groups, an aspect that could offer additional context about the consistency or reliability of the algorithms' performances.

The Nemenyi test [98] compares the average ranks of each pair of groups and determines whether the difference between these ranks is greater than the expected standard error. The formula for the critical difference (CD) for the Nemenyi test is:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{4.6}$$

q_α is the critical value based on the studentized range statistic, obtainable from the studentized range distribution table (see 2) for infinite degrees of freedom at a chosen significance level α. k is the number of algorithms, and N the number of datasets.

Figure 4.7 shows the Nemenyi diagram. Each horizontal line represents one of the algorithms tested (MLDGR, SI, Naive, LWF, EWC). The length of the bracketed line represents the critical difference value. If the average ranks of two algorithms are further apart than this critical distance, their performance difference is considered statistically significant [99]. The Nemenyi test does not successfully indicate where the significant differences lie among the algorithms. This conclusion is based on the observation that, despite the Friedman test indicating a significant overall difference, the Nemenyi test did not identify significant pairwise differences between the algorithms. This outcome can

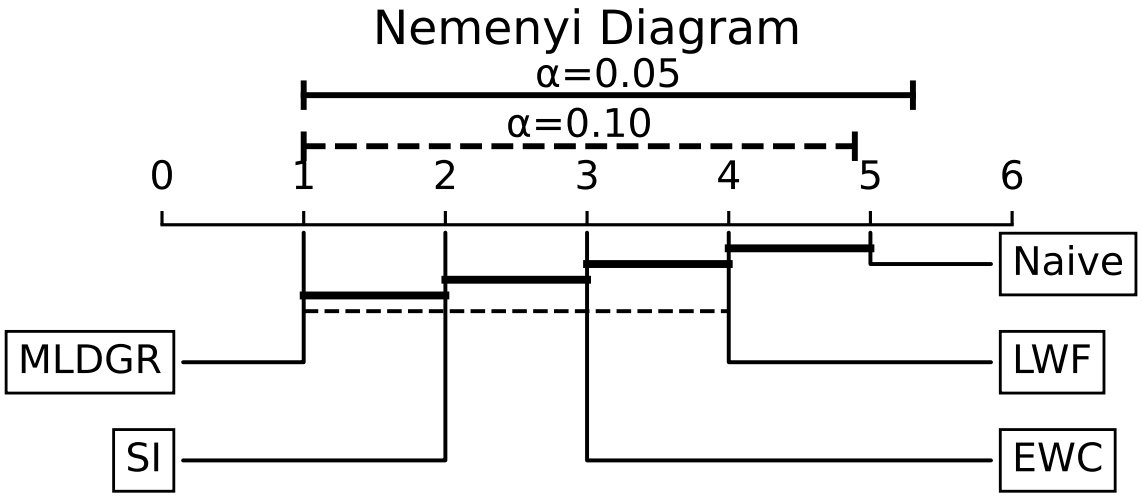


Figure 4.7: Nemenyi post hoc test

occur due to several reasons, especially when dealing with a small number of datasets as in this scenario. With only two datasets, the CD value in the Nemenyi test can be relatively large. This means that the test requires a substantial difference in the average ranks of the algorithms to declare a significant pairwise difference. If the actual performance differences are not larger than this CD, the Nemenyi test may not indicate significant differences, even if the Friedman test suggests overall differences. As is rightly pointed out by Benavoli et al., the outcome of the comparison between any two algorithms is dependent also on the performance of the other $k - 2$ algorithms [100]. This limitation of the post-hoc test is important and should not be ignored when making conclusions [100, 101].

At a 0.1 significance level, there is a statistical significant difference between MLDGR and the Naive Baseline. There is no significant difference between any other pair. This suggests that the difference in performance between the Naive Baseline and all the other algorithms is by chance. However, before coming to such a conclusion, it is important to consider that the critical difference is inversely related to the number of datasets, and a

small dataset size means a large value of CD. Increasing the number of datasets significantly decreases the value of CD. We can infer that a different conclusion would be drawn if similar results is replicated across more datasets,

4.5 Discussion and Analysis of Results

Here, we provide an in-depth evaluation of the experimental outcomes obtained from the MSCOCO and NUSWIDE datasets. This section dissects the implications of the findings, emphasizes the strengths of the methods tested and acknowledges their limitations. First, we start with the results in Table 4.7. It indicates a notable performance differential between the methods. The MLDGR method, our proposed approach, stands out with the highest mAP (mean Average Precision) of 89.05, which suggests a significant ability to correctly label multiple objects within an image. Its OF1 and CF1 scores, which measure the harmonic mean of precision and recall, also top the chart. These metrics collectively suggest that MLDGR not only accurately identifies the relevant objects in the images but also maintains this accuracy consistently across different objects.

The Naive Baseline, in contrast, displays considerably lower performance across all metrics, highlighting the enhanced sophistication and effectiveness of the more advanced methods like EWC, LWF, and SI. EWC and SI, in particular, show strong performance, with SI achieving better CF1 scores, indicating its effectiveness in handling imbalances, which is an inherent property of the dataset. Interestingly, the mAP values appear relatively high, yet the OF1 and CF1 values are significantly lower. This discrepancy can be explained with a practical example. Consider a single test image that is correctly labeled as '1000', meaning the first category is present and the remaining three are absent. If the Naive approach incorrectly predicts '1111' for this image, it erroneously identifies objects

in all categories, including those that are not actually present. the mAP is calculated by averaging precision values at different thresholds of recall. If the algorithm ranks the correct label (first category) highly among its predictions, it can still achieve a high mAP score for this instance, because mAP doesn't penalize for the false positives as harshly as it rewards the correct prediction. In multi-label tasks, if the algorithm consistently ranks the correct labels higher than the incorrect ones, even with many false positives, it can still result in a reasonably high mAP. The F1 score on the other hand is the harmonic mean of precision and recall, which means they balance the two. In our example, the precision would be 25% because only one out of four predicted labels is correct. The recall would be 100% because the algorithm did not miss the correct label. However, due to the high number of false positives, the OF1 and CF1 scores would be low because they are more sensitive to the incorrect labels that were predicted. The F1 score directly penalizes the algorithm for every false positive (incorrectly predicted label) and false negative (missed label), leading to a more accurate reflection of the model's performance.

In terms of forgetting, the forgetting metric values range from -1 to 1, where -1 signifies a positive transfer of knowledge, indicating that previous learning has been beneficial for learning new tasks, 0 indicates no forgetting and 1 represents total forgetting (suggesting that learning new tasks has completely overwritten the knowledge from previous tasks). The Naive Baseline shows the highest forgetting metric values for both datasets, indicating a significant amount of forgetting. This is expected as the Naive Baseline does not include mechanisms to prevent forgetting. Our proposed method, MLDGR, shows the lowest forgetting metrics for both datasets. For MSCOCO, the forgetting metric is almost zero, indicating negligible forgetting. For NUS-WIDE, the forgetting metric is higher than MSCOCO but still significantly lower than the other methods. While EWC, LWF, and SI can reduce forgetting on the multi-label tasks, they are not as effective as MLDGR.

The superior performance of MLDGR points to the potential of specialized architectures in handling complex, multi-label datasets. It is also important to consider the limitations reflected in these results. The disparities in performance across different methods could be influenced by various factors such as hyperparameter settings, the complexity of the models, and the inherent challenges presented by the MSCOCO dataset. Moreover, while mAP, OF1, and CF1 are comprehensive metrics, they do not capture every aspect of model performance, such as inference time and robustness to adversarial attacks, which are important for real-time and secure applications.

4.6 Summary

In this chapter, we outline the framework adopted to evaluate our proposed method and to conduct comparative analyses with other existing methods. We start by detailing the characteristics and relevance of the two key datasets employed in our study: MSCOCO and NUSWIDE. The sequentialized versions of these datasets are instrumental in assessing the performance of our method and are chosen to represent the challenges encountered in multi-label lifelong learning scenarios. Next, we elaborate on the specific criteria used to measure the success of our method. We discuss the mean average precision, overall F1, class-wise F1, as well as performance indicators unique to lifelong learning, specifically, forgetting. We further present and analyze our experimental results and conduct statistical significance tests to ascertain the reliability of our results. We explain the use of the Friedman test and post-hoc Nemenyi tests, which help confirm that the observed differences in performance are indeed significant and not due to random variation. We also dive deeper into interpreting the experimental outcomes by exploring the implications of our findings, discussing strengths and limitations. Lastly, we describe the technical setup that underpins

our experimental work and restate the importance of replicability in our research.

Chapter 5

Conclusion and Future Work

In this concluding chapter, we review the main goals and results of our research. We began this research with the aim of applying lifelong learning algorithms in the multi-label domain and potentially improving their performance. Through detailed experiments and analysis, we have introduced MLDGR, a new method that augments the generative replay approach and thoroughly assessed its performance. In this chapter, we will look back at our initial objectives to see how well they were met and consider the impact of our work on the field. We'll also address the limitations we faced and suggest ways to overcome them in future research. The practical applications of our findings and their significance in real-world settings will be a key part of our final discussion. Finally, we will outline directions for future research, pointing out how the insights from our study might guide further advances in the field.

5.1 Contribution

Our major contribution is the development of Multi-label Deep Generative Replay (MLDGR), a method that enhances Deep Generative Replay by incorporating a memory buffer and a multi-label problem-solving mechanism. In the first step of the MLDGR approach, multiple labels are assigned to images using a convolutional neural network to extract features from the images. These features are then used by a base classifier to perform multi-label classification. In the second step, selected image samples are converted into text descriptions and saved in a buffer. When the model is learning a new task, these text descriptions from previous tasks are used to create synthetic images which are mixed with real images from the current task to help the model retain the knowledge it has learned before. This procedure aids in preventing the model from forgetting previous knowledge. Thus, MLDGR is able to effectively process and learn from multi-label datasets across a series of tasks while maintaining a robust performance on previously learned tasks. Our algorithm leverages the intrinsic capability of CNNs, to identify relations among multiple labels within an image. This is achieved without the need for any supplementary mechanisms or complex components.

Our innovative approach to creating synthetic images in the lifelong learning pipeline involves generating textual descriptions of images and using this information to condition the image generator. This conditioning process ensures that the images generated are representative of the previous tasks, therefore improving performance on past tasks. Generating synthetic data has many other use cases including handling imbalanced datasets.This makes MLDGR a good fit in such scenarios. Let's consider a scenario in the transportation domain, specifically in the context of autonomous vehicle development as mentioned earlier. One of the critical tasks for autonomous vehicles is the detection of various road

signs to navigate safely and obey traffic rules. However, certain road signs are less common than others, leading to an imbalanced dataset where some types of road signs are underrepresented. In such a dataset, stop signs and speed limit signs might be abundant, while school zone signs are rare. This imbalance can cause the trained model to be less accurate in detecting the less common signs. To address this issue, more synthetic images of the underrepresented signs could be generated during learning and augmented with the real dataset to balance the distribution.

By addressing the existing research gap through extensive experiments, we demonstrated that MLDGR can sustain the acquisition of new knowledge across multi-label tasks over time. Our results show that MLDGR mitigates the loss of previously learned information and also maintains high performance on new tasks. The model's ability to generate synthetic data also offers a first step to address concerns regarding data privacy. By using synthetic data for replay, the model can continue learning without the need to store or revisit sensitive real-world data from past tasks.

5.2 Future Work

Future research directions stemming from our work could further refine and expand upon the methodology we've developed. Our approach, which enhances the generative replay method with a memory buffer, was designed to overcome the shortcomings of the traditional generative replay when faced with the challenge of creating representative samples from our dataset. We introduced a technique whereby the algorithm randomly selects an image from each mini-batch of the previous task and converts it into text that captures the labels in the task, for more efficient storage. This random selection process is limitation of our study, and a natural extension of this research would be to develop a more sophisticated

sample selection process. Rather than random selection, future algorithms could develop strategies to intelligently choose samples that are more representative of the full range of labels within a task. This could potentially lead to improvements in the model's ability to recall and utilize past knowledge. To elaborate further, let's consider an example where we have a dataset consisting of images of cats, dogs, and chinchillas. Each image in the dataset is labeled with at least one of these. With random selection of images from each mini-batch, we could end up with a memory buffer that contains descriptions of cats and dogs, but not chinchillas. This could lead to performance degradation as more tasks are introduced to the model. Isele and Cosgun worked on a sample selection method for experience replay [102]. Although their approach is not generative, and the authors focus on selecting samples from the original dataset to be replayed, their work shows that intelligent sample selection can improve performance and prevent catastrophic forgetting.

Furthermore, we focused on image datasets in this study and hence generated textual description of select images for storage. However, this may not be a suitable approach for datasets that consist primarily of text.The pursuit of a more universal compression technique for stored data represents an exciting avenue for further investigation. The goal would be to devise a method that is not only effective for images but can be applied across various data modalities including text and audio. Such a method would have far-reaching implications, enabling a more versatile and efficient use of storage space and processing power. This could significantly enhance the applicability of lifelong learning systems in real-world scenarios where data is not limited to a single modality.

References

[1] Z. Chen and B. Liu, *Lifelong machine learning*, vol. 1. Springer, 2018.

[2] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances In Neural Information Processing Systems*, vol. 30, 2017.

[3] A. L. Samuel, "Some studies in machine learning using the game of checkers," *Ibm Journal Of Research And Development*, vol. 3, no. 3, pp. 210–229, 1959.

[4] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers In Psychology*, vol. 4, p. 504, 2013.

[5] D. C. Mocanu, M. T. Vega, E. Eaton, P. Stone, and A. Liotta, "Online contrastive divergence with generative replay: Experience replay without storing data," *Arxiv Preprint Arxiv:1610.05555*, 2016.

[6] M. A. Kassim, H. Viktor, and W. Michalowski, "Multi-label lifelong machine learning: A scoping review of algorithms, techniques, and applications," *Submitted For Publication*, 2024.

[7] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, pp. 1–27, 2011.

[8] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proceedings Of The 14th Acm International Conference On Information And Knowledge Management*, pp. 195–200, 2005.

[9] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *Ieee Transactions On Knowledge And Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2013.

[10] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proceedings Of The 16th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining*, pp. 999–1008, 2010.

[11] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[12] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *European Conference On Principles Of Data Mining And Knowledge Discovery*, pp. 42–53, Springer, 2001.

[13] F. De Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision trees from texts and data," in *International Workshop On Machine Learning And Data Mining In Pattern Recognition*, pp. 35–49, Springer, 2003.

[14] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," *Advances In Neural Information Processing Systems*, vol. 14, 2001.

[15] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Machine Learning*, vol. 73, pp. 133–153, 2008.

[16] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *Ieee Transactions On Knowledge And Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[17] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multilabel video annotation," in *Proceedings Of The 15th Acm International Conference On Multimedia*, pp. 17–26, 2007.

[18] N. Ueda and K. Saito, "Parametric mixture models for multi-labeled text," *Advances In Neural Information Processing Systems*, vol. 15, 2002.

[19] S. Zhu, X. Ji, W. Xu, and Y. Gong, "Multi-labelled classification using maximum entropy method," in *Proceedings Of The 28th Annual International Acm Sigir Conference On Research And Development In Information Retrieval*, pp. 274–281, 2005.

[20] S. Ji, L. Tang, S. Yu, and J. Ye, "Extracting shared subspace for multi-label classification," in *Proceedings Of The 14th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining*, pp. 381–389, 2008.

[21] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia Conference On Knowledge Discovery And Data Mining*, pp. 22–30, Springer, 2004.

[22] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, pp. 211–225, 2009.

[23] R. Yan, J. Tesic, and J. R. Smith, "Model-shared subspace boosting for multi-label classification," in *Proceedings Of The 13th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining*, pp. 834–843, 2007.

[24] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *2008 Eighth Ieee International Conference On Data Mining*, pp. 995–1000, IEEE, 2008.

[25] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, pp. 333–359, 2011.

[26] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *European Conference On Machine Learning*, pp. 406–417, Springer, 2007.

[27] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal Of Data Warehousing And Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[28] G. Tsoumakas, I. Katakis, and I. Vlahavas, *Mining multi-label data*, pp. 667–685. Springer, 2010.

[29] K. Dembczyński, W. Cheng, and E. Hüllermeier, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings Of The 27th International Conference On Machine Learning (ICML)*, pp. 279–286, 2010.

[30] M. Antenreiter, R. Ortner, and P. Auer, "Combining classifiers for improved multilabel image classification," in *Proceedings Of The 1st Workshop On Learning From Multilabel Data (MLD) Held In Conjunction With ECML/PKDD, Bled, Slovenia*, pp. 16–27, 2009.

[31] J. Read, "A pruned problem transformation method for multi-label classification," in *Proceedings Of The Nz Computer Science Research Student Conference*, pp. 143–150, 2008.

[32] J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan Kaufmann Publishers, Inc., 1993.

[33] K. Cram and Y. Singer, "A family of additive online algorithms for category ranking," *Journal Of Machine Learning Research*, vol. 3, no. Feb, pp. 1025–1058, 2003.

[34] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proceedings Of The Fifteenth International Conference On Machine Learning (ICML '98)*, (San Francisco, CA, USA), pp. 55–63, 1998.

[35] M. Petrovskiy, "Paired comparisons method for solving multi-label learning problem," in *Sixth International Conference On Hybrid Intelligent Systems (HIS '06)*, p. 42, 2006.

[36] S. Wan and J. Xu, "A multi-label classification algorithm based on triple class support vector machine," in *International Conference On Wavelet Analysis And Pattern Recognition (ICWAPR '07), Vol. 4*, pp. 1447–1452, 2007.

[37] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *2005 Ieee International Conference On Granular Computing*, vol. 2, pp. 718–721, IEEE, 2005.

[38] P. Vateekul and M. Kubat, "Fast induction of multiple decision trees in text categorization from large scale, imbalanced, and multi-label data," in *2009 Ieee International Conference On Data Mining Workshops*, pp. 320–325, IEEE, 2009.

[39] E. Gibaja and S. Ventura, "Multi-label learning: a review of the state of the art and ongoing research," *Wiley Interdisciplinary Reviews: Data Mining And Knowledge Discovery*, vol. 4, no. 6, pp. 411–444, 2014.

[40] J. M. Moyano, E. L. Gibaja, K. J. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: models, experimental study and prospects," *Information Fusion*, vol. 44, pp. 33–45, 2018.

[41] M. Han, H. Wu, Z. Chen, M. Li, and X. Zhang, "A survey of multi-label classification based on supervised and semi-supervised learning," *International Journal Of Machine Learning And Cybernetics*, vol. 14, no. 3, pp. 697–724, 2023.

[42] M. Mundt, Y. Hong, I. Pliushch, and V. Ramesh, "A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning," *Neural Networks*, vol. 160, pp. 306–336, 2023.

[43] S. Thrun, *Lifelong learning: A case study.* School Of Computer Science, Carnegie Mellon University, 1995.

[44] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[45] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[46] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catas-

trophic forgetting in neural networks," *Proceedings Of The National Academy Of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[47] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference On Machine Learning*, pp. 3987–3995, PMLR, 2017.

[48] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings Of The European Conference On Computer Vision (ECCV)*, pp. 532–547, 2018.

[49] R. Aljundi, M. Rohrbach, and T. Tuytelaars, "Selfless sequential learning," *Arxiv Preprint Arxiv:1806.05421*, 2018.

[50] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[51] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings Of The Aaai Conference On Artificial Intelligence*, vol. 32, 2018.

[52] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances In Neural Information Processing Systems*, vol. 32, 2019.

[53] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances In Neural Information Processing Systems*, vol. 30, 2017.

[54] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *Arxiv Preprint Arxiv:1812.00420*, 2018.

[55] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach, "Uncertainty-guided continual learning with bayesian neural networks," *Arxiv Preprint Arxiv:1906.02425*, 2019.

[56] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *Arxiv Preprint Arxiv:1606.04671*, 2016.

[57] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*, pp. 3366–3375, 2017.

[58] H. Arksey and L. O'Malley, "Scoping studies: towards a methodological framework," *International Journal Of Social Research Methodology*, vol. 8, no. 1, pp. 19–32, 2005.

[59] M. Roseberry, B. Krawczyk, Y. Djenouri, and A. Cano, "Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams," *Neurocomputing*, vol. 442, pp. 10–25, 2021.

[60] C. D. Kim, J. Jeong, and G. Kim, "Imbalanced continual learning with partitioning reservoir sampling," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pp. 411–428, Springer International Publishing, 2020.

[61] Y.-S. Liang and W.-J. Li, "Optimizing class distribution in memory for multi-label online continual learning," *Arxiv Preprint Arxiv:2209.11469*, 2022.

[62] N. Masuyama, Y. Nojima, C. K. Loo, and H. Ishibuchi, "Multi-label classification via adaptive resonance theory-based clustering," *Ieee Transactions On Pattern Analysis And Machine Intelligence*, 2022.

[63] T.-N. Pham, Q.-T. Ha, M.-C. Nguyen, and T.-T. Nguyen, "A probability-based close domain metric in lifelong learning for multi-label classification," in *Advanced Computational Methods For Knowledge Engineering: Proceedings Of The 6th International Conference On Computer Science, Applied Mathematics And Applications, ICCSAMA 2019 6*, pp. 143–149, Springer International Publishing, 2020.

[64] D. Dalle Pezze, D. Deronjic, C. Masiero, D. Tosato, A. Beghi, and G. A. Susto, "A multi-label continual learning framework to scale deep learning approaches for packaging equipment monitoring," *Engineering Applications Of Artificial Intelligence*, vol. 124, p. 106610, 2023.

[65] K. Du, F. Lyu, F. Hu, L. Li, W. Feng, F. Xu, and Q. Fu, "Agcn: augmented graph convolutional network for lifelong multi-label image recognition," in *2022 Ieee International Conference On Multimedia And Expo (ICME)*, pp. 01–06, IEEE, 2022.

[66] Y. Wang, N. J. Bryan, M. Cartwright, J. P. Bello, and J. Salamon, "Few-shot continual learning for audio classification," in *ICASSP 2021-2021 Ieee International Conference On Acoustics, Speech And Signal Processing (ICASSP)*, pp. 321–325, IEEE, 2021.

[67] C.-H. Chen, C.-H. Tu, J.-D. Li, and C.-S. Chen, "Defect detection using deep lifelong learning," in *2021 Ieee 19th International Conference On Industrial Informatics (INDIN)*, pp. 1–6, IEEE, 2021.

[68] S. Dong, H. Luo, Y. He, and X. W. Y. Gong, "Knowledge restore and transfer for multi-label class-incremental learning," *Arxiv Preprint Arxiv:2302.13334*, 2023.

[69] Y. Wang, Z. Wang, Y. Lin, L. Khan, and D. Li, "Cifdm: continual and interactive feature distillation for multi-label stream learning," in *Proceedings Of The 44th*

International Acm Sigir Conference On Research And Development In Information Retrieval, pp. 2121–2125, 2021.

[70] G. Song, K. Huang, H. Su, F. Song, and M. Yang, "Deep continual hashing for real-world multi-label image retrieval," *Computer Vision And Image Understanding*, p. 103742, 2023.

[71] J. Jia, F. He, N. Gao, X. Chen, and K. Huang, "Learning disentangled label representations for multi-label classification," *Arxiv Preprint Arxiv:2212.01461*, 2022.

[72] G. A. Carpenter and S. Grossberg, "Adaptive resonance theory," 2010.

[73] K. Du, L. Li, F. Lyu, F. Hu, Z. Xia, and F. Xu, "Class-incremental lifelong learning in multi-label classification," *Arxiv Preprint Arxiv:2207.07840*, 2022.

[74] D. D. Pezze, D. Deronjic, C. Masiero, D. Tosato, A. Beghi, and G. A. Susto, "A multi-label continual learning framework to scale deep learning approaches for packaging equipment monitoring," *Arxiv Preprint Arxiv:2208.04227*, 2022.

[75] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The Handbook Of Brain Theory And Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[76] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[77] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*, pp. 4510–4520, 2018.

[78] A. Géron, *Hands-On Machine Learning With Scikit-Learn, Keras, And Tensorflow*. " O'Reilly Media, Inc.", 2022.

[79] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *Arxiv Preprint Arxiv:1511.08458*, 2015.

[80] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *Arxiv Preprint Arxiv:1312.6114*, 2013.

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

[82] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications Of The ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[83] R. Gibbons and R. Gibbons, "A primer in game theory," 1992.

[84] C. Luo, "Understanding diffusion models: A unified perspective," *Arxiv Preprint Arxiv:2208.11970*, 2022.

[85] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings Of The IEEE/CVF Conference On Computer Vision And Pattern Recognition*, pp. 10684–10695, 2022.

[86] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International Conference On Machine Learning*, pp. 12888–12900, PMLR, 2022.

[87] A. Carta, L. Pellegrini, A. Cossu, H. Hemati, and V. Lomonaco, "Avalanche: A pytorch library for deep continual learning," *Journal Of Machine Learning Research*, vol. 24, no. 363, pp. 1–6, 2023.

[88] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances In Neural Information Processing Systems*, vol. 32, 2019.

[89] T. Maintainers and Contributors, "Torchvision: Pytorch's computer vision library." https://github.com/pytorch/vision, 2016.

[90] N. S. Detlefsen, J. Borovec, J. Schock, A. Harsh, T. Koker, L. D. Liello, D. Stancl, C. Quan, M. Grechkin, and W. Falcon, "Torchmetrics - measuring reproducibility in pytorch." https://www.pytorchlightning.ai, 2022.

[91] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[92] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[93] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[94] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal Of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[95] V. Stodden and S. Miguez, "Best practices for computational science: Software infrastructure and environments for reproducible and extensible research," *Available At SSRN 2322276*, 2013.

[96] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.

[97] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *Proceedings Of The ACM International Conference On Image And Video Retrieval*, pp. 1–9, 2009.

[98] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

[99] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal Of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[100] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?," *The Journal Of Machine Learning Research*, vol. 17, no. 1, pp. 152–161, 2016.

[101] M. A. Fligner, "A note on two-sided distribution-free treatment versus control multiple comparisons," *Journal Of The American Statistical Association*, vol. 79, no. 385, pp. 208–211, 1984.

[102] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings Of The Aaai Conference On Artificial Intelligence*, vol. 32, 2018.

APPENDIX

Critical values for the Friedman Test

$$M = \frac{12}{nk(k+1)} \sum R_j^2 - 3n(k+1)$$

	k=3		k=4		k=5		k=6	
n	α=5%	α=1%	α=5%	α=1%	α=5%	α=1%	α=5%	α=1%
2	—	—	6.000	—	7.600	8.000	9.143	9.714
3	6.000	—	7.400	9.000	8.533	10.130	9.857	11.760
4	6.500	8.000	7.800	9.600	8.800	11.200	10.290	12.710
5	6.400	8.400	7.800	9.960	8.960	11.680	10.490	13.230
6	7.000	9.000	7.600	10.200	9.067	11.870	10.570	13.620
7	7.143	8.857	7.800	10.540	9.143	12.110	10.670	13.860
8	6.250	9.000	7.650	10.500	9.200	13.200	10.710	14.000
9	6.222	9.556	7.667	10.730	9.244	12.440	10.780	14.140
10	6.200	9.600	7.680	10.680	9.280	12.480	10.800	14.230
11	6.545	9.455	7.691	10.750	9.309	12.580	10.840	14.320
12	6.500	9.500	7.700	10.800	9.333	12.600	10.860	14.380
13	6.615	9.385	7.800	10.850	9.354	12.680	10.890	14.450
14	6.143	9.143	7.714	10.890	9.371	12.740	10.900	14.490
15	6.400	8.933	7.720	10.920	9.387	12.800	10.920	14.540
16	6.500	9.375	7.800	10.950	9.400	12.800	10.960	14.570
17	6.118	9.294	7.800	10.050	9.412	12.850	10.950	14.610
18	6.333	9.000	7.733	10.930	9.422	12.890	10.950	14.630
19	6.421	9.579	7.863	11.020	9.432	12.880	11.000	14.670
20	6.300	9.300	7.800	11.100	9.400	12.920	11.000	14.660
∞	5.991	9.210	7.815	11.340	9.488	13.280	11.070	15.090

For values of ***n*** greater than 20 and/or values of ***k*** greater than 6, use χ^2 tables with ***k*-1** degrees of freedom

Figure 1: Friedman non-parametric hypothesis test

Table 1: Performance with different seed values

Exp ID	**Algorithm**	**mAP**	**OF1**	**CF1**
Seed = 15				
43474751	Naïve Baseline	59.57	4.93	4.73
43474751	EWC	78.29	63.72	62.05
43474751	LWF	77.24	61.07	58.35
43449205	MLDGR (Ours)	89.72	80.1	79.04
43474751	SI	80.52	69.42	68.29
Seed = 25				
43474766	Naïve Baseline	61.39	6.12	6.82
43474766	EWC	77.24	62.39	60.27
43474766	LWF	75.18	59.04	56.91
43467525	MLDGR (Ours)	89.27	79.29	78.47
43474766	SI	76.08	63.47	62.38
Seed = 40				
43474769	Naïve Baseline	60.03	2.92	3.64
43474769	EWC	77.16	61.35	59.87
43474769	LWF	76.18	59.86	58.07
43467533	MLDGR (Ours)	89.21	78.98	78.27
43474769	SI	79.98	68.45	67.34
Seed = 60				
43474773	Naïve Baseline	64.52	14.99	15.44
43474773	EWC	75.75	60.82	59.81
43474773	LWF	77.71	62.49	60.72
43467534	MLDGR (Ours)	89.13	77.9	77.65
43474773	SI	80.19	68.58	67.14
Seed = 85				
43474789	Naïve Baseline	65.81	5.21	5.08
43474789	EWC	79.25	66.23	64.59
43474789	LWF	77.26	61.92	59.64
43467536 / 43522258	MLDGR (Ours)	87.93	77.19	77.13
43474789	SI	79.91	68.44	67.05

	R = Range (Number of Groups)													
v_2	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*12*	*14*	*16*	*18*	*20*
6	3.46	4.34	4.90	5.30	5.63	5.90	6.12	6.32	6.49	6.79	7.03	7.24	7.43	7.59
	5.24	6.33	7.03	7.56	7.97	8.32	8.61	8.87	9.10	9.10	9.48	10.08	10.32	10.54
7	3.34	4.16	4.68	5.06	5.36	5.61	5.82	6.00	6.16	6.43	6.66	6.85	7.02	7.17
	4.95	5.92	6.54	7.01	7.37	7.68	7.94	8.10	8.37	8.71	9.00	9.24	9.46	9.65
8	3.26	4.04	4.53	4.89	5.17	5.40	5.60	5.77	5.92	6.18	6.39	6.57	6.73	6.87
	4.75	5.64	6.20	6.62	6.96	7.24	7.47	7.68	7.86	8.18	8.44	8.66	8.85	9.03
9	3.20	3.95	4.41	4.76	5.02	5.24	5.43	5.59	5.74	5.98	6.19	6.36	6.51	6.64
	4.60	5.43	5.96	6.35	6.66	6.91	7.13	7.33	7.49	7.78	8.03	8.23	8.41	8.57
10	3.15	3.88	4.33	4.65	4.91	5.12	5.30	5.46	5.60	5.83	6.03	6.19	6.34	6.47
	4.48	5.27	5.77	6.14	6.43	6.67	6.87	7.05	7.21	7.49	7.71	7.91	8.08	8.23
11	3.11	3.82	4.26	4.57	4.82	5.03	5.20	5.35	5.49	5.71	5.90	6.06	6.20	6.33
	4.39	5.15	5.62	5.97	6.25	6.48	6.67	6.84	6.99	7.25	7.47	7.65	7.81	7.95
12	3.08	3.77	4.20	4.51	4.75	4.95	5.12	5.27	5.39	5.62	5.80	5.95	6.09	6.21
	4.32	5.05	5.50	5.84	6.10	6.32	6.51	6.67	6.81	7.06	7.27	7.44	7.59	7.73
13	3.06	3.73	4.15	4.45	4.69	4.88	5.05	5.19	5.32	5.53	5.71	5.86	6.00	6.11
	4.26	4.96	5.40	5.73	5.98	6.19	6.37	6.53	6.67	6.90	7.10	7.27	7.42	7.55
14	3.03	3.70	4.11	4.41	4.64	4.83	4.99	5.13	5.25	5.46	5.64	5.79	5.92	6.03
	4.21	4.89	5.32	5.63	5.88	6.08	6.26	6.41	6.54	6.77	6.96	7.13	7.27	7.40
15	3.01	3.67	4.08	4.37	4.59	4.78	4.94	5.08	5.20	5.40	5.57	5.72	5.85	5.96
	4.17	4.84	5.25	5.56	5.80	5.99	6.16	6.31	6.44	6.66	6.85	7.00	7.14	7.26
16	3.00	3.65	4.05	4.33	4.56	4.74	4.90	5.03	5.15	5.35	5.52	5.66	5.79	5.90
	4.13	4.79	5.19	5.49	5.72	5.92	6.08	6.22	6.35	6.56	6.74	6.90	7.03	7.15
17	2.98	3.63	4.02	4.30	4.52	4.70	4.86	4.99	5.11	5.31	5.47	5.61	5.73	5.84
	4.10	4.74	5.14	5.43	5.66	5.85	6.01	6.15	6.27	6.48	6.66	6.81	6.94	7.05
18	2.97	3.61	4.00	4.28	4.49	4.67	4.82	4.96	5.07	5.27	5.43	5.57	5.69	5.79
	4.07	4.70	5.09	5.38	5.60	5.79	5.94	6.08	6.20	6.41	6.58	6.73	6.85	6.97
19	2.96	3.59	3.98	4.25	4.47	4.65	4.79	4.92	5.04	5.23	5.39	5.53	5.65	5.75
	4.05	4.67	5.05	5.33	5.55	5.73	5.89	6.02	6.14	6.34	6.51	6.65	6.78	6.89
20	2.95	3.58	3.96	4.23	4.45	4.62	4.77	4.90	5.01	5.20	5.36	5.49	5.61	5.71
	4.02	4.64	5.02	5.29	5.51	5.69	5.84	5.97	6.09	6.29	6.45	6.59	6.71	6.82
24	2.92	3.53	3.90	4.17	4.37	4.54	4.68	4.81	4.92	5.10	5.25	5.38	5.44	5.59
	3.96	4.55	4.91	5.17	5.37	5.54	5.69	5.81	5.92	6.11	6.26	6.39	6.51	6.61
30	2.89	3.49	3.85	4.10	4.30	4.46	4.60	4.72	4.82	5.00	5.15	5.27	5.38	5.48
	3.89	4.45	4.80	5.05	5.24	5.40	5.54	5.65	5.76	5.93	6.08	6.20	6.31	6.41
40	2.86	3.44	3.79	4.04	4.23	4.39	4.52	4.63	4.73	4.90	5.04	5.16	5.27	5.36
	3.82	4.37	4.70	4.93	5.11	5.26	5.39	5.50	5.60	5.76	5.90	6.02	6.12	6.21
60	2.83	3.40	3.74	3.98	4.16	4.31	4.44	4.55	4.65	4.81	4.94	5.06	5.15	5.24
	3.76	4.28	4.59	4.82	4.99	5.13	5.25	5.36	5.45	5.60	5.73	5.84	5.93	6.02
120	2.80	3.36	3.68	3.92	4.10	4.24	4.36	4.47	4.56	4.71	4.84	4.95	5.04	5.13
	3.70	4.20	4.50	4.71	4.87	5.01	5.12	5.21	5.30	5.44	5.56	5.66	5.75	5.83
∞	2.77	3.31	3.63	3.86	4.03	4.17	4.29	4.39	4.47	4.62	4.74	4.85	4.93	5.01
	3.64	4.12	4.40	4.60	4.76	4.88	4.99	5.08	5.16	5.29	5.40	5.49	5.57	5.65

Note: Studentized range q distribution table of critical values for $\alpha = .05$ and $\alpha = .01$.

Figure 2: Studentized range statistic