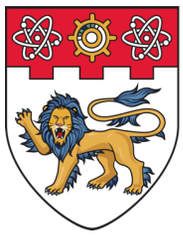
Deep Transfer Learning on Continual Learning



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Marcus Vinícius Sousa Leite de Carvalho

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2023

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

11/10/2023

.....

Date

.....

Marcus Vinícius Sousa Leite de Carvalho

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

11/10/2023

.....

Date

.....

Prof. Zhang Jie

Authorship Attribution Statement

This thesis contains material from 3 paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Marcus de Carvalho, Mahardhika Pratama, Jie Zhang, and Edward Yapp Kien Yee. ACDC: Online Unsupervised Cross-Domain Adaptation. Knowledge-Based Systems Journal, 2022

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. Prof Mahardhika Pratama and Prof Zhang Jie revised the manuscript.
- All programming, code optimization, and re-design of the benchmarks source code was conducted by me.
- I conducted item all experiments and revisions.

Chapter 4 is published as Marcus de Carvalho, Mahardhika Pratama, Jie Zhang, and Yajuan Sun. Class-Incremental Learning via Knowledge Amalgamation. In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD '22

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. Prof Mahardhika Pratama and Prof Zhang Jie revised the manuscript.
- I conducted all programming, code optimization, and re-design of the benchmarks source code.
- I conducted item all experiments and revisions.

Chapter 5 is submitted as Marcus de Carvalho, Mahardhika Pratama, Jie Zhang, Chua Haoyan and Edward Yapp. Toward Cross-Domain Continual Learning. Submitted to the proceedings of the International Conference of Data Engineering, ICDE '24

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof Mahardhika Pratama, Prof Zhang Jie and Dr. Edward Yapp.
- The concept idea was discussed between Chua Haoyan and me.

- I conducted all programming, code optimization, and re-design of the benchmarks source code.
- I conducted item all experiments and revisions.

11/10/2023

. . . . . .

Date

. . . . . .

Marcus Vinícius Sousa Leite de Carvalho

Acknowledgements

I want to express my heartfelt gratitude to the following individuals who have been instrumental in the success of my journey:

- My advisors, Professors Mahardhika Pratama and Zhang Jie, for their unwavering guidance, expertise, and support throughout my academic career. Their insights and mentorship have been invaluable in shaping my intellectual growth.
- My colleagues Weng Weiwei, Tushar Chouhan, Abhay Aradhya, Gaurav Sharma, Jakub Černý, Subhrajit Samanta, and Edward Yapp, for their constant encouragement, collaboration, and teamwork. I am grateful for the knowledge and skills I have gained from our discussions and projects, which have honed my professional abilities.
- For their unending love, laughter, and companionship, my friends Thiago Balducci, Lucas Menezes, Filipe Rios, Paula Motta, Marcus Ruzzon, Juliana Silva, Gabriel Mascarenhas, Mathews Howard Michels, Leander Michels, Vitória Faria, Erismar Moura, Tatiana Azevedo, Luiz Carlos, Rajarshi, Koh Yun Quan, Kelvin Ong, and Bian Qingtian. Their presence in my life has brought joy, comfort, and inspiration, and I cherish our shared memories.
- Lastly, I would like to thank my family - especially my grandmother Antônia, my auntie Eliane, my cousins Ellen and Erivânio, and my sister Lígia - for their unconditional love, support, and encouragement. Their unwavering belief in me has been a constant source of strength and motivation, and I owe my success to their selfless sacrifices and guidance.

Thank you all for being a part of my journey and making it memorable and fulfilling.

"É preciso amar as pessoas como se não houvesse amanhã, por que se você parar para pensar, na verdade não há."

—Pais e Filhos, Legião Urbana

Contents

List of Figures

List of Tables

Abstract

Artificial intelligent agents acting in the real world interact with a multitude of data streams. As a result, they must attain, accumulate and record various tasks from non-stationary data distributions. In addition, self-governing computational agents must acquire an understanding of new experiences and transfer knowledge from prior learning over a long period. Continual learning or lifelong learning refers to the capacity to continuously acquire new knowledge, adapt to changing circumstances, and integrate new experiences with previously learned ones over an extended span of time.

Catastrophic forgetting is the primary obstacle in computational models that use continual learning. It refers to the neural networks' tendency to disrupt the existing learned knowledge while being trained on new information. This leads to a sudden decline in performance when the new information overwrites the previous knowledge entirely or partially. The learning agents should assimilate the new information seamlessly to enhance their existing knowledge and prevent catastrophic forgetfulness. The model must be capable of preserving most or all of the acquired knowledge, ensuring that the new information does not hinder the previously acquired knowledge.

Computational models that engage in continuous learning are often designed to mimic the learning abilities of humans and other mammals. These animals can acquire, distill, and communicate knowledge over long periods of time. They benefit from various neurophysiological processes that facilitate the development of perception and motor skills through experience. Unlike machines, humans and other mammals can effortlessly acquire new skills and transfer knowledge between different domains and tasks. Moreover, the human brain has a remarkable ability to integrate multisensory information, allowing it to respond effectively in situations where there is sensory ambiguity and to draw on knowledge from different domains to achieve a common objective. Consequently, agents that engage in continual learning in the real world need to be able to deal with uncertainty, process

a continuous stream of multisensory data, and learn multiple tasks without disrupting their previously acquired knowledge. Achieving these goals has proven to be a persistent challenge for machine learning, neural network research, and the development of general intelligent systems.

Given this disparity, the primary aim of this investigation is to create cutting-edge algorithms that can manage numerous data distributions in a lifelong learning approach. The benefits are divided into three parts:

1. ACDC is an unsupervised learning framework that uses adversarial methods to handle multiple data streams. It is designed to address the challenge of learning in different domains simultaneously. ACDC's performance was evaluated using the prequential test-then-train protocol and compared to other baseline models. The results indicate that ACDC has better generalization capabilities and higher target accuracy, with a substantial improvement of over 10% in some cases.

2. The CFA approach proposes a solution to catastrophic forgetting by amalgamating knowledge from multiple specialist models. It exploits static agents that specialize in specific data domains by further combining them into a final model capable of handling all tasks simultaneously. CFA is a novel approach to continual learning that allows easy integration into existing static pipelines. It transforms them into continual learning systems with performance comparable or superior to the best existing static methods.

3. The CDCL framework aims to solve the unsupervised domain adaptation problem in the task-incremental learning scenario. CDCL has two unique mechanisms: an inter- and intra-task cross-attention block and an intra-task center-aware pseudo-labeling procedure. It utilizes the visual transformer architecture to facilitate the learning of several domains in a continual learning setup.

Thus, these novel techniques, frameworks, and algorithms are fundamental steps toward general intelligent agents that leverage their accumulated knowledge to obtain new information while preserving and adapting past knowledge.

Symbols and Acronyms

Symbols

General probability symbols

$x \in \mathbb{R}^u$	a sample with feature-space of size u, denote as sample
$X \in \mathbb{R}^{n \times u}$	a set of n samples with feature-space of size u, denoted as samples
$Y \in \mathbb{Z}^{n \times m}$	a set of n one-hot-vectors of size m, denoted as labels or annotations
$\mathcal{P}(y\|x)$	the conditional probability for label y given sample x
$\mathcal{P}(Y\|X)$	the conditional distribution for the set of labels Y given samples X
$H(\cdot, \cdot)$	the cross-entropy function
$H(\cdot)$	the entropy function
$E[\cdot]$	the expected value
$\mathcal{N}(\mu, \sigma^2)$	the Gaussian distribution
x_i	the i-th sample in a set of samples
y_i	the i-th label in a set of labels
$\mathcal{P}(X, Y)$	the joint distribution from a set of samples X and labels Y
$\mathcal{P}(x, y)$	the joint probability for sample x and label y
$y \in \mathbb{Z}^m$	the label or annotation of a sample in the one-hot-vector of size m format
$\mathcal{P}(X)$	the marginal distribution of the set of samples$X$
$\mathcal{P}(x)$	the marginal probability of sample x
$\hat{Y} \in \mathbb{R}^{n \times m}$	the predicted labels of a set of samples
$\hat{y} \in \mathbb{R}^m$	the predicted label of a sample
μ	the samples mean
σ	the samples standard deviance
σ^2	the samples variance

Deep learning related symbols

$\mathcal{D}$	a dataset
$\mathcal{T}$	a task, i.e., the type of inference being made according to the problem
$\mathcal{D}_i$	the i-th dataset in a set of datasets
$\mathcal{T}_i$	the i-th task in a set of tasks
b	the bias (parameters decomposition) of the model
λ	the learning rate
r^*	the least significant hidden node in a layer
η	the momentum rate
θ	the parameters of the model
$\mathcal{R}$	the total number of hidden nodes in a model
$\mathcal{W}$	the weights (parameters decomposition) of the model
$d_{\mathcal{H}\Delta\mathcal{H}}$	the $\mathcal{H}\Delta\mathcal{H}$ divergence function
$\mathcal{L}(\,\cdot\,)$	the loss function, also denoted as error loss or objective function
$\mathcal{L}_{\text{mse}}(\,\cdot\,,\,\cdot\,)$	the mean-squared error loss
$\mathcal{L}_{\log}(\,\cdot\,,\,\cdot\,)$	the multi-class logarithmic loss
$\Phi(\,\cdot\,)$	the probit function
$\sigma(\,\cdot\,)$	the sigmoid function

Transfer learning related symbols

$x_{\mathcal{S}}$	a sample from the source dataset/stream
$x_{\mathcal{T}}$	a sample from the target dataset/stream
$y_{\mathcal{S}}$	the label of a source sample
$n_{\mathcal{S}}$	the number of samples in the source dataset/stream
$n_{\mathcal{T}}$	the number of samples in the source dataset/stream
$\hat{y}_{\mathcal{S}}$	the predicted label of a target sample
$\mathcal{T}_{\mathcal{S}}$	the source task
$\mathcal{D}_{\mathcal{S}}$	the source dataset, usually containing samples and annotations
$\mathcal{T}_{\mathcal{T}}$	the target task
$\mathcal{D}_{\mathcal{T}}$	the target dataset, usually containing only samples, without annotations
$\tilde{x}$	a corrupted/noised sample, input of a denoising auto-encoder

$g(\cdot)$	a decoder function, part of an auto-encoder
$\hat{x}$	a reconstructed sample, output of an auto-encoder
$\hat{x}_{\mathcal{S}}$	a reconstructed source sample
$\hat{x}_{\mathcal{T}}$	a reconstructed target sample
Z	a set of soft outputs, also referred to as logits
$h(\cdot)$	an encoder function, part of an auto-encoder
z	the soft output of the model, also referred to as logit
$\mathbb{P}$	a set containing pairs of source and target samples
ϵ	the error bound of a model
C^*	the error of an optimal classifier
D_{KL}	the Kullback-Leibler divergence
ϵ_S	the source domain error bound of a model
ϵ_T	the target domain error bound of a model

Knowledge amalgamation related symbols

T_i	the i-th teacher model
F_{T_i}	the original features of a teacher model
F_S	the original features of the student model
f_{T_i}	the aligned features of a teacher model
f_S	the aligned features of the student model
$\hat{f}_{T_i}$	the projected features of a teacher model into a common space
$\hat{f}_S$	the projected features of the student model into a common space

Online learning related symbols

$\mathcal{S}_{[0,t]}$	a set of samples from a time period $[0, t]$
$\mathcal{P}_{[0,t]}(X, Y)$	the distribution from a set of samples from a period $[0, t]$
$W'_{\mathcal{S}}$	the paired source stream sliding window
$W'_{\mathcal{T}}$	the paired target stream sliding window
$W_{\mathcal{S}}$	the source stream sliding window
$W_{\mathcal{T}}$	the target stream sliding window
$n_{W_{\mathcal{S}}}$	the size of the source stream sliding window

$n_{W_\mathcal{T}}$	the size of the target stream sliding window
$d'_\mathcal{S}$	a flag indicating that the sample is from the source dataset/stream
$d'_\mathcal{T}$	a flag indicating that the sample is from the target dataset/stream

Continual learning related symbols

x^R	an example of rehearsed sample
$R \in \mathbb{R}^{T \times T}$	the continual learning test classification matrix
M	the memory used to hold rehearsal samples
$\mathcal{S}_{\text{past}}$	the past knowledge of a continual learning model
$\bar{b}_i$	the random performance of a continual learning model on task i

Transformers related symbols

$\mathbf{K} \in \mathbb{R}^{n \times d}$	the key vector projection
$\mathbf{b}_i \in \mathbb{R}^{1 \times d}$	the task-related bias vector projection
$\mathbf{K}_i \in \mathbb{R}^{n \times d}$	the task-related key vector projection
$\mathbf{K}_i^T \in \mathbb{R}^{n \times d}$	the task-related key vector projection from the target dataset
$\mathbf{Q} \in \mathbb{R}^{n \times d}$	the query vector projection
$\mathbf{Q}_S \in \mathbb{R}^{n \times d}$	the query vector projection from the source dataset
$\mathbf{V} \in \mathbb{R}^{n \times d}$	the value vector projection
$\mathbf{V}_T \in \mathbb{R}^{n \times d}$	the value vector projection from the target dataset

Miscellaneous symbols

$\mathbf{c}_k$	a k-means clustering centroind
$O(\cdot)$	order of magnitude or ergodic convergence rate (running average)
$\|\cdot\|$	the 2-norm of a vector or matrix in Euclidean space

Acronyms

AI	Artificial Intelligence
AV	Autonomous Vehicle

CNN	Convolution Neural Network
CL	Continual Learning
HL	Hidden Layer
ML	Machine Learning
NLP	Natural Language Programming
OL	Online Learning
SGD	Stochastic Gradient Descent
SVM	Support Vector Machines
TL	Transfer Learning

ACDC	Autonomous Cross-Domain Conversion
CFA	Catastrophic Forgetting Solution based on Knowledge Amalgamation
CPU	Central Processing Unit
CDCL	Cross-Domain Continual Learning
GPU	Graphics Processing Unit

AGMM	Autonomous Gaussian Mixture Model
DAE	Denoising Autoencoder
DISC	Discriminator
DAA	Domain Adversarial Adaptor
GRL	Gradient Reverse Layer
SPC	Statistical Process Control
DA	Domain Adaptation
KA	Knowledge Amalgamation
KD	Knowledge Distillation
KL	Kullback-Leibler
MMD	Maximum Mean Discrepancy
SDA	Supervised Domain Adaptation
SSDA	Semi-Supervised Domain Adaptation
UDA	Unsupervised Domain Adaptation

ACC	Average Accuracy
FGT	Average Forgetting
BWT	Backward Transfer
CIL	Class-Incremental Learning

DIL	Domain-Incremental Learning
FWT	Forward Transfer
TIL	Task-Incremental Learning
AM(X)	Amazon dataset
CF	CIFAR10 dataset
LD	London Bike dataset
MN	MNIST dataset
ST	STL10 dataset
US	USPS dataset
WA	Washington Bike dataset
Conv2d	A 2D convolutional operation
MaxPool	the maximum pooling operation
ReLU	the rectified linear unit function
Softmax	the soft maximum operation
i.i.d.	independent and identically distributed

Chapter 1

Introduction

When intelligent agents function in the real world, they encounter various streams of information, making it essential to learn and retain multiple tasks through constant changes in the data distributions. Additionally, computational agents must gain knowledge from new experiences and use the knowledge from past learning for an extended period. *Lifelong learning* refers to the ability to keep acquiring new knowledge over time by incorporating it with previously learned experiences.

Lifelong learning computational models face a significant obstacle in *catastrophic forgetting*. This refers to the tendency for artificial neural networks to interfere with prior knowledge when trained on new information, resulting in a sudden decrease in performance [2–4]. Furthermore, static deep neural networks rely on batch-setting training, which assumes that all training data are accessible and known in advance [5, 6]. When new knowledge is introduced, the entire network parameters must be retrained, which is inefficient and hinders real-time learning. Hence, conventional neural networks trained on incremental tasks show a significant decrease in performance as new tasks are learned [7, 8]. Besides, in scenarios where the autonomous agents must actively interact with the environment to learn on the go, training and testing phases merge, requiring the agent to adapt and trigger behavioral responses simultaneously [9, 10].

To prevent catastrophic forgetting, it is crucial for learning agents to continually assimilate novel data, thereby enhancing their existing understanding. In doing so, the incorporation of new insights must not impede or disrupt the previously learned information. Rather, it should aim to conserve the majority, if not the

entirety, of prior knowledge. This delicate equilibrium between an agent's requirement to be flexible enough to adopt new data and stable enough to protect acquired information is referred to as the stability-plasticity dilemma.

The stability-plasticity quandary has been a subject of considerable investigation across both biological and computational domains. Researchers have explored this conundrum in an effort to better understand the mechanisms that govern learning systems and their capacity to adapt without compromising established knowledge. The extensive body of work in this area can be found in various studies, such as the survey by Ditzler et al. [11].

Humans and other mammals exhibit the remarkable capacity for ongoing learning and decision-making grounded in previous experiences [10, 12]. This enduring learning process is made possible by intricate neurophysiological mechanisms that facilitate the gathering, honing, and application of knowledge over long durations. These mechanisms play a pivotal role in the cultivation and specialization of sensory perception and motor abilities, which are molded through experience [13–16].

As a result, the multifaceted nature of continuous learning has prompted researchers to draw inspiration from the mammalian brain's learning factors in developing computational methods. By examining the intricacies of these biological processes, scientists hope to emulate their efficiency and effectiveness in artificial systems, paving the way for more sophisticated and adaptive learning algorithms.

The past few years have witnessed a growing interest in lifelong learning, particularly because of its significance for self-governing learning agents and robots. Although neural network approaches are created to conform to data samples incrementally gathered in controlled surroundings, presented in a haphazard manner, these methods are far from the real-life conditions that humans and other creatures encounter throughout their lifetimes [9, 17–19]. Agents that learn throughout their lifespan are forced to deal with sensory uncertainty, manage ongoing streams of multisensory data, and competently pick up multiple tasks without disrupting previously acquired knowledge. As a result, there is a vast disparity between current neural network models and the more advanced lifelong learning agents that are expected to learn incrementally from their continuous sensorimotor experiences.

When it comes to *transfer learning* [20], humans have the ability to learn new skills and apply their knowledge to different domains and tasks without much difficulty,

whereas artificial systems are still in the nascent stages of mastering this ability [21]. Furthermore, the human brain greatly benefits from the integration of information from multiple senses, which helps in efficient problem-solving when there is a lack of sensory clarity. The brain also has the ability to repurpose knowledge from related domains towards the achievement of a common goal [12, 14, 22, 23].

Lifelong learning initiatives have posed a persistent difficulty for artificial intelligence systems, and therefore, for the advancement of machine learning and neural networks [24, 25].

1.1 Research objectives

The main objective of this research is to develop state-of-the-art algorithms capable of handling multiple data distributions in a lifelong learning manner. The objectives of this research work are as follows:

- **Handle multiple data streams from different data domains**: By handling multiple data streams from different domains, we want to generate learning agents that can acquire sequential knowledge from multiple experiences that share the same task while only one experience is labeled.
- **Alleviate the catastrophic forgetting problem in incremental learning with transfer learning techniques**: There are many ways to use transfer learning techniques. While existing work aims to network pre-training and fine-tuning tasks to enhance incremental learning [26], we want to use transfer learning to build models that can extract knowledge from previously trained models (previously acquired knowledge).
- **Alleviate the catastrophic forgetting problem in incremental learning multiple domains with transfer learning techniques**: By taking advantage of previously learned tasks, we want to build learning agents that can incrementally learn multiple complementary domains simultaneously.
- **Alleviate the catastrophic forgetting problem while handling multiple incremental learning data streams**: Finally, by combining all objectives above, we can build an agent that can handle incremental learning (continuous learning) of multiple complementary data source domains.

1.2 Major Contributions

Our main contributions can be stated as follows:

- *Autonomous Cross-Domain Conversation*: The Autonomous Cross-Domain Conversation (ACDC) framework represents a groundbreaking solution to the unsupervised cross-domain adaptation problem. By utilizing a dynamic architecture, it actively adapts to data shifts throughout the learning process. The ACDC framework is composed of three primary components: a denoising autoencoder that functions as a feature extractor, a domain adversarial adaptation network that facilitates cross-domain adaptation, and a classifier in charge of inference tasks.

  ACDC's flexible online neural network framework can be adapted to accommodate a diverse array of network structures, such as convolution layers, distributed computing, recurrent layers, and varying levels of layer depth. This adaptability allows for seamless integration with various network designs, showcasing the framework's versatility in addressing the complex challenges of cross-domain adaptation.

- *Catastrophic Forgetting Solution via Knowledge Amalgamation*: A novel method for addressing catastrophic forgetting involving multiple static agents. The solution, known as the catastrophic forgetting solution via knowledge amalgamation (CFA), has the capability to combine the learned representation of multiple diverse trained teacher models. Each of these models is focused on a previous task, and the resulting single-headed student model can handle all tasks at once. Additionally, CFA allows easy integration into current static learning pipelines, facilitating a smooth transition from offline to continual learning.

- *Towards Cross-Domain Continual Learning*: A novel framework for Cross-Domain Continual Learning (CDCL) to solve the unsupervised cross-domain task-incremental learning problem while maintaining a good stability-plasticity trade-off. CDCL proposes two methods to solve the feature-alignment catastrophic forgetting problem: an inter- intra-task cross-attention mechanism

that aligns domain features in current tasks and consolidates previous alignment knowledge; and an intra-task-based center-aware pseudo-labeling strategy that identifies similar task-specific samples between the domains.

- *Towards Cross-Domain Continual Learning*: A novel approach to tackle the challenge of unsupervised cross-domain task-incremental learning problem, named cross-domain continual learning (CDCL) framework. Its primary focus is to maintain a stable balance between adaptability and stability while solving the feature-alignment catastrophic forgetting problem. CDCL offers two contributions to address this problem: an inter- and intra-task cross-attention mechanism that aligns domain features in current tasks and consolidates previous alignment knowledge; and an intra-task-based center-aware pseudo-labeling strategy that identifies similar task-specific samples between the domains.

1.3 Outline of the thesis

This thesis is organized as follows:

In Chapter 1, the concepts of lifelong learning and the central challenge of catastrophic forgetting are introduced. The impetus for this research is the creation of computational agents that can consistently gather, polish, and utilize knowledge and skills throughout their existence, akin to humans and other mammals. Furthermore, the chapter delineates the research objectives and the contributions of the present study.

Chapter 2 offers an in-depth examination of the lifelong learning literature. A minor distinction between online learning and continual learning is discussed, along with a thorough analysis of the features and challenges of each learning type. Ultimately, the difficulties encountered in previous studies in the fields of *lifelong learning* and *transfer learning* serve as the motivation for the current research.

Chapters 3, 4, and 5 provide comprehensive accounts of various studies undertaken in this research to tackle the challenges and issues of applying *transfer learning* to neural network continual learning. A concise summary of these chapters is given below.

Chapter 3 acquaints readers with the idea of Autonomous Cross-Domain Conversion (ACDC). This framework represents a form of adversarial unsupervised cross-domain adaptation that employs a self-adapting neural network architecture to manage multiple data streams. Specifically, the framework is designed to handle diverse non-stationary data streams that span different domains and feature spaces.

Chapter 4 presents an innovative approach for incrementally learning new tasks through the knowledge amalgamation of heterogeneous teachers, termed *Catastrophic Forgetting Solution Based on Knowledge Amalgamation* (CFA). This post-processing continual learning method proposes knowledge amalgamation as a means to merge the expertise of heterogeneous teacher models into a single student model, capable of addressing all tasks simultaneously.

Chapter 5 introduces a groundbreaking framework for incorporating unsupervised cross-domain capabilities into an incremental learning model, named *Towards Cross-Domain Continual Learning* (CDCL). This visual transformer-based method resolves the feature alignment catastrophic forgetting issue.

Lastly, Chapter 6 summarizes the conclusions drawn from the studies presented in this thesis by emphasizing the main contributions, potential extensions of the current work, and future research directions.

Chapter 2

Literature Review

Machine learning (ML) has been crucial in advancing data analysis, artificial intelligence, and decision-making algorithms in general [27]. More recently, deep learning has demonstrated extraordinary achievements, surpassing humans in many activities [28–33]. As a result, almost all areas of computer science, engineering, natural sciences, and social sciences have used ML algorithms. In addition, many modern industries - such as logistics, retail analytics, lifestyle, and drug discovery - would not have been disrupted without the practical usage of ML algorithms [34–37].

The majority of machine learning (ML) algorithms work by training intelligent agents using a specific dataset and then applying the acquired knowledge to practical tasks. This prevalent approach, often referred to as static, offline, or isolated learning, is observable in both supervised and unsupervised learning paradigms. Nevertheless, the fundamental drawback of the offline learning method is its susceptibility to catastrophic forgetting when employed in dynamic environments. In other words, it fails to maintain and utilize previously learned knowledge for future learning, resulting in a significant performance decline in earlier tasks. In contrast, humans and animals are capable of retaining past knowledge and leveraging it to enhance future learning and problem-solving [12, 14, 21–23].

Usually, ML algorithms require a significant amount of training data to achieve efficient learning without relying on prior knowledge. Consequently, when new information is introduced, the entire network parameters need to undergo a retraining process. This issue persists even in unsupervised learning settings, where accumulating vast amounts of data may be impractical in numerous situations.

FIGURE 2.1: Diagram depicting how a static intelligent agent learns and how a human learns. A human accumulates knowledge over time, while the static agent suffers catastrophic forgetting when encountering new knowledge.

As a result, traditional ML algorithms face limitations in adapting to dynamic environments that require continual knowledge accumulation and application.

Researchers have proposed transfer learning techniques to solve the data availability problem, including few-shot learning, one-shot learning, and zero-shot learning [20, 38]. However, a system must have prior domain knowledge for such techniques. The majority of the information typically comes from human users, either through the design engineer or an interactive learning process [39], i.e., learning the model still has to rely on humans instead of learning by itself. Therefore, learning and accumulating prior knowledge from past tasks is more desirable [24, 25].

2.1 Lifelong Learning

Humans learn by accumulating and maintaining previously learned tasks and then use them seamlessly in learning new tasks and solving new problems, becoming more knowledgable and more effective at learning over time [9, 17–19]. Humans and other mammalians feel this learning style is natural because their brains can closely relate to and interconnect things around them with previous experiences. As an example, building an accurate animal binary classifier through machine learning algorithms would require a larger dataset of dog and cat images (e.g., 1,000 samples per class) compared to what would be necessary for human use. In other words, humans do not require as many dog and cat pictures as an ML algorithm would need for effective classification. A human would probably be able to perform this classification task without a single training example, with a possible chance of enhancing it for other similar classes, e.g., wolfs and tigers. The reason is that humans have accumulated knowledge in the past - in this case, about animals in

general - and can create some internal mental connection - in this case, canines and felines.

To broaden the scope of the argument, we employ an illustration from the realm of self-driving cars (AVs) as well as one from the area of processing language naturally (NLP).

Example 2.1.1. *Lifelong learning for AVs*: Most AVs are being designed to drive into cities and countryside roads, where multiple aspects are common around the world:

1. Roads have directions.
2. Vehicles operating on roads must follow the law reinforced by road signs.
3. Roads can be referred to by their name, code, localization, and nearby landmarks.

Although some countries and regions can have specific laws, referred to by (2), we can assume that there is a great intersection between them. A human who grew up in London, UK, and learned to drive in the country understands the above-mentioned three items. When this same individual decides to cross the Channel Tunnel to France, they can adapt to a scenario – such as driving on a different side of the road, new road sign designs, and new landmarks – instantaneously by relying on their knowledge and experience driving in London, even though there are significant differences and novelties introduced.

Example 2.1.2. *Lifelong learning for NLP*: Lifelong learning for NLP is crucial for several reasons:

1. The syntax or grammar of sentences follows the same pattern in all domains.
2. The significance of terms and expressions remains consistent across various fields and activities.
3. NLP issues, such as relationship between words, are closely related and have varying degrees of impact on each other.

The first and second factors enable the application of knowledge learned across domains and tasks. Humans do not need to learn a new language when they encounter a new application domain. The third factor allows for lifelong learning across various tasks. For example, Giorgi et al. [40] discovered that named entity recognition (NER) and relation extraction (RE) analysis are critical tasks in information extraction and retrieval, and NER systems benefit significantly from RE analysis.

Almost every piece of knowledge identified by humanity is interconnected; hence, information acquired in certain fields can be utilized in similar scenarios in other areas in the future. The traditional offline learning method is incapable of achieving this lifelong learning approach, which is crucial to create intelligent systems that can learn and advance continuously like humans. However, lifelong learning is gaining significance, especially in the fields of AI and ML, to adapt to our dynamic surroundings and develop intelligent agents that learn and evolve more efficiently with time.

Definition 2.1 (Lifelong learning)**.** *Lifelong learning* involves an ongoing acquisition of knowledge. At any given moment, the individual engaged in this learning endeavor has assimilated a sequence of tuples $S_{\text{past}} = \{(\mathcal{T}_1, \mathcal{D}_1), (\mathcal{T}_2, \mathcal{D}_2), ..., (\mathcal{T}_N, \mathcal{D}_N)\}$, also known as *previous tasks*, where $\mathcal{T}_i$ and $\mathcal{D}_i$ are respectively a task and its corresponding dataset, and usually $(\mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_N) = \{\emptyset\}$. When faced with the new task tuple $(\mathcal{T}_{N+1}, \mathcal{D}_{N+1})$ - also known as the *new* or *current task* -, the learner can leverage the *past knowledge* S_{past} to help learn $\mathcal{T}_{N+1}$. The learning of $\mathcal{T}_{N+1}$ consists of checking, reasoning, transfer, and meta-mining of additional higher-level knowledge.

Given the expansive nature of this definition, it is crucial to emphasize the following points:

- The definition underscores three essential aspects of lifelong learning models: (1) ongoing learning, (2) accumulation and preservation of knowledge, and (3) the ability to apply prior knowledge for facilitating future acquisition of knowledge. An individual committed to continuous learning participates in a series of tasks that have the potential to be endless, and gradually enhances their knowledge and proficiency in acquiring knowledge. These attributes

set lifelong learning apart from related learning paradigms like multi-task learning [41], transfer learning [20], and multi-target learning [42], which may lack one or more of these characteristics.

- Tasks are not necessarily limited to the same domain.
- The transition to a different task may happen abruptly or gradually. Neither an oracle nor any external source is required to provide the tasks or data. Ideally, a lifelong machine should be capable of discovering novel learning tasks and training data by interacting with its environment and engaging in self-directed learning.
- Our comprehension of knowledge and its representation is limited, which is why the definition does not provide specific details on these aspects. Presently, studies generally utilize only a small number of knowledge types that are most appropriate for their intended approaches [43]. The representation of knowledge poses a significant research challenge, with few fully developed outcomes that can be implemented in practice. Furthermore, the description lacks guidance on how to safeguard and refresh the obtained knowledge.
- According to the definition, an effective approach to lifelong learning necessitates a systematic combination of diverse learning algorithms and knowledge representation schemes. It is highly unlikely for a solitary learning algorithm to attain the lifelong learning goal [43].
- At present, no universal lifelong learning system exists that can perform lifelong learning in any conceivable domain or for any type of task.

For the rest of this report, we will break the lifelong learning definition into two directions. First, although many researchers use these terms as synonyms, here we will use them as a way to define our problem formulations, making it easier to guide our research:

- **Online learning**, also known as *sequential learning* or data stream mining, is the process of extracting knowledge from continuous, rapid data samples generated from a non-stationary environment without being able to revisit a previously learned sample. Often, the data stream suffers from concept drifts over time, representing changes in the underlying data distribution

[44]. Unlike offline learning, we treat all incoming chunks of data as holdout sets, evaluating them in a *prequential test-then-train* process.[1]

- **Continual learning**, also known as *incremental learning*, is the processing of learning knowledge from new tasks while preserving previous knowledge and avoiding catastrophic forgetting. When a continual learning model knows all the future tasks, we call it a *task incremental learning* setting, usually by receiving information regarding the task upon evaluation. Contrary to that, when a continual learning model does not know future tasks, we call it a *class incremental learning*, forcing the model to evolve according to incoming data [45].

2.1.1 Online learning

Online learning (OL) refers to a method of learning where data points are presented sequentially and at a rapid pace, without any restrictions on their number. As each new data point arrives, the model must respond immediately, and then be updated quickly to prepare for the next data sample. Moreover, the data-generating process is usually non-stationary and susceptible to *concept drifts*, which makes it necessary for the learning agent to adjust accordingly. This method differs from the conventional approach of offline learning, where all of the training information is readily accessible from the beginning.

In OL, it would be prohibitively expensive to retrain the entire data set every time a new data point is added. Additionally, the rapid flow of data streams means that the model would become obsolete during the retraining process. As a result, OL techniques are usually designed to be efficient in terms of memory usage and processing time in order to meet the real-world demands of low latency.

Numerous OL algorithms are presently available. The majority of these models are learning algorithms based on kernels, such as SVM [46]. It is typical for OL algorithms to enhance the traditional stochastic gradient descent (SGD) with computationally efficient methods for classification, regression, and novelty detection. The literature also showcases online variational Bayes algorithms [47] and online

[1] Multiple researchers treat online learning as a field itself, separated from lifelong learning. Here, we consider it an essential part of lifelong learning because a generic lifelong learner should be able to learn - or adapt its knowledge - in real-time based on interactions with the real world.

learning on graphs [48]. Predicting users' preferences towards products in a social network is one of the applications of this problem. Lastly, existing techniques also employ neural networks to address the online learning problem [49–51].

Numerous online algorithms utilize adaptation mechanisms to manage fluctuations in the distribution of data streams [49, 50, 52–55]. For example, certain methods may activate evolution mechanisms, such as generating new models from the ground up, a common tactic in ensemble solutions. These techniques can then undergo retraining using a sliding window or memory [54, 56].

On the other hand, passive drift detection techniques gradually adjust the model to accommodate any changes that might occur in the data streams. Consequently, data stream mining has achieved substantial progress across various domains, including regression [57, 58], clustering [59], and classification [49–51, 60, 61]. This advancement demonstrates the effectiveness of adaptation mechanisms in addressing the challenges posed by dynamic data stream environments.

When it comes to transferring learning in the online domain, we are particularly interested in learning from multiple streams at the same time. Most online learning research currently focuses on a single domain or task. Ensemble methods based on SVMs have been proposed by researchers to handle multiple data streams [60–62]. Du et al. [63, 64] developed ensemble-based random forest solutions to handle non-stationary learning from multiple sources. Dredze and Crammer [65] developed a multi-domain online learning method that combines multiple classifiers' parameters. The model considers both the new instance/example and its domains. Meanwhile, Li et al. [66], Tao et al. [67] proposed multi-domain online learning methods that use a warm-up initialization period and an SVM ensemble to handle different concepts.

While OL deals with upcoming data in a sequential or streaming manner, its purpose is distinct from lifelong learning. OL still undertakes learning and inference over the same task over time, even though it aims to learn more effectively with the sequential arrival of data. On the other hand, the objective of lifelong learning is to gain knowledge through a variety of tasks, maintain the knowledge acquired thus far, and utilize it to ease the acquisition of knowledge in subsequent tasks.

2.1.1.1 Concept Drifts

A myriad of products and services generate vast amounts of real-time data, requiring efficient data analysis and machine learning techniques to enable accurate predictions and decisions. *Concept drift* describes a situation in which the statistical properties of the target variable, which the model aims to predict, shift unexpectedly over time [68, 69]. In such circumstances, previously learned data patterns may no longer be relevant to incoming samples, resulting in inferior performance. Concept drifts often originate from hidden variables that are not directly measurable [70, 71] and are the main reason for the declining effectiveness of various data-driven information systems, such as decision support and data-driven early warning systems [54].

In a constantly changing big data landscape, the need for more reliable data-driven predictions and decision-making capabilities has become essential. Addressing the challenges posed by concept drifts is critical to maintaining the accuracy and relevance of machine learning models in real-world applications, ensuring that they continue to deliver meaningful insights and support informed decision-making.

The study on concept drift aims to address various difficulties, including the precise identification of concept drift in datasets that are unorganized and contain a lot of noise [72, 73]. It also involves gaining a quantitative and comprehensible understanding of concept drift [69, 71], as well as devising strategies to adjust the learning approach in response to drift [74, 75]. Overcoming these challenges enables flexible forecasting and decision-making in an environment where uncertainty is prevalent.

The implementation of methods for managing concept drift has resulted in a noteworthy enhancement of learning agents in the fields of data science, pattern recognition, and data stream mining. Concept drift is a prevalent and consequential concern in environments that have access to a large amount of data, such as data streams or the real world, due to the unpredictability that arises from the fundamental characteristics of big data.

In academic literature, the phenomenon known as concept drift is also referred to as dataset shift [76] or concept shift [72]. Nevertheless, the majority of recent publications, including this particular study, utilize the term concept drift to describe

the overall issue where there exists a point in time where the probability distribution of input and output data changes, i.e., $\exists t : P_t(X, Y) \neq P_{t+1}(X, Y)$ [69], where $P_t(\cdot)$ represents the probability distribution in a time interval t.

Definition 2.2 (Concept Drift)**.** Assuming $S_{[0,t]} = \{(x_i, y_i)\}_{i=0}^{t}$ is a set of samples from a time period $[0, t]$, where x_i represents the feature vector and y_i represents the label, and $P_{[0,t]}(X, Y)$ represents a distribution for $S_{[0,t]}$. We can say that if the distribution of $S_{[0,t]}$, $P_{[0,t]}(X, Y)$, is not equal to the distribution for $S_{[t+1,\infty]}$, $P_{[t+1,\infty]}(X, Y)$, then there is a concept drift at timestamp $t + 1$. Therfore, concept drift is denoted as $\exists t : P_t(X, Y) \neq P_{t+1}(X, Y)$, as given in Lu et al. [73].

Given that the probability $P_t(X, Y)$ can be expressed as the product of two probabilities, $P_t(X)$ and $P_t(Y|X)$, concept drift can be caused by the following [75]:

- Virtual drift: $P_t(X)$ changes while $P_t(Y|X)$ remains the same. This type of drift does not affect the decision boundary.
- Natural drift: $P_t(Y|X)$ changes while $P_t(X)$ remains the same. This type of drift causes changes in the decision boundary and decreases learning performance.
- Mixed drift: Both $P_t(X)$ and $P_t(Y|X)$ change. This type of drift conveys crucial information about the learning environment.

Concept drift can also be categorized into four types [77]:

- **Abrupt/Sudden Drift**: A new concept appears within a short interval.
- **Gradual drift**: An old concept is gradually replaced by a new one.
- **Incremental drift**: An old concept changes gradually to a new one.
- **Recurring concepts**: An old concept reappears after a period of time.

The term *intermediate concept* has been introduced to demonstrate better the differences between these types and the transformation between the concepts themselves [77].

2.1.2 Continual Learning

The primary objective of continual learning (CL) is to create learning agents that can learn a sequence of tasks or classes without revisiting previous data. These agents must use their prior knowledge to learn new tasks more efficiently. Continual learning poses two significant challenges: backward and forward knowledge transfer [45]. Backward transfer (BWT) evaluates how learning a current task affects the performance of previously learned tasks. If BWT is negative, it can lead to catastrophic forgetting [78]. A common alternative for the BWT metric is the average forgetting metric (FGT) [79], which gives a numerical indicator of how much the cumulative knowledge of the model has been forgotten. Forward transfer (FWT) assesses how learning a current task impacts the performance of future, unseen tasks. Positive FWT enables continual learning methods to leverage shared representations, resulting in more effective training on new tasks. BWT and FWT determine how effectively CL methods can adapt and refine feature representations. Previous studies have focused mostly on classification tasks because of the difficulty in avoiding negative BWT and achieving positive FWT [4, 45, 80–82].

Which metrics are such:

$$\textbf{Average Accuracy: } \text{ACC} = \frac{1}{T}\sum_{i=1}^{T} R_{T,i} \tag{2.1}$$

$$\textbf{Backward Transfer: } \text{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i} \tag{2.2}$$

$$\textbf{Forward Transfer: } \text{FWT} = \frac{1}{T-1}\sum_{i=2}^{T} R_{i-1,i} - \bar{b}_i \tag{2.3}$$

$$\textbf{Average Forgetting: } \text{FGT} = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{i\in 1,\cdots,T-1} (R_{i,i} - R_{T,i}) \tag{2.4}$$

where $R \in \mathbb{R}^{TxT}$ is a test classification matrix, with $R_{i,j}$ representing the test accuracy in task t_j after completely learning t_i. The details are given by Lopez-Paz and Ranzato [45]. A desired model presents a high ACC, BWT, and FWT while showcasing a low FGT metric.

To develop more effective benchmarks in the area, van de Ven and Tolias [83] have suggested three separate CL situations that increase in complexity. The distinction between these situations is based on whether task identity is given or needs to be deduced during the testing phase.

- **Task-incremental learning (TIL)**: In the most basic continual learning situation, the learning agent is consistently notified of the task it should execute. Due to the availability of task identity information, models with task-specific components can be trained in this scenario. The *multi-headed output layer* is a common network architecture used in this scenario, with each task having its own output unit while the remaining network is (possibly) shared across tasks.
- **Domain-incremental learning (DIL)**: The task identity cannot be accessed. Nevertheless, the learning agent is only responsible for solving the domain assigned to it, without the need to deduce what the task is. This scenario is commonly disregarded in literature since it mainly serves as a transitional phase between the other two scenarios.
- **Class-incremental learning (CIL)**: The learning mechanism should possess the capability to tackle every encountered task and deduce the nature of the task presented. This is an arduous situation and resembles the real-world problem of gradually gaining knowledge about new categories of entities.

The continual learning domain proposes various approaches to tackle catastrophic forgetting, especially in the class-incremental learning setting. Regularization techniques limit perturbations of essential parameters for inference of previous tasks during the learning of new tasks [84]. Knowledge distillation methods jointly optimize knowledge from previous and incoming tasks [4]. Memory replay, which re-uses crucial knowledge from past experiences for faster training, is also an effective approach, inspired by reinforcement learning [85].

The current techniques for addressing the challenge of continuous learning can be broadly categorized into three distinct groups:

- **Structure-based approach**: The phenomenon of catastrophic forgetting can arise when adjustments are made to a neural network's parameters to

accommodate new tasks, leading to the loss of knowledge associated with previous tasks. To counter this issue, the network's internal model can be modified by incorporating new layers or altering connections, or by focusing optimization efforts on specific sections of the network for new tasks, while keeping the older parameters intact. Some key methods include:

  - Progressive Neural Networks (PNN) [55]: Adds new columns (layers and connections) to the network's structure for each new task, while freezing the parameters of the previous tasks. This approach maintains separate paths for each task, thereby retaining the learned information from previous tasks.
  - Context-dependent Gating (XdG) [86]: Uses a gating mechanism to activate or deactivate specific parts of the network based on the current task. This enables the network to learn and adapt to new tasks while preserving previously learned knowledge.

- **Regularization-based approach**: When the knowledge necessary for a task is available only during the training phase, it remains feasible to train distinct portions of the network for each task, while still employing the entire network for inference. Conventional methods in this category estimate the importance of the network parameters for tasks learned earlier, penalizing subsequent adjustments in proportion to their significance. Key methods include:

  - Elastic Weight Consolidation (EWC) [87] and its online variant (EWCo) [88]: Employ the Fisher information matrix for evaluating the significance of individual parameters, and subsequently impose regularization penalties to safeguard the crucial parameters during the acquisition of new skills.
  - Synaptic Intelligence (SI) [89]: Measures the importance of each parameter by tracking the cumulative gradient throughout learning. The method then penalizes future modifications to important parameters, thus protecting previously learned knowledge.

- **Memory-based approach**: This strategy involves revisiting previously stored samples in memory when learning new tasks, allowing the model to retain knowledge from prior tasks. Key methods include:

- Learning without Forgetting (LWF) [4]: Retains a model trained on previous tasks to provide soft targets for current task samples, creating a blended training process.
- Gradient Episodic Memory (GEM) [45] and Averaged GEM (A-GEM) [81]: Store a subset of samples from previous tasks in memory and use them to estimate and limit forgetting during training on new tasks.

As an alternative, generative models are used to synthesize samples for rehearsal, thus mitigating catastrophic forgetting. Key methods include:

- Deep Generative Replay (DGR) [85]: Trains a separate generative model sequentially on all tasks to generate samples resembling the data distribution of past tasks. This allows the network to rehearse previous tasks while learning new ones.
- DGR with Knowledge Distillation (DGR+distill) [83]: Combines DGR with knowledge distillation to pair generated samples with soft target knowledge - also known as the logits from the model before learning current knowledge - further improving the model's ability to retain past information.

2.1.2.1 Catastrophic forgetting

Grasping the concept of catastrophic forgetting necessitates an appreciation of the delicate interplay between stability and plasticity in neural networks [90, 91]. The perfect manifestation generated by a neural network ought to be versatile enough to accommodate novel insights and adapt to shifting circumstances, while concurrently maintaining sufficient stability to preserve previously acquired data. Both attributes are undeniably essential, yet they demand distinct considerations.

On one hand, stability calls for the maintenance of the knowledge representation framework, ensuring that past learnings remain intact. On the other hand, plasticity entails modifying this very structure to accommodate new information and adapt to evolving situations. Regretfully, striking the right balance between these two competing aspects proves to be a formidable task.

Neglecting stability and plasticity in neural networks leads to catastrophic forgetting, which occurs when newly introduced information disrupts or overwrites

previously learned knowledge. This phenomenon is comparable to a person moving from one city to another and forgetting about their past city [91]. However, human memory does not typically exhibit such cognitive limitations. Once a person learns something, they can recall it without forgetting it, despite learning new information over time.

The issue of catastrophic forgetting has historically been linked to fixed systems or traditional non-continuous learning, which currently constitute the bulk of popular and implemented neural networks. Conversely, continuous learning networks need to be adaptable in some manner by employing a learning process that allows for the expansion or reduction of units and connections in the network, in accordance with learning demands. In other words, new units can be incorporated to encode recently gained knowledge without interfering with pre-existing ones [3, 4, 84, 86, 92–94].

Catastrophic forgetting is not always a consequence of introducing new information to the network. For instance, when incoming data points are simply additional examples of a pattern already established by the existing knowledge base, the forgetting effect is minimal. As a result, strategies to mitigate catastrophic forgetting are not necessary in such situations.

Finally, remnants of prior knowledge can occasionally persist in the neural network even after catastrophic forgetting has taken place. As a result, the disrupted knowledge can be reacquired faster than it was initially learned. The rate at which a network relearns previously known information is sometimes employed as an indicator to evaluate the effectiveness of suggested approaches to lessen catastrophic forgetting [95, 96].

2.2 Transfer Learning

Transfer learning has emerged as a prominent area of inquiry within the disciplines of machine learning and data mining. This technique generally encompasses a pair of domains: the so-called *source domain* and the *target domain*. Although it is possible for multiple source domains to coexist, the preponderance of current research focuses on utilizing just one. Commonly, the origin field is recognized by having a substantial amount of marked instructional information, while the target field could possess a limited or nonexistent quantity of such data.

Transfer learning is usually discussed in the context of an offline environment. In this context, a model that was initially created to perform a specific task is reutilized to enhance the learning process for a distinct and separate task [20]. This approach has garnered significant attention and interest in recent years due to its potential to streamline the learning process and improve outcomes.

Neural networks, known for their ability to learn high-level features, have also found applications within the domain of transfer learning [97]. These networks possess the power to extract and identify complex patterns from data, which can then be applied to facilitate the learning process in the target domain. This is particularly valuable in cases where the target domain suffers from a scarcity of labeled training data.

To sum up, transfer learning is an essential and expanding field of investigation within the domain of machine learning and data mining, where neural networks have a pivotal function in the operation. By leveraging models and knowledge from source domains, transfer learning can enhance the learning outcomes in target domains, even when faced with limited training data.

Definition 2.3 (Transfer Learning)**.** Given a source domain $\mathcal{D}_S$ and a target domain $\mathcal{D}_T$, the objective of *transfer learning* is to enhance the learning of the target prediction function $f_T(\,\cdot\,)$ within $\mathcal{D}_T$ by leveraging the knowledge contained in $\mathcal{D}_S$.

The topic of feature transferability within the layers of a deep neural network has been previously explored by Chen et al. [98]. Currently, rather than relying on conventional raw inputs as characteristics that may not be versatile enough across various domains, the majority of techniques opt to synchronize the low-dimensional features of distinct domains [99]. In particular, denoising auto-encoders [100] and stacked denoising auto-encoder [101] are frequently employed for this purpose.

Usually, an auto-encoder is composed of two key elements: an encoder function $h(\,\cdot\,)$ and a decoder function $g(\,\cdot\,)$. The initial input x can be recovered as $\hat{x} = g(h(x))$. The objective of auto-encoder training is to decrease the reconstruction error loss, which is typically measured using a mean-squared error function $\mathcal{L}_{\text{mse}}(x, \hat{x})$. After being arranged in a hierarchy, auto-encoders may undergo training. When denoising is performed, the original input vector x is subjected to a random transformation that results in a different vector, $\tilde{x}$, which could involve introducing Gaussian

noise. The objective is to minimize the loss of the denoising reconstruction error, represented by $\mathcal{L}_{\text{mse}}(x, \hat{x})$, where $\hat{x}$ is given by $g(h(\tilde{x}))$ in this particular case.

Conventionally, transfer learning differs from lifelong learning in the following respects:

- The field of transfer learning does not focus on accumulating knowledge; thus, standard solutions do not preserve the transferred knowledge from the source domain to the target domain for subsequent learning utilization. In contrast, knowledge retention and accumulation are crucial for lifelong learning systems, allowing them to acquire additional knowledge more rapidly and accurately.
- Transfer learning is, by tradition, one-way. Given the dearth of training data in the target domain, the primary focus is to transfer knowledge from the source domain to facilitate or amplify the former. If necessary, lifelong learning agents apply newly gained knowledge to refine previously learned information.
- In general, transfer learning involves a source domain and a target domain, although there could be instances where there are multiple domains for each. The underlying assumption is that the source domain shares similarities with the target domain, otherwise, there may be negative transfer leading to adverse outcomes. Lifelong learning algorithms contemplate a vast - theoretically infinite - number of tasks and domains. If the newly arriving domain is dissimilar to prior knowledge, a lifelong learning system must learn it without disrupting the existing knowledge base. Nevertheless, since lifelong learning generally involves numerous past domains, new learning tasks are likely to find some aspects of previous knowledge helpful.

This study aims to blur the boundary separating lifelong learning from transfer learning, rendering one intrinsically vital to accomplish the other.

2.2.1 Domain Adaptation

There are various sub-fields of transfer learning, but our primary focus is on domain adaptation (DA) [102], which is a type of *transductive transfer learning* [20]. DA

necessitates that the source and target tasks are identical, although the domains may differ.

Numerous sub-topics exist within the scope of transfer learning, yet our primary focus is on domain adaptation (DA) [102], a particular kind of *transductive transfer learning* [20]. This approach necessitates that the source and target tasks remain identical, even though the domains may vary.

Definition 2.4 (Domain Adaptation)**.** Given a source domain $\mathcal{D}_S$ and an associated learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and an associated learning task $\mathcal{T}_T$, *domain adaptation* seeks to enhance the learning of the target prediction function $f_T(\cdot)$ within $\mathcal{D}_T$ by leveraging the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$. X_S, X_T, Y_S, and Y_T can all be observed.

This setup can be further broken down into two distinct cases: (1) the source and target domains exhibit dissimilar feature spaces, represented as $X_S \neq X_T$, and (2) the feature spaces of both domains are the same, denoted by $X_S = X_T$; however, the probability distributions at the margins of the input data differ, expressed as $P(X_S) \neq P(X_T)$.

For all domain adaptation problems, the discriminator's performance in the target domain is inherently limited by the error it exhibits in the source domain and the extent of deviation between source and target distributions. To address this limitation, a variety of approaches have been proposed to assess and reduce the differences between the distributions across domains. Some notable examples of these techniques comprise Maximum Mean Discrepancy (MMD) [103], Kullback-Leibler divergence (KL) [104], and domain-adversarial networks [1].

Theorem 2.1. *Theorem 1 (Ben-David et al. [102]): Given two domain distributions $\mathcal{D}_{\mathrm{S}}(X_{\mathrm{S}})$ and $\mathcal{D}_{\mathrm{T}}(X_{\mathrm{T}})$, the target domain error ε_{T} is bound by:*

$$d_{\mathcal{H}\Delta\mathcal{H}}(X_{\mathrm{S}}, X_{\mathrm{T}}) = 2 \underset{\eta \in \mathcal{H}}{sup} \left| P[\eta(X_{\mathrm{S}}) = 1] - P[\eta(X_{\mathrm{T}}) = 1] \right| \tag{2.5}$$

$$\varepsilon_{\mathrm{T}} \leq \varepsilon_{\mathrm{S}} + d_{\mathcal{H}\Delta\mathcal{H}}(X_{\mathrm{S}}, X_{\mathrm{T}}) + C^* \tag{2.6}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}(X_{\mathrm{S}}, X_{\mathrm{T}})$ represents the $\mathcal{H}\Delta\mathcal{H}$ divergence, which depends on the ability of the hypothesis class $\mathcal{H}$ to differentiate between samples generated by $\mathcal{D}_S(X_S)$ and

those generated by $\mathcal{D}_T(X_T)$. C^* denotes the error of an optimal classifier for both the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$.

DA can be classified into three distinct categories in the literature based on the availability of target data labels: *supervised domain adaptation* (SDA) [105], *semi-supervised domain adaptation* (SSDA) [106], and *unsupervised domain adaptation* (UDA) [107–110]. A representation that facilitates cross-domain transfer is considered effective when an algorithm is incapable of identifying the source domain of the input sample [102].

Although the definition of SDA remains consistent with Definition 2.4, SSDA can be viewed as an expanded version where a portion of unlabeled target domain data is available during training. Conversely, for UDA, the predicted labels are considered hidden variables, like clusters within lower-dimensional spaces.

Definition 2.5 (Semi-Supervised Domain Adaptation)**.** Given a source domain $\mathcal{D}_S$ and an associated learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and an associated learning task $\mathcal{T}_T$, *domain adaptation* seeks to enhance the learning of the target prediction function $f_T(\cdot)$ within $\mathcal{D}_T$ by leveraging the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$. X_S, X_T, and Y_S can be observed, while Y_T is partially known.

Definition 2.6 (Unsupervised Domain Adaptation)**.** Given a source domain $\mathcal{D}_S$ and an associated learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a corresponding task $\mathcal{T}_T$, *unsupervised domain adaptation* seeks to enhance the learning of the target function $f_T(\cdot)$ within $\mathcal{D}_T$ by leveraging the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$. X_S, X_T, Y_S can be observed, while Y_T remains unknown.

Advancing further, DA techniques can be classified into two sub-groups: Domain-level DA creates a domain-agnostic latent space [99, 111, 112], while category-level DA establishes a feature-aligned network between the two domains [113, 114]. It is typical for category-level DA approaches to outperform domain-level ones, as the resulting domain-agnostic latent space positions the category distributions of both domains in close proximity.

2.2.2 Multistream transfer learning

The multi-stream domain has recently garnered increasing attention from researchers. Data streams typically originate from non-stationary distributions and are prone to concept drifts [44]. Moreover, when handling dual or multiple streams from distinct domains, *covariate shift* [115], *disparate feature spaces* [116], and *varying throughput* can also be anticipated. The area of multi-data stream mining has made substantial progress in numerous directions, such as regression [57, 58], clustering [59], and classification [60–62].

In the domain of classification, ensemble-based methods have received much attention. One popular technique is *multistream classification* (MSC) [60], which leverages SVM [46] as its primary classifier. To actively recognize concept drifts, MSC employs *kernel mean matching* (KMM) [52] to adapt the model based on the changes in the input distribution. KMM identifies drifts by matching the kernel mean of the source and target distributions. By doing so, it helps to overcome the problem of covariate shift, which occurs when the source and target distributions have different marginal distributions. Another method, *multistream classification with relative density ratio estimation* (MSCRDR) [62], also uses SVM as its main classifier but introduces a unique approach of relative density ratio estimation for drift detection and adaptation. *Efficient multistream classification using direct density ratio estimation* (FUSION) [61] is another ensemble-based technique that utilizes SVM for classification. Additionally, FUSION adopts the drift detection method (DDM) [27] to actively identify and respond to drifts.

Furthermore, in non-stationary environments, transfer learning has been proposed as a viable approach to handle concept drift. Two prominent transfer learning methods for non-stationary environments are *multi-source transfer learning for non-stationary environments* (Melanie) [63] and *multi-source mapping transfer learning for non-stationary environments* (MARLINE) [64], both of which utilize random forests. Melanie employs the drift detection method (DDM) to recognize and adapt to drifts in the input distribution. In contrast, MARLINE utilizes a drift detection method based on Hoeffding's inequality (HDDM) [53] to detect and respond to drifts. While SVM and random forests are commonly used in multi-stream classification, compared to neural networks, they are generally considered less effective in learning high-level features from the data.

Other notable methods for addressing concept drift in multi-stream environments include *autonomous transfer learning* (ATL) [51], *online transfer learning* (OTL) [117], and *automatic online multi-source domain adaptation* (AOMSDA) [118]. ATL and AOMSDA employ a self-evolving neural network structure with a feature extractor and discriminator, driven by an active drift detector to detect changes in the data distribution. In contrast, OTL utilizes a passive drift detector and continuously adapts the model to potential distribution changes. While ATL and AOMSDA align the source and target distributions within the feature extractor using the Kullback-Leibler divergence (KL), their discriminators remain unaware of the target distribution. On the other hand, OTL updates its weight continuously, utilizing information from both source and target streams, to construct an intermediate domain-invariant representation of their distributions. These methods have demonstrated effectiveness in handling concept drift in multi-stream environments – although still single domain – and offer valuable insights into designing and applying machine learning models that can adapt to evolving data distributions.

All the aforementioned methods are single-domain solutions, meaning they can manage multiple streams but were not specifically designed to address multiple domains.

Ultimately, examples of transfer learning methods that consider multiple data streams and domains include *multistream domain adaptation* (MSDA) [66] and *multistream cross-domain adaptation* (COMC) [67]. These UDA solutions rely on a warm-up initialization stage and utilize support vector machines (SVM) as their principal classifiers. Additionally, they feature active drift detection methodologies.

In this research, multistream transfer learning is also called online unsupervised cross-domain adaptation, and its problem definition follows.

Definition 2.7 (Online Unsupervised Cross-Domain Adaptation)**.** Consider the set $\{(X_S^{(i)}, Y_S^{(i)})\}_{i=1}^{n_S}$ as a set of labeled instances with size n_S originating from a non-stationary source domain stream $\mathcal{D}_S$. In the same manner, let $\{(X_T^{(i)})\}_{i=1}^{n_T}$ indicate a set of unlabeled instances with size n_T from a separate non-stationary target domain stream $\mathcal{D}_T$, showcasing *a scarcity of labeled instances*. In cases where $X_S \neq X_T$, the domains display *varying feature spaces*, requiring a transformation to achieve a *distinct marginal probability distribution*, meaning $Y_S = Y_T$, but $P_S(X) \neq P_T(X)$ and $P_S(Y|X) = P_T(Y|X)$.

The aim is to develop a classifier that leverages $X_S \in \mathbb{R}^{n_S \times u}$, $Y_S \in \mathbb{R}^{n_S \times m}$, and $X_T \in \mathbb{R}^{n_T \times u}$ to forecast the class label $\hat{Y}_T \in \mathbb{R}^{n_T \times m}$ for X_T.

Both the source and target distributions emerge from separate non-stationary processes with differing speeds, leading to *contrasting throughput*. They may change over time due to *covariate shift*, for example, $P_S(X,Y)_{t_S} \neq P_S(X,Y)_{t_S+1}$ and $P_T(X,Y)_{t_T} \neq P_T(X,Y)_{t_T+1}$, where t_S and t_T are the timestamps for incoming samples from the source and target, respectively. Finally, *asynchronous drift* happens when $t_S \neq t_T$ during the previously mentioned covariate shift, resulting in modifications in the source and target domains at different timestamps.

Altogether, five challenges must be addressed in online unsupervised cross-domain adaptation: *Scarcity of labeled samples*, *Different feature space*, *Covariate shift*, *Asynchronous drift*, and *Contrasting throughput*.

2.2.3 Cross-Domain Continual Learning

Unlike multistream transfer learning, which happens in the sequential learning environment, cross-domain continual learning happens in an incremental setting. Its definition can be adapted from the previous Definition 2.7, yielding:

Definition 2.8 (Cross-Domain Continual Learning)**.** Let $(\mathcal{D}_{\mathrm{S}_i}, \mathcal{D}_{\mathrm{T}_i})$ be an incremental flow, with the data tuples $(x_{\mathrm{S}_i}, y_{\mathrm{S}_i})$ satisfying $(x_{\mathrm{S}_i}, y_{\mathrm{S}_i}) \overset{iid}{\sim} \mathcal{D}_{\mathrm{S}_i}(X_{\mathrm{S}}, Y_{\mathrm{S}})$, and (x_{T_i}) satisfying $(\mathbf{x}_{\mathrm{T}_i}) \overset{iid}{\sim} \mathcal{D}_{\mathrm{T}_i}(X_{\mathrm{T}})$. The data tuples contain a labeled set X_{S_i} paired with its target set Y_{S_i}, along with an unlabeled input set X_{T_i}, organized into sequential tasks $t_i \in \mathcal{T} = \{1, \cdots, T\}$, where the total number of tasks T is unknown a priori. The goal is to learn a predictor $f : (X_{\mathrm{S}} \cup X_{\mathrm{T}}) \times \mathcal{T} \rightarrow Y_{\mathrm{T}}$, which can be queried *at any time* to predict the target vector y_{T} associated to a sample in the target domain x_{T}, where $(x_{\mathrm{T}}, y_{\mathrm{T}}) \sim P_t$.

The solution to this problem is not trivial, as the model needs to handle *label scarcity in the target domain*, and also possibles *task drifts* $P_i(X_{\mathrm{S}}, Y_{\mathrm{S}}) \neq P_{i+1}(X_{\mathrm{S}}, Y_{\mathrm{S}}) \wedge P_i(X_{\mathrm{T}}, Y_{\mathrm{T}}) \neq P_{i+1}(X_{\mathrm{T}}, Y_{\mathrm{T}})$ and *domain drifts* $P_t(X_{\mathrm{S}}) \neq P_t(X_{\mathrm{T}}) \wedge P_t(Y_{\mathrm{S}}|X_{\mathrm{S}}) = P_t(Y_{\mathrm{T}}|X_{\mathrm{T}})$ along the training.

Consider an incremental data flow represented by $(\mathcal{D}_{\mathrm{S}_i}, \mathcal{D}_{\mathrm{T}_i})$. Here, the data tuples $(x_{\mathrm{S}_i}, y_{\mathrm{S}_i})$ follow $(x_{\mathrm{S}_i}, y_{\mathrm{S}_i}) \overset{iid}{\sim} \mathcal{D}_{\mathrm{S}_i}(X_{\mathrm{S}}, Y_{\mathrm{S}})$, and (x_{T_i}) adhere to $(X_{\mathrm{T}_i}) \overset{iid}{\sim} \mathcal{D}_{\mathrm{T}_i}(X_T)$.

The tuples contain a labeled set X_{S_i} associated with a label set YSi, and an unlabeled set X_{T_i}. These sets are arranged into sequential tasks $t_i \in \mathcal{T} = 1, \cdots, T$, where the overall number of tasks T is not known in advance. The aim is to learn a predictor $f : (X_S \cup X_T) \times \mathcal{T} \rightarrow Y_T$ that can be used *at any moment* to predict the target vector y_{T_t} for a sample in the target domain x_{T_t}, with $(x_{T_t,T_t}) \sim P_t$.

The solution to this problem is not trivial, as the model has to manage the *scarcity of labels in the target domain* and potential *task drifts* $P_i(X_S, Y_s) \neq P_{i+1}(X_s, Y_s) \wedge P_i(X_t, Y_t) \neq P_{i+1}(X_t, Y_t)$, as well as *domain drifts* $P_t(X_s) \neq P_t(X_t) \wedge P_t(Y_s|X_s) = P_t(Y_t|X_t)$ during the training process.

Cross-domain continual learning is also referred to as continual UDA in this research.

2.2.4 Knowledge Distillation

The technique known as knowledge distillation (KD) [119] facilitates the transfer of knowledge from one model to another by means of compression. The underlying principle involves a larger teacher model overseeing the training of a smaller student model, which is then trained to replicate the actions of the teacher model. KD boasts a notable advantage in its ability to handle varied structures, allowing for the teacher and student models to have different network structures. The teacher model supervises the training of the student by providing it with its output, referred to as the soft target.

In essence, KD works by minimizing the discrepancy between the output $\hat{z}$ of the student model and the logits z that the teacher model generates, given a specific input sample. The objective function for KD can be defined as:

$$\mathcal{L}_{KD} = \mathcal{L}_{mse}(z, \hat{z}) = ||z - \hat{z}||. \tag{2.7}$$

Although KD has become a popular research topic in the machine learning community, many existing approaches still follow a single teacher-student relationship [119]. Recent studies have explored the use of multiple teachers, where each teacher contributes a different aspect of the knowledge to the student model. This approach

is referred to as ensemble knowledge distillation and has shown promising results in improving the performance of the student model [120].

Moreover, there are also techniques that use self-distillation, where a single model acts as both teacher and student, learning from its own previous iterations [120]. In this case, the objective is to compress the knowledge acquired by the model during training into a smaller and more efficient model.

Overall, KD is a powerful technique for knowledge transfer that can enhance the performance of smaller models. The ability to handle heterogeneous structures, as well as the use of multiple teachers, opens up new possibilities for improving the performance of models in various applications.

2.2.5 Knowledge Amalgamation

Knowledge Amalgamation (KA) [121–123] refers to a technique for creating a single student model that can address the inclusive collective goal of various teacher models. KA is an evolution of Knowledge Distillation (KD) that allows for an arbitrary number of teacher models to be combined, each of which implements a specific task.

We define knowledge amalgamation in terms of an incremental task-learning scenario as follows:

Definition 2.9 (Knowledge Amalgamation)**.** Suppose there are N teacher models $t_{i=1}^{N}$ that have been trained to perform specific tasks T_i. Let $\mathcal{D}_i$ represent the set of classes that each teacher model t_i can handle. Without sacrificing generality, it is assumed that $\mathcal{D}_i \neq \mathcal{D}_j$ for $\forall i \neq j$, meaning that each pair of teacher models handles different tasks. The main objective of knowledge amalgamation is to create a compact single-head student model that can simultaneously classify all classes in $\mathcal{D} = \cup_{i=1}^{N} \mathcal{D}_i$. By doing so, the knowledge extracted from each teacher model is amalgamated into a single student model, which is able to handle various tasks.

The post-processing mechanism of knowledge amalgamation offers increased flexibility compared to current continual learning techniques since each teacher model can be separately developed for a particular task. This allows their knowledge to

be merged into a student model without sacrificing generalization ability. Essentially, KA consolidates insights from numerous teachers to generate a single student model that outperforms any individual teacher.

2.2.6 Pseudo-labeling

Pseudo-labeling [124, 125] has emerged as a recent driver for more accurate UDA procedures. The basic idea behind pseudo-labeling is to generate synthetic labels for unlabeled samples, which can then be added to the training set. By doing this, confident unlabeled data can be used to augment the limited labeled data available for the target domain.

To perform pseudo-labeling for domain adaptation, the source domain data is usually used as a pseudo-label generator for the target domain data. Specifically, a pre-trained classifier on the source domain data or a similar dataset, such as ImageNet [126], is used to generate pseudo-labels for the target domain data. This process often involves several steps, such as feature separation [127], regularization [128], fine-tuning [129], and more.

While pseudo-labeling can be effective in improving the performance of domain adaptation models, it can also introduce noise into the training data. This noise can be caused by miss-synthetic labeling, which can harm the model's predictive performance. As a result, several approaches have been proposed to mitigate the effects of noisy labeling. One approach is to explicitly re-evaluate and correct wrong labels [130]. Another approach is to focus on better-unlabeled sample selection via models' confidence, which can help filter out samples with potentially incorrect labels [131, 132]. Finally, some works propose objective functions that are tolerant to labeling noise, allowing the model to learn from noisy training data while still maintaining good performance [133].

Overall, pseudo-labeling has shown promise as a technique for improving UDA, but careful consideration must be given to the potential for noise in the synthetic labels.

2.3 Preview of the work

This report contain three main contributions to the area of lifelong learning, detailed in the following chapters. Although each one uses transfer learning techniques, they are used into three unconnected different ideas that ties together the area's landscape for a future common step forward.

At chapter 3, the autonomous cross-domain conversation (ACDC) framework is presented, where transfer learning - mainly unsupervised domain adaptation - is put forward to put deep neural network as probable solutions for the online unsupervised cross-domain adaptation problem. In other words, this work is able to perform real-time unsupervised domain adaptation in an non-stationary environment, adapting the structure of the network for incoming concept drifts. Contrary to other previous solutions that rely heavely on random forests and assembles, ACDC brings forward deep neural networks as its main solver.

At chapter 4, the catastrophic forgetting solution via knowlege amalgamation (CFA) is put forward, showcasing a novel transfer learning method - knowledge amalgamation - as a possible solution for the class-incremental learning. Here, transfer learning is used in the continual learning scenario to accumulate and adapt to new incoming knowledge. Until the moment, memory-based approaches have the best performance in terms of average accuracy among all continual learning models. CFA exploit such fact to record not only previous samples, but also previous specialist models - called teachers - which are used to amalgamete their knowledge into a common student model. Different from the ACDC problem, CFA is not so worried about a real-time learning, being able to expend more computation power in a solution that aggregates knowledge from different classes into a single multi-head model.

Finally, at chapter 5, cross-domain continual learning (CDCL) is introduced, showcasing logic connections to ACDC. Here, transfer learning is also used to learn multiple domains together in an unsupervised manner; However, instead of being worried about the problem of concept drifts generated via non-stationary real-time distributions, CDCL handles the problem of catastrophic forgetting in a scenario where incoming new knowledge is presented in a known time interval. CDCL is, until the moment, the only unsupervised cross-domain continual learning model, capable or learning multiple incremental domains simultaneously.

This report is organized as follows:

All three contributions are pieces of a bigger puzzle, discussed at chapter 6.

Chapter 3

Online Unsupervised Cross-Domain Adaptation

The task of online unsupervised cross-domain adaptation is highly complex and challenging, as it necessitates learning from two distinct but related data streams with varying feature spaces. In this setting, the labeled source stream and the unlabeled target stream exhibit distinct traits and encounter challenges, such as a *shift in covariate distribution*, *asynchronous concept shifts*, and *divergent data processing rates*. To overcome these challenges, we propose ACDC, a novel adversarial unsupervised domain adaptation framework that leverages a self-evolving neural network structure.

ACDC comprises three modules that are integrated into a single model. The first module is a denoising autoencoder that extracts features from data, while the second module is an adversarial component that performs domain conversion. The third and final module is a discriminator that learns the source stream and predicts the target stream. ACDC is highly flexible and expandable, requiring minimal hyperparameter tuning. Our experimental results, based on the prequential test-then-train protocol, demonstrate that ACDC outperforms baseline approaches in terms of target accuracy. In some cases, ACDC improves target accuracy by more than 10%, providing a novel approach to address the challenges of online unsupervised cross-domain adaptation by leveraging both labeled and unlabeled data

from multiple sources to enhance predictive models' performance. Source-code: `github.com/Ivsucram/ACDC`[1]

3.1 Introduction

In numerous data stream domains, including but not limited to classification, regression, and clustering, data mining and machine learning methods have accomplished remarkable outcomes [27]. These techniques have allowed us to extract valuable insights from vast amounts of data, enabling a wide range of applications, from recommendation systems to fraud detection and personalized medicine. Nonetheless, several incremental learning approaches presuppose that the training and test data originate from identical probability distributions [20]. In other words, they assume that the data generating process does not change over time. In practical settings, this assumption is frequently contravened, as data streams may exhibit dynamism and non-stationarity. For example, a model trained to classify spam emails might become less accurate over time as spammers adapt their tactics. As the distribution undergoes changes, the majority of statistical models necessitate reconstruction using fresh training data that has been gathered [44]. This process can be expensive or even impossible in some applications, especially when the data is scarce, costly, or difficult to collect. Moreover, rebuilding the model can cause a delay in the decision-making process, which can be unacceptable in time-critical applications, such as fraud detection or autonomous systems.

In numerous instances of continuous operations, utilizing unsupervised cross-domain adaptation online can prove to be extremely advantageous. For instance, the Internet of Things (IoT) has led to an explosion of data streams, posing a significant challenge to the field of streaming mining. Initially, a forecasting model can be constructed using only one occurrence of an application, in which an oracle or an algorithm that operates offline delivers input on the accurate classification labels. However, duplicating this procedure for every fresh occurrence can consume a lot of time and money, thereby rendering it more feasible to automatically harmonize the distribution between an already labeled source stream and a new unlabeled target stream. This real-world scenario presents five significant challenges when introducing a new stream model:

[1] Paper published in the Knowledge-based Systems Journal, 2022

1. The first challenge is the *scarcity of labeled samples* in the new unlabeled stream [134]. In many cases, labeling the data in the new stream is difficult or even impossible, making it challenging to apply supervised learning techniques.

2. The second challenge is the *difference in the feature space* or *marginal probability distribution* between the initial predictive model and the new stream [116]. The new stream may contain features that were not present in the original model or features that have a different distribution, making it necessary to adapt the model to the new feature space.

3. The third challenge is the *covariate shift*, which occurs when the data are drawn from a different distribution than the one used to train the initial model [115]. This shift can lead to a decrease in the accuracy of the model and must be addressed to ensure the model's performance in the new stream.

4. The fourth challenge is the *asynchronous drift*, which occurs when both streams undergo independent drifts over time [44]. The model must adapt to these changes and update its parameters accordingly to maintain its accuracy.

5. The fifth challenge is the *contrasting throughput*, which happens when the two streams generate samples at different speeds. This difference can lead to an imbalance in the amount of data from each stream and must be addressed to ensure that the model remains unbiased.

The *prequential test-then-train* approach poses an additional challenge for online transfer learning models, requiring accuracy evaluation on a test set before incorporating new data into the training set [135]. While existing online transfer learning methods have demonstrated the ability to overcome the five challenges mentioned above [51, 60, 61, 136], it is assumed that the origin and destination streams belong to a common domain. Hence, they engage in supervised or semi-supervised domain adaptation to address cases where there is some prior knowledge of the target domain. In contrast, online unsupervised cross-domain adaptation seeks to harmonize the distribution of several related data streams without any initial understanding of the target domain. This task is more challenging but has the potential to address many real-world applications. To date, only a few studies, such as Li et al. [66] and Tao et al. [67], have tackled this issue. Nonetheless, their proposed remedies hinge on the utilization of support vector machines (SVMs),

which possess constraints in managing high-dimensional quandaries in contrast to neural network-oriented methodologies.

The Adversarial Unsupervised Cross-Domain Adaptation (ACDC) is a novel framework that aims to address the challenges of online unsupervised cross-domain adaptation in a multi-stream setting. The system is built on a comprehensive, self-adapting neural network architecture capable of managing diverse data streams, including two types of constantly changing data streams in separate fields with distinct characteristics. In this configuration, a single stream possesses abundant labeled data and is known as the source stream. The other stream, referred to as the target stream, is an independent process that generates unlabeled data. The goal of the ACDC framework is to align the distribution of the two streams and adapt the model to the target stream without any labeled data in the latter.

The ACDC framework comprises three modules: a generator network (DAE), an adversarial domain-adaptation network (DAA), and a discriminator network (DISC). These modules work together to adapt the model to the target domain, which generates unlabeled data. The generator network takes as input the data from the source and target streams and generates a set of latent features that are common to both domains. The adversarial domain-adaptation network takes the generated latent features as input and determines whether they come from the source or target domain. The network further enforces a domain-invariant condition through a gradient reversal layer [1] to make the latent features suitable for both domains. Finally, the discriminator network learns a map between the domain-invariant latent space and the annotated labels received from the source domain, which are expected to behave similarly to the target domain.

All three modules of the ACDC framework have an independent self-evolving structure. This flexible architecture enables each component to respond proactively to asynchronous alterations in both source and destination distributions. The self-evolving structure of the ACDC framework enables it to adapt to new data streams and maintain its performance over time. Specifically, the framework can adjust its architecture and parameters dynamically, depending on the characteristics of the data streams. This adaptation enables the model to solve the five challenges of online unsupervised cross-domain adaptation, namely the *scarcity of labeled samples* in the target stream, *different feature spaces* or *marginal probability distributions*, *covariate shift*, *asynchronous drift*, and *contrasting throughput*.

To be more specific, ACDC deals with each challenge as follows:

- *Scarcity of labeled samples*: Handled by the domain latent invariant space in DAE and the final prediction by DISC. By generating domain-invariant latent features that are suitable for both domains, the ACDC framework can make accurate predictions even in the absence of labeled data in the target stream.

- *Different feature space* or *different marginal probability distribution*: Tackled by DAA, which forces an aligned feature distribution into DAE's latent space via a gradient reversal layer. This technique enables the ACDC framework to learn domain-invariant features that are suitable for both domains, even when the feature spaces or marginal probability distributions are different.

- *Covariate shift*: Resolved mainly by the domain adaptation procedure in DAA. By adapting the model to changes in the data distribution overtime via the DAA module, the ACDC framework can maintain its performance even in the face of covariate shift.

- *Asynchronous drift*: Addressed by all modules' dynamic structures. The self-evolving structure of the ACDC framework enables it to adjust its architecture and parameters dynamically, depending on the characteristics of the data streams. This adaptation empowers the ACDC framework to react actively to changes in the source and target distributions, even in the presence of asynchronous drift.

- *Contrasting throughput*: Unfolded by the ACDC learning flow, which pairs and permutes incoming samples into processing sliding windows. By pairing and permuting incoming samples, the ACDC framework can balance the throughput of the source and target streams and ensure that the model remains unbiased.

The ACDC framework represents a significant contribution to the field of online unsupervised cross-domain adaptation. It highlights the potential of neural networks as a solution to this problem and comprises several innovative features, including:

- **A novel framework**: The ACDC framework is a novel approach to online unsupervised cross-domain adaptation that leverages the power of neural

networks to align the distribution of multiple data streams. This framework represents a significant advance in the field of online learning and has the potential to address many real-world applications.

- **A fully autonomous data-driven structure**: The ACDC framework comprises a fully autonomous data-driven structure that can grow and prune nodes on the three training modules. This structure enables the model to adjust its architecture and parameters dynamically, depending on the characteristics of the data streams. This adaptation empowers the ACDC framework to react actively to changes in the source and target distributions, even in the presence of asynchronous drift.

- **Domain-adversarial bias-variance trade-off**: The ACDC framework employs a domain-adversarial bias-variance trade-off tracker to adapt the discriminator to possible concept drifts. This approach enables the model to maintain its performance over time by balancing the trade-off between overfitting to the source domain and underfitting to the target domain.

- **Domain-adversarial network learning**: The ACDC framework integrates a domain-adversarial network that learns an online unsupervised cross-domain configuration. This network enables the ACDC framework to generate domain-invariant features that are suitable for both domains, even in the absence of labeled data in the target stream.

3.2 ACDC

The ACDC model is an unsupervised adaptation framework designed for online use across multiple domains. It employs adversarial training to extract deep network representations that are invariant across domains. Adversarial UDA methods have been shown to have good target-domain accuracy [137], and ACDC capitalizes on its achievement by utilizing its domain-adversarial adaptation module (DAA) to collect data from both the source and destination streams. The system has the capability to detect slight variations in data distribution and inform its discriminator DISC about potential drifts while simultaneously achieving a domain-agnostic feature extraction on DAE. Additionally, ACDC carries out all of these functions in a

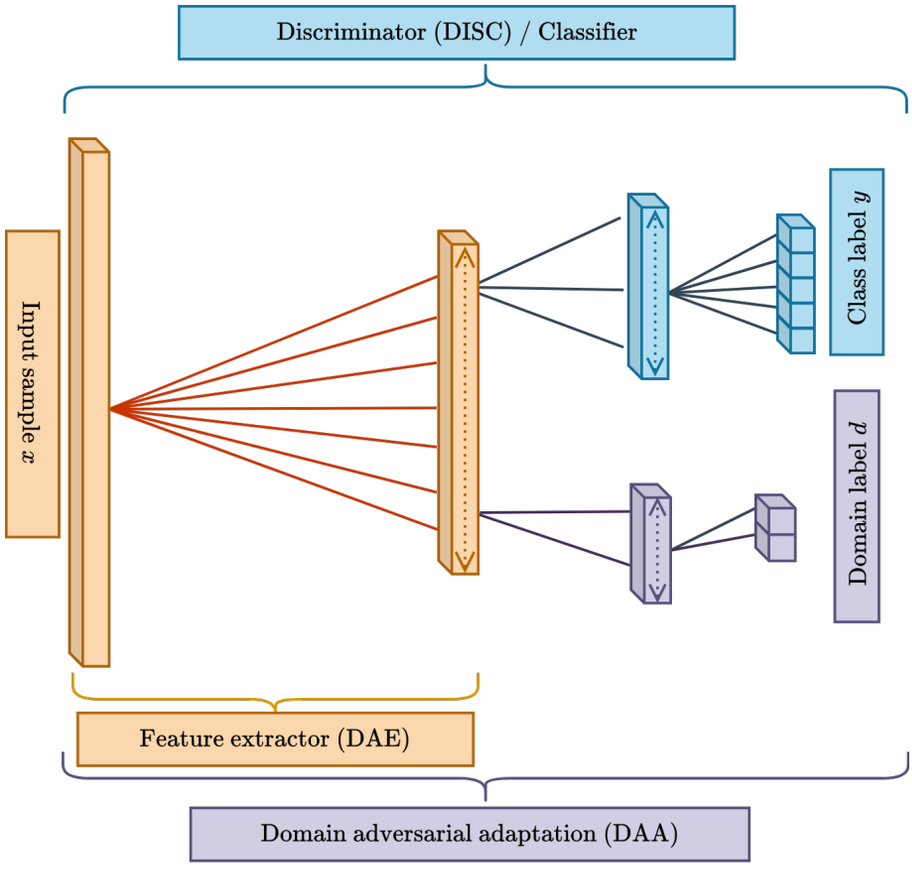


FIGURE 3.1: The proposed ACDC architecture. It includes the modules DISC and DAA, which extend from the latent space of the third module DAE. The architecture allows for the dynamic growth and pruning of the latest hidden layer of each module.

unique adaptive model, allowing it to adjust to fluctuations in the data distribution as they occur.

An overview of ACDC's structure is depicted in Figure 3.1, which illustrates the framework's architecture and information flow. Figure 3.3 provides a detailed view of the information flow within the ACDC framework, while the procedural steps are depicted by Algorithm 1.

The ACDC learning algorithm comprises of three neural network components, namely, feature extraction, domain-adversarial adaptation, and discriminator, all of which encompass two primary stages, that is, *adaptation* and *learning*. During the adaptation phase, the ACDC framework grows and prunes nodes, adjusting its architecture and parameters dynamically based on the characteristics of the data streams. The ACDC framework gains the ability to respond promptly to modifications in the source and target distributions, even when asynchronous drift

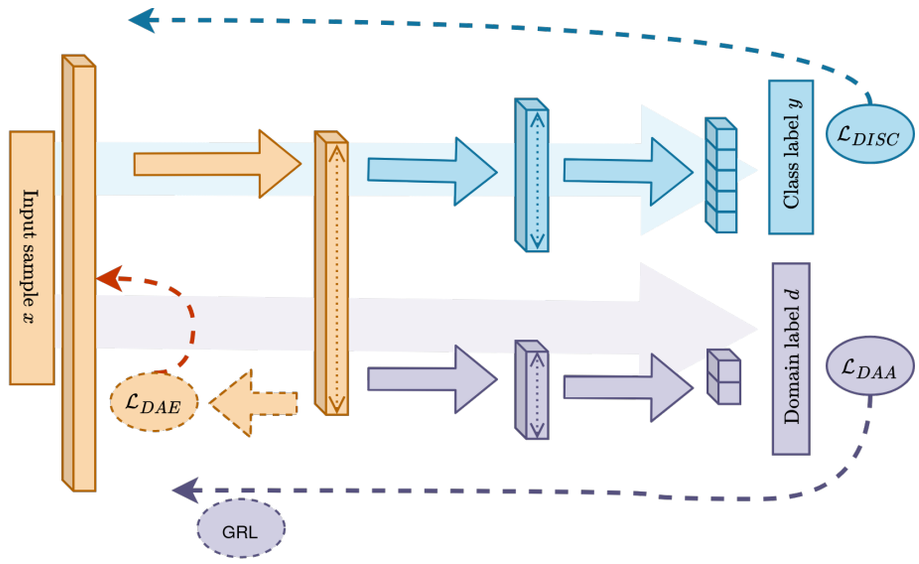


FIGURE 3.2: The suggested ACDC information flow. The solid, blocky arrows signify the feed-forward flow, while the dashed line arrows represent the back-propagation flow. It is noteworthy that DAA conducts a gradient reversal layer (GRL) on DAE, as per Ganin and Lempitsky [1].

is present, through this adaptation. In the learning phase, the ACDC framework utilizes conventional feed-forwarding and back-propagation methods for neural networks to adjust its weights and biases. This process enables the model to learn from the data and improve its accuracy over time.

3.2.1 Parameter learning

The ACDC framework is a set of three loss functions that are both simple and effective in achieving the desired results. The three functions are:

- $\mathcal{L}_{DAE}$, which is a combination of mean-squared error loss between the reconstructed source and target samples $\hat{x}_s$ and $\hat{x}_t$, respectively. This is shown in Equation (3.1).
- $\mathcal{L}_{DAA}$, which is a combination of binary cross-entropy loss between the binary flags d'_s and d'_t, which indicate the origin of the source and target domains, respectively. This is shown in Equation (3.2).
- $\mathcal{L}_{DISC}$, which is the multi-class logarithmic loss between the model prediction $\hat{y}_s$ and the source samples y_s. This is shown in Equation (3.3).

Algorithm 1: ACDC

Input: Fully labeled source stream $\mathcal{D}_S$, unlabeled target stream $\mathcal{D}_T$
Output: Predicted labels for the target stream $\hat{Y}_T$
while *samples exist in both* $\mathcal{D}_S$ *and* $\mathcal{D}_T$ **do**
  Read N_m incoming samples from $\mathcal{D}_S$ and $\mathcal{D}_T$ into W_S and W_T;
  Predict target stream labels by passing W_T through the discriminator module: $W_T \rightarrow \hat{Y}_T$;
  Handle contrasting throughput by pairing and permuting W_S and W_T to W'_S and W'_T;
  \\where κ is the number of internal epochs
  for $i \leftarrow 1$ **to** κ **do**
    foreach x_s, y_s, x_t ***in*** *(*W'_S, W'_T*)* **do**
      if $i == 1$ **then begin** Adaptation: \\Sub-section 3.2.2
        Evaluate DAE on $x_s \rightarrow x_t$, $x_t \rightarrow x_s$;
        Evaluate DAA on $x_s \rightarrow 0$, $x_t \rightarrow 1$;
        Evaluate DISC on $x_s \rightarrow y_s$;
      begin Learning: \\Sub-Section 3.2.1
        Fit DAE on $x_s \rightarrow x_t$, $x_t \rightarrow x_s$ using the loss function in Equation (3.1);
        Fit DAA on $x_s \rightarrow 0$, $x_t \rightarrow 1$ using the loss function in Equation (3.2);
        Fit DISC on $x_s \rightarrow y_s$ using the loss function in Equation (3.3);

$$\mathcal{L}_{DAE} = \mathcal{L}_{mse}(\hat{x}_s, x_t) + \mathcal{L}_{mse}(\hat{x}_t, x_s) \tag{3.1}$$

$$\mathcal{L}_{DAA} = \mathcal{L}_{log}(d'_s, 0) + \mathcal{L}_{log}(d'_t, 1) \tag{3.2}$$

$$\mathcal{L}_{DISC} = \mathcal{L}_{log}(\hat{y}_s, y_s) \tag{3.3}$$

The loss functions for mean-squared error and multi-class logarithmic loss are represented as $\mathcal{L}_{mse}$ and $\mathcal{L}_{log}$, correspondingly.

In order to address the challenge of inconsistent throughput, a permuted paired sliding window is utilized to allocate the samples. The training process involves

implementing a standard stochastic gradient descent (SGD) technique with a specific learning rate λ on sets of source and target samples. The network parameters[2] are updated as shown in Equations (3.4), (3.5), and (3.6).

$$\theta_{dae} \leftarrow \theta_{dae} - \lambda\Big(\frac{\partial \mathcal{L}_{DAE}}{\partial \theta_{dae}} + \frac{\partial \mathcal{L}_{DISC}}{\partial \theta_{dae}} - \frac{\partial \mathcal{L}_{DAA}}{\partial \theta_{dae}}\Big) \tag{3.4}$$

$$\theta_{daa} \leftarrow \theta_{daa} - \lambda\frac{\partial \mathcal{L}_{DAA}}{\partial \theta_{daa}} \tag{3.5}$$

$$\theta_{disc} \leftarrow \theta_{disc} - \lambda\frac{\partial \mathcal{L}_{DISC}}{\partial \theta_{disc}} \tag{3.6}$$

The DAE utilizes a generative loss function that utilizes mean-squared error to tackle the difficulty of disparate feature spaces across the streams. For each input sample pair, a one-epoch greedy-layer-wise pre-training is carried out without noise, as suggested by Bengio et al. [138], followed by conventional tied-weight training with a masking noise rate of 10%.

Conversely, training for DAA and DISC involves the utilization of the multi-class logarithmic loss. In addition, DAA employs a gradient reversal layer (GRL) [1] transformation at θ_{dae}, as shown by Equation (3.4).

3.2.2 Module adaptation

The ACDC relies on data and has the ability to evolve on its own, based on incoming data, that adapts to handle asynchronous drift in the data stream, guided by its active drift detector. The framework uses various methods to analyze and approximate the model's generalization power, as well as detect situations of under-fitting or over-fitting. Figure 3.3 provides an overview of this process.

Initially, the framework scrutinizes the DAE module's reconstruction error and the DAA and DISC modules' discriminative error to detect plausible problems.

[2]For brevity of notation, we will refer to ($\mathcal{W}_{dae}$, b_a, b_b) as θ_{dae}, ($\mathcal{W}_{daa_1}$, $\mathcal{W}_{daa_2}$, b_c, b_d) as θ_{daa}, and ($\mathcal{W}_{disc_1}$, $\mathcal{W}_{disc_2}$, b_e, b_f) as θ_{disc}.

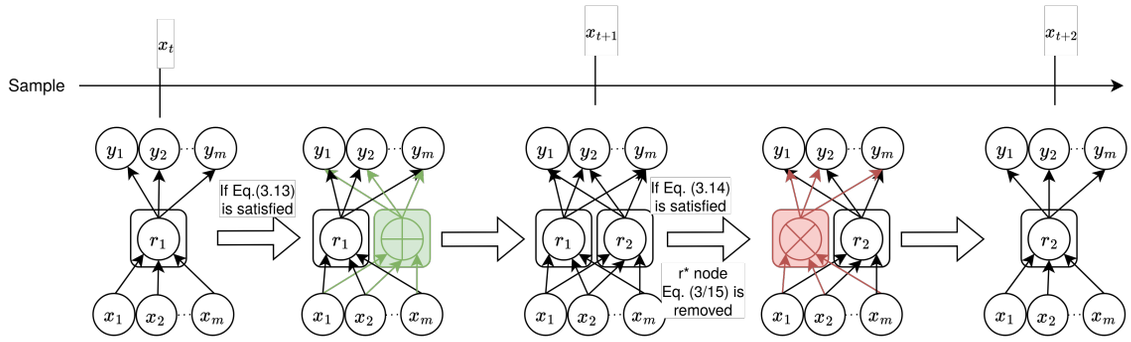


FIGURE 3.3: This figure displays how ACDC's evolution process works. A module hidden node is either created or removed based on specific conditions. The sample x_t, which represents a sample calculated at time t, can originate from either X_S or X_T. If the sample comes from X_S, it is used to determine whether DISC, DAE, and DAA should evolve. If the sample comes from X_T, it is used to evaluate whether DAE and DAA should evolve. DISC cannot assess its own evolution using samples from X_T as it requires access to Y_T, which is currently unknown. Equation (3.10) specifies this requirement. The green block represents a node being created, while the red block represents the least significant node being pruned.

Afterwards, it estimates the model's generalization ability by performing its bias-variance decomposition, indicating the circumstances of high bias (under-fitting) and high variance (over-fitting).

ACDC uses the sigmoid function $\sigma(\cdot)$ for all its activation functions. This makes it easier to approximate the model's generalization by using a probit function $\Phi(\xi\mathbf{X}) = \int_{-\infty}^{\mathbf{X}} \mathcal{N}(\theta|0,1)d\theta$, where $\xi = \pi/8$ [139].

$$E[\hat{y}]_{DAE} = \sigma\left(\frac{\mu}{\sqrt{1+\frac{\pi\sigma^2}{8}}}\mathcal{W}_{dae} + b_a\right)\mathcal{W}_{dae}^T + b_b \tag{3.7}$$

$$E[\hat{y}]_{DAA} = \sigma\left(\sigma\left(\frac{\mu}{\sqrt{1+\frac{\pi\sigma^2}{8}}}\mathcal{W}_{dae} + b_a\right)\mathcal{W}_{daa_1} + b_c\right)\mathcal{W}_{daa_2} + b_d \tag{3.8}$$

$$E[\hat{y}]_{DISC} = \sigma\left(\sigma\left(\frac{\mu}{\sqrt{1+\frac{\pi\sigma^2}{8}}}\mathcal{W}_{dae} + b_a\right)\mathcal{W}_{disc_1} + b_e\right)\mathcal{W}_{disc_2} + b_f \tag{3.9}$$

where μ and σ^2 are respectively the mean and variance of the samples fed, and $\hat{y}$ is the expected output. The integral of the probit function is then used to measure the module's bias and variance:

$$Bias + Var = (E[\hat{y}]^2 + y) + (E[\hat{y}^2] + E[\hat{y}]^2) \tag{3.10}$$

ACDC employs a modified statistical process control (SPC) algorithm to determine the conditions for module growth and pruning [54]:

$$\beta = \alpha_1(-Bias) + \alpha_2 \tag{3.11}$$

$$\Lambda = 2 * (\alpha_1(-Var) + \alpha_2) \tag{3.12}$$

$$\text{Growing condition: } \mu_{Bias} + \sigma_{Bias} \geq \mu_{Bias}^{min} + \beta * \sigma_{Bias}^{min} \tag{3.13}$$

$$\text{Pruning condition: } \mu_{Var} + \sigma_{Var} \geq \mu_{Var}^{min} + \Lambda * \sigma_{Var}^{min} \tag{3.14}$$

The values of β and Λ are carefully chosen to enable flexible growing and pruning conditions for each module in the ACDC architecture. By setting these values, a confidence interval is achieved in the range of $[\mu \pm \sigma, \mu \pm 2\sigma]$ and $[\mu \pm \sigma, \mu \pm 3\sigma]$ for the growing and pruning conditions, respectively. The hyper-parameters α_1 and α_2 control these conditions. This setup is designed to provide a node growing process that is sensitive to the high-bias case, while the node pruning process is reactive to the high-variance situation.

It is worth noting that Λ is set to double β to avoid a scenario known as "direct-pruning-after-growing," which can occur when the pruning condition is triggered immediately after a new node is added to the module. Additionally, the values of μ_{Bias}^{min}, σ_{Bias}^{min}, μ_{Var}^{min}, and σ_{Var}^{min} are re-initialized if the growing or pruning conditions are met.

Every individual module within the ACDC framework has the ability to adjust autonomously, persistently expanding or removing nodes from the final layer prior

to its output. This process is illustrated in the color scheme in Figure 3.1. If a condition due to low capacity or drift detection is met, a new node is added to the respective module using Xavier's initialization [140]. This increases the module's capacity and reduces its bias. In the absence of drift, the bias of the network ought to reduce or maintain a consistent level.

In contrast, should the pruning criterion be fulfilled, the least significant hidden unit r^* will be eliminated, taking into account its anticipated level of activation. This process is carried out by minimizing $\sigma(\cdot)$ over all hidden $\mathcal{R}$ nodes in the module's last layer, using the equation:

$$r^* = \min_{r=1,\ldots,\mathcal{R}} \sigma \left(\frac{\mu^r}{\sqrt{1 + \pi \frac{\sigma^{2^r}}{8}}} \mathcal{W}^r + b^r \right) \tag{3.15}$$

the module's last hidden layer comprises of $\mathcal{R}$ nodes, with $\mathcal{W}$ representing the weight and b representing the bias. The objective of pruning is to reduce the module's capacity and mitigate over-fitting. It is noteworthy that if the value of r^* is small, it indicates that the corresponding hidden node plays a minimal role in generating the module's output and can be pruned without causing substantial accuracy degradation.

Each module in the ACDC has a distinct goal function, so their growing and pruning conditions activate at different times in response to various drift levels. Additionally, any modification to the hidden layer of the DAE's encoder affects DAA and DISC, as the encoder is shared within the ACDC structure. To tackle this issue, ACDC includes an optional hyper-parameter called κ which decides the number of internal epochs that ACDC can carry out. ACDC still adheres to the prequential test-then-train protocol, which means that it does not revisit a sample after completing a batch. However, within the same batch, the first internal epoch functions as a drift detection and adaptation epoch, while the following internal epochs are solely learning iterations. This technique allows the network to learn more efficiently under the new configuration. A batch is a collection of N_m samples that are assigned to sliding windows W_S and W_T, such that W_S and W_T may have different sizes but add up to N_m.

In addition, DAA's loss function enables it to identify asynchronous shift with greater nuance, and ACDC exploits this benefit by indicating DISC to add a node

whenever DAA's expansion criterion is triggered. This approach enhances DISC's adaptability to variations in the source and target distributions, as confirmed by the ablation analyses. In the event that the compelled expansion signal from DAA to DISC causes any damage to the latter, DISC will promptly correct the issue by initiating its pruning criteria.

Ultimately, regarding the starting quantity of concealed nodes, DAA and DISC commence with one node in their covert layer, while DAE sets off its hidden layer with $u/2$ nodes, where u represents the feature vector size or feature dimensionality. Therefore, the standard practice is to opt for $u/2$ as the initial number of hidden nodes, which furnishes ample compression capacity and learning speed for a denoising autoencoder.

3.2.3 Cross-domain adaptation theoretical analysis

Many approaches bound the target error by the sum of the source error and a notion of distance between the source and the target distributions to tackle the challenging domain adaptation problem, as described by Theorem 2.1.

We assume in Theorem 2.1 that the hypothesis class $\mathcal{H}$ is a (discrete or continuous) set of binary classifiers $\eta : X \to [0, 1]$, which represents the source and target domain streams. The $\mathcal{H}$-divergence relies on the capacity of the hypothesis class $\mathcal{H}$ to distinguish between examples generated by $\mathcal{D}_{\mathcal{S}}$ from examples generated by $\mathcal{D}_{\mathcal{T}}$. For a symmetric hypothesis class $\mathcal{H}$, one can compute the *empirical* $\mathcal{H}$*-divergence* between two domains $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}$ using Equation (2.5).

ACDC aims to address two key challenges in the target stream: the *lack of labeled samples* and *covariate shift*. Our approach involves utilizing a dynamic domain-invariant network to acquire a model that can generalize well across multiple domains. To prevent the neural network's internal representation from encoding any discriminative knowledge about the input's original domain, we utilize a simplified domain-adversarial classifier [1]. The classifier, represented by a logistic regressor DAA : $\mathbb{R}^{\mathcal{D}} \to [0, 1]$, estimates Equation (2.5) and is responsible for cross-domain adaptation. It is worth noting that the proposed method leverages the domain-adversarial classifier to perform feature extraction and adapt to the target domain's

changing statistics. Equation (3.16) describes the loss function for our approach, where we optimize the parameters $\hat{\theta}dae$, $\hat{\theta}disc$, and $\hat{\theta}_{daa}$.

$$E(\theta_{dae}, \theta_{disc}, \theta_{daa}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}^{i}_{DISC}(\theta_{dae}, \theta_{disc}) \\ - \frac{1}{n}\sum_{i=1}^{n} \hat{\mathcal{L}}^{i}_{DAA}(\theta_{dae}, \theta_{daa}) - \frac{1}{N-n}\sum_{i=n+1}^{N} \hat{\mathcal{L}}^{i}_{DAA}(\theta_{dae}, \theta_{daa}) \tag{3.16}$$

$$(\hat{\theta}_{dae}, \hat{\theta}_{disc}) = \underset{\theta_{dae}, \theta_{disc}}{argmin}\, E(\theta_{dae}, \theta_{disc}, \hat{\theta}_{daa}) \tag{3.17}$$

$$\hat{\theta}_{daa} = \underset{\theta_{daa}}{argmax}\, E(\hat{\theta}_{dae}, \hat{\theta}_{disc}, \theta_{daa}) \tag{3.18}$$

Thus, the optimization problem at hand requires minimizing certain parameters while maximizing others.

3.3 Experiments

The evaluation of ACDC is thoroughly conducted using a comprehensive series of experiments that involve 11 datasets, resulting in 26 distinct experiments. These experiments evaluate ACDC's performance under various conditions, such as different types of concept drifts, sensitivity to different hyperparameters, and performance under sub-optimal GPU conversion.

To evaluate ACDC's ability to adapt to different concept drifts, we design experiments that simulate different scenarios where the data distribution changes over time. Specifically, we consider scenarios with abrupt, gradual, and incremental drifts, each with different magnitudes and rates of change. Additionally, we evaluate ACDC's performance under sub-optimal conditions, such as low GPU memory and reduced computational resources, to assess its robustness and real-world applicability.

To understand the impact of hyperparameters on ACDC's performance, we conduct experiments with varying hyperparameter values and analyze the resulting performance metrics. This allows us to fine-tune the model's hyperparameters to achieve optimal performance.

Furthermore, to analyze the individual contributions of each of ACDC's modules, we conduct an ablation study that dissects the model's performance with and without each module. This analysis provides insight into the role and importance of each module in the online cross-domain adaptation problem. Overall, the comprehensive set of experiments conducted provides a detailed evaluation of ACDC's performance and robustness, allowing us to draw meaningful conclusions about its suitability for real-world applications.

3.3.1 Setup

The evaluation was conducted on a Windows 10 computer equipped with an Intel Core i9-9900K processor operating at 5.0 GHz and 32GB of primary memory. Three instances of ACDC were contrasted against the benchmarks, with the values of κ being $1, 3, 5$, corresponding respectively to the models ACDC-1, ACDC-3 and ACDC-5. Each instance had a sliding window size of $N_m = 1000$, a learning rate of $\lambda = 0.01$, and a momentum rate of $\eta = 0.95$.

For all trials, $\alpha_1 = 1.25$ and $\alpha_2 = 0.75$, except for CIFAR10↔STL10 where $\alpha_1 = 1.45$ and $\alpha_2 = 0.95$. This allowed for appropriate overfitting/underfitting regulation in a densely populated feature domain.

Simultaneously, the benchmarks were executed following the default settings described in their respective publications. However, minor alterations were applied to MSC and FUSION, which only received ten samples during their warm-up phase. These adjustments were implemented after multiple tests to ensure that the benchmarks were not unfairly disadvantaged compared to ACDC.

3.3.2 Benchmarks

Table 3.1 enumerates the datasets used in our experiments. Each of these datasets consists of real-world data and is publicly accessible.

Dataset	Features	Classes	Samples
MNIST(MN)	784	10	70,000
USPS(US)	256	10	9,298
CIFAR10(CF)	512	9	54,000
STL10(ST)	512	9	11,700
London Bike(LD)	8	2	17,414
Washington Bike(WA)	8	2	18,110
Amazon@Beauty(AM1)	300	5	5,150
Amazon@Books(AM2)	300	5	500,000
Amazon@Industrial(AM3)	300	5	73,146
Amazon@Luxury(AM4)	300	5	33,784
Amazon@Magazine(AM5)	300	5	2,230

TABLE 3.1: The attributes of the datasets.

MNIST (MN) ↔ USPS (US): These datasets include grayscale images of handwritten digits from various sources, with 10 shared classes. The USPS dataset [141] has 9,298 images sized 16x16, and the MNIST dataset [142] contains 70,000 images sized 28×28. We resize all images uniformly, with US→MN images being 16×16 and MN→US images being 28×28.

CIFAR10 (CF) ↔ STL10 (ST): Full-color images for training recognition models are found in these datasets. From each set, we remove a distinct category that does not intersect with the other set. CIFAR10 dataset [143] has 54,000 usable images (32×32), while the STL10 [144] contains 11,700 usable images (96×96). All samples undergo feature extraction from images by a ResNet-18 [28] model that has been pre-trained on ImageNet [126].

Amazon@X (AM): The sentiment dataset from Amazon.com [145] encompasses various domains and features X as the product category. Out of the many product types available, we randomly selected five, among which two had comparable contexts, while one was dissimilar. The characteristics of the raw review text are obtained by utilizing the mean aggregated result produced by Google's pre-trained word2vec model, which has been trained on 100 billion words [146].

London (LD) ↔ Washington (WA): These tabular datasets [147, 148] describe bike-sharing practices in London and Washington D.C. We preprocess the features

to ensure consistent information representation across both datasets[3].

We simulate abrupt concept drifts in these datasets, with the exception of LD↔WA – which displays genuine displacement caused by its inherent characteristics, using a scaling hyperplane method. Following a specific time instance, each data point is transformed into $x_i = (d_z \times x_i)/||x||$, where $d_z \in [0, ...2)^u$ is a randomly generated concept drift vector. The number of concept drifts in the stream is represented by z, with $z = 5$ for source streams and $z = 7$ for target streams, necessitating asynchronous drift during sample throughput[4]. Finally, $d_1 = 1$ for source and target streams, and adjacent d_z are generated using fixed pseudo-random seeds to ensure fair comparison between baselines.

3.3.3 Baselines

We opted for a set of baseline methods that provided their source code online, including ATL [51], FUSION [61], Melanie [63], and MSC [60]. To handle contrasting throughput, we modified their respective codes[5].

In order to incorporate a varying throughput data-stream scenario into the benchmarks, we adapted ACDC and each baseline method to account for a specific ratio of source to target samples. The ratio is given by $(n_S - n_{W_S})/(n_S + n_T - n_{W_S} - n_{W_T})$. In this formula, n_S, n_T, n_{W_S}, and n_{W_T} represent the total number of source and target samples and the number of source and target samples already received, respectively.

As the data-stream progresses, the incoming ratio is continually updated with each new sample. This ensures that the system can accommodate and respond to the fluctuating data throughput, making the evaluation of the baseline methods more realistic and informative.

Experiment Source	Target	MSC (%)	ATL (%)	FUSION (%)	Melanie (%)	**ACDC-1** (%)	**ACDC-3** (%)	**ACDC-5** (%)
MN	US	46.83 ± 0.29	*54.18 ± 4.50	24.39 ± 1.87	10.67 ± 0.00	53.42 ± 0.92	*56.57 ± 2.27	***59.12 ± 4.62**
US	MN	09.19 ± 0.48	21.82 ± 5.51	16.49 ± 1.04	13.20 ± 0.00	41.16 ± 1.54	***48.21 ± 1.28**	***48.71 ± 1.14**
CF	ST	***50.50 ± 1.57**	17.02 ± 2.35	19.90 ± 1.47	11.22 ± 0.00	42.34 ± 4.43	43.38 ± 2.53	36.73 ± 2.72
ST	CF	36.60 ± 0.31	19.39 ± 0.30	17.80 ± 0.82	11.20 ± 0.00	***44.00 ± 1.66**	*42.86 ± 1.26	37.79 ± 1.50
LD	WA	65.73 ± 0.28	64.25 ± 0.96	63.05 ± 2.43	63.01 ± 0.02	66.91 ± 0.01	***69.73 ± 0.01**	**69.35 ± 0.01**
WA	LD	60.17 ± 0.06	65.06 ± 0.20	51.92 ± 0.73	62.15 ± 0.01	64.49 ± 0.01	***66.22 ± 0.01**	65.62 ± 0.02
AM1	AM2	***62.56 ± 0.04**	59.37 ± 0.45	***62.60 ± 0.01**	14.71 ± 0.01	***62.55 ± 0.25**	***62.41 ± 0.21**	***62.62 ± 0.12**
	AM3	**72.50 ± 0.03**	69.27 ± 1.84	**72.53 ± 0.00**	26.58 ± 0.01	***72.98 ± 0.00**	***72.98 ± 0.00**	***72.98 ± 0.00**
	AM4	57.78 ± 0.01	55.88 ± 2.13	57.07 ± 0.66	9.52 ± 0.02	***60.25 ± 0.02**	**60.24 ± 0.02**	***60.25 ± 0.02**
	AM5	63.87 ± 0.59	64.87 ± 0.26	64.64 ± 0.00	5.60 ± 0.11	***71.28 ± 0.15**	***71.11 ± 0.28**	**70.96 ± 0.23**
AM2	AM1	86.84 ± 1.24	68.98 ± 15.2	86.06 ± 1.52	4.35 ± 0.00	***88.34 ± 0.03**	***88.36 ± 0.02**	***88.33 ± 0.04**
	AM3	67.29 ± 2.99	54.06 ± 9.93	71.28 ± 0.14	6.89 ± 0.01	***72.58 ± 0.00**	***72.58 ± 0.00**	***72.58 ± 0.00**
	AM4	53.32 ± 0.61	47.50 ± 7.77	54.11 ± 1.23	4.10 ± 0.00	***57.95 ± 0.02**	***57.96 ± 0.01**	***57.95 ± 0.01**
	AM5	60.51 ± 2.99	58.48 ± 3.24	***64.63 ± 0.00**	3.62 ± 0.01	***64.57 ± 0.07**	***64.60 ± 0.03**	**64.51 ± 0.03**
AM3	AM1	**88.35 ± 0.00**	86.92 ± 2.72	87.78 ± 0.00	8.40 ± 0.00	***88.78 ± 0.00**	**88.77 ± 0.00**	**88.77 ± 0.00**
	AM2	61.99 ± 0.15	52.83 ± 3.83	**62.57 ± 0.02**	13.39 ± 0.02	***62.67 ± 0.00**	***62.67 ± 0.00**	***62.67 ± 0.00**
	AM4	57.81 ± 0.00	55.74 ± 1.91	57.71 ± 0.04	5.67 ± 0.01	***58.65 ± 0.02**	**58.63 ± 0.01**	**58.63 ± 0.03**
	AM5	64.64 ± 0.00	64.11 ± 0.27	64.64 ± 0.00	3.65 ± 0.01	***65.25 ± 0.12**	**65.15 ± 0.09**	**65.11 ± 0.03**
AM4	AM1	86.55 ± 1.41	73.65 ± 15.7	88.35 ± 0.00	13.40 ± 0.01	***89.08 ± 0.02**	88.76 ± 0.59	***89.06 ± 0.02**
	AM2	47.41 ± 1.72	48.51 ± 2.10	**62.49 ± 0.07**	14.10 ± 0.00	**62.28 ± 0.89**	***62.73 ± 0.00**	***62.73 ± 0.00**
	AM3	56.22 ± 2.63	59.94 ± 10.1	**72.53 ± 0.00**	20.46 ± 0.01	**72.94 ± 0.21**	***73.05 ± 0.00**	***73.04 ± 0.00**
	AM5	64.44 ± 0.21	54.98 ± 5.25	64.64 ± 0.00	3.90 ± 0.01	***65.34 ± 0.08**	***65.40 ± 0.06**	***65.40 ± 0.09**
AM5	AM1	84.87 ± 2.37	79.06 ± 1.35	22.32 ± 10.6	57.05 ± 0.05	***90.72 ± 0.03**	**90.68 ± 0.04**	**90.66 ± 0.05**
	AM2	43.12 ± 3.67	58.71 ± 2.62	61.94 ± 0.14	14.80 ± 0.00	54.89 ± 14.9	61.51 ± 1.39	***62.33 ± 0.35**
	AM3	52.51 ± 7.96	71.65 ± 1.24	70.42 ± 1.35	27.36 ± 0.01	***73.03 ± 0.00**	***73.04 ± 0.00**	***73.04 ± 0.00**
	AM4	50.66 ± 4.72	55.79 ± 2.23	51.66 ± 0.83	10.16 ± 0.01	**58.89 ± 0.01**	***58.90 ± 0.00**	***58.90 ± 0.00**

TABLE 3.2: A comparison of the target accuracy for each experiment. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final obtained target accuracy. Any results that achieved statistical significance with a p-value greater than 0.05 are indicated with an asterisk (*) for easy identification. The best result in each experiment is highlighted with bold font, while the second-best is highlighted with underline font.

3.3.4 Numerical results

Positive conclusions can be drawn from Tables 3.2, 3.3, and 3.4. In each trial, except for CF→ST, ACDC has the ability to attain the utmost target classification accuracy rate frequently outperforming the baselines. Although Melanie and the other baselines show competent performance on various benchmarks, only ACDC demonstrates a consistent delivery of outstanding metrics, including F1 scores (both macro and weighted)[6].

Figure 3.4 provides additional insights by illustrating the changes in the categorization accuracy of ACDC's origin and destination in the US→MN trial over time.

[3]As an illustration, instead of utilizing discrete datasets with binary characteristics of "weekday" and "weekend," preprocessing is employed to ensure the characteristics signify the corresponding information in both datasets.

[4]Artificial asynchronous drifts are ensured, as $z = 5$ and $z = 7$ are prime numbers

[5]Find these modifications in ACDC's code repository: `https://github.com/Ivsucram/ACDC`

[6]We could not edit Melanie's code to compute its F1 scores.

Experiment Source	Target	MSC (%)	ATL (%)	FUSION (%)	**ACDC-1** (%)	**ACDC-3** (%)	**ACDC-5** (%)
MN	US	42.53 ± 0.54	49.10 ± 1.60	13.67 ± 3.12	51.67 ± 3.03	**58.08 ± 01.05**	52.14 ± 2.48
US	MN	18.43 ± 0.36	23.38 ± 1.60	08.20 ± 2.63	**43.68 ± 1.38**	**43.72 ± 01.75**	**44.76 ± 1.66**
CF	ST	**47.26 ± 7.15**	20.00 ± 3.45	13.40 ± 3.02	37.19 ± 1.59	39.29 ± 4.07	40.45 ± 2.78
ST	CF	37.18 ± 0.78	17.80 ± 2.74	11.85 ± 3.74	**42.52 ± 2.81**	**43.68 ± 1.58**	40.56 ± 2.39
LD	WA	66.55 ± 0.43	63.91 ± 3.74	34.69 ± 7.93	65.60 ± 3.17	66.81 ± 1.35	**67.88 ± 0.32**
WA	LD	56.00 ± 1.75	**64.55 ± 0.54**	31.93 ± 5.76	61.22 ± 3.24	**64.12 ± 1.41**	63.82 ± 3.23
AM1	AM2	15.40 ± 0.01	**17.17 ± 0.20**	12.93 ± 0.07	15.42 ± 0.02	15.40 ± 0.01	15.41 ± 0.01
	AM3	16.81 ± 0.00	**18.29 ± 0.02**	14.24 ± 0.14	16.82 ± 0.00	16.82 ± 0.00	16.82 ± 0.00
	AM4	14.67 ± 0.05	**17.04 ± 0.02**	13.12 ± 0.13	14.71 ± 0.00	14.71 ± 0.00	14.71 ± 0.00
	AM5	15.77 ± 0.09	**17.85 ± 0.01**	13.92 ± 0.71	15.69 ± 0.01	15.69 ± 0.01	15.70 ± 0.01
AM2	AM1	18.93 ± 0.25	**20.89 ± 1.03**	17.39 ± 0.28	18.76 ± 0.00	18.76 ± 0.01	18.76 ± 0.01
	AM3	18.27 ± 0.40	**20.48 ± 1.43**	14.48 ± 0.53	16.82 ± 0.00	16.82 ± 0.00	16.82 ± 0.00
	AM4	18.49 ± 0.37	**21.48 ± 1.11**	15.00 ± 2.30	14.65 ± 0.00	14.65 ± 0.00	14.65 ± 0.00
	AM5	17.75 ± 3.19	**22.85 ± 0.27**	14.16 ± 1.01	15.69 ± 0.01	15.69 ± 0.01	15.70 ± 0.01
AM3	AM1	18.76 ± 0.00	**19.02 ± 0.00**	15.62 ± 0.28	18.79 ± 0.00	18.79 ± 0.00	18.79 ± 0.00
	AM2	17.27 ± 1.92	**18.84 ± 0.18**	13.20 ± 0.59	15.40 ± 0.00	15.40 ± 0.00	15.40 ± 0.00
	AM4	14.65 ± 0.00	**15.18 ± 0.00**	14.88 ± 0.75	14.66 ± 0.00	14.66 ± 0.00	14.66 ± 0.00
	AM5	**15.70 ± 0.00**	**15.68 ± 0.00**	14.86 ± 1.02	**15.69 ± 0.01**	**15.69 ± 0.00**	**15.69 ± 0.01**
AM4	AM1	**18.89 ± 0.06**	18.72 ± 0.00	13.91 ± 2.34	18.79 ± 0.01	18.78 ± 0.01	18.78 ± 0.00
	AM2	**19.60 ± 0.00**	**19.61 ± 0.03**	13.95 ± 0.74	15.40 ± 0.00	15.40 ± 0.00	15.40 ± 0.00
	AM3	19.00 ± 0.12	**19.92 ± 0.00**	14.48 ± 0.30	16.89 ± 0.13	16.95 ± 0.17	16.88 ± 0.13
	AM5	15.75 ± 0.07	**17.10 ± 0.00**	12.88 ± 0.09	15.67 ± 0.01	15.67 ± 0.00	15.67 ± 0.01
AM5	AM1	**18.90 ± 1.89**	18.14 ± 0.00	08.68 ± 1.38	18.78 ± 0.00	18.79 ± 0.00	18.79 ± 0.00
	AM2	**17.47 ± 2.42**	17.66 ± 1.11	13.13 ± 0.22	15.80 ± 0.48	15.47 ± 0.04	15.62 ± 0.02
	AM3	16.37 ± 3.02	**17.13 ± 0.00**	15.12 ± 0.40	16.82 ± 0.00	16.82 ± 0.00	16.82 ± 0.00
	AM4	**16.10 ± 1.98**	14.88 ± 0.00	14.88 ± 0.75	14.64 ± 0.00	14.64 ± 0.00	14.64 ± 0.00

TABLE 3.3: The comparison of F1 scores in macro format for each experiment. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final F1 macro score. The best result in each experiment is highlighted with bold font, while the second-best is highlighted with underline font.

Given the stochastic nature of the incoming data, it is difficult to precisely identify when the concept drift takes place. However, in our conducted experiment, we partitioned the source stream into five distinct concepts, and the target stream into seven. Considering the source stream classification rate, it is evident that ACDC quickly recovers from concept drifts thanks to its evolution capabilities, addressing network bias and variance disturbances. The labeled source stream makes this analysis particularly valuable as it enables us to assess DISC's adaptability performance mainly while minimizing any disturbance from DAA.

The CF↔ST experiments, as shown in Table 3.2, are curious cases. The feature spaces became too dense after applying ResNet-18, requiring us to increase α_1 and α_2. However, despite ACDC's ability to grow and prune nodes during training, it overfits when κ is increased, resulting in a decline in performance when employing the prequential test-followed-by-training procedure. The following sub-section

Experiment Source	Target	MSC (%)	ATL (%)	FUSION (%)	**ACDC-1** (%)	**ACDC-3** (%)	**ACDC-5** (%)
MN	US	46.55 ± 0.58	53.39 ± 1.62	13.67 ± 3.12	55.83 ± 3.04	52.73 ± 0.96	**56.18 ± 2.44**
US	MN	18.71 ± 0.34	24.62 ± 1.62	09.24 ± 2.85	43.88 ± 1.34	43.93 ± 1.74	**44.97 ± 1.65**
CF	ST	**47.26 ± 7.16**	21.14 ± 3.55	14.89 ± 3.35	37.19 ± 1.58	39.28 ± 4.08	40.44 ± 2.77
ST	CF	37.01 ± 0.54	18.02 ± 2.72	13.17 ± 4.15	42.51 ± 2.81	**43.67 ± 1.58**	40.55 ± 2.38
LD	WA	66.56 ± 0.43	66.54 ± 00.00	52.04 ± 11.85	65.60 ± 3.16	66.80 ± 1.35	**67.87 ± 0.32**
WA	LD	56.00 ± 1.75	**64.16 ± 00.00**	51.34 ± 4.55	61.13 ± 3.24	**64.03 ± 1.41**	63.74 ± 3.26
AM1	AM2	48.20 ± 0.01	**48.41 ± 0.18**	48.18 ± 0.10	48.19 ± 0.02	48.19 ± 0.02	48.18 ± 0.03
	AM3	**60.94 ± 0.02**	60.45 ± 0.02	**60.90 ± 0.07**	**60.98 ± 0.00**	**60.98 ± 0.00**	**60.98 ± 0.00**
	AM4	42.34 ± 0.03	43.61 ± 0.02	45.07 ± 6.54	**52.79 ± 0.01**	**52.79 ± 0.01**	**52.78 ± 0.01**
	AM5	50.37 ± 0.36	50.16 ± 0.04	48.24 ± 1.12	**50.65 ± 0.07**	**50.60 ± 0.08**	**50.72 ± 0.05**
AM2	AM1	82.43 ± 0.57	72.39 ± 1.05	81.58 ± 0.52	**82.90 ± 0.05**	**82.86 ± 0.07**	**82.87 ± 0.07**
	AM3	60.05 ± 1.13	47.60 ± 1.52	57.78 ± 6.33	**60.98 ± 0.00**	**60.99 ± 0.00**	**60.99 ± 0.00**
	AM4	63.80 ± 26.91	45.35 ± 1.05	44.48 ± 2.55	**82.90 ± 0.05**	**82.86 ± 0.07**	**82.87 ± 0.07**
	AM5	46.56 ± 0.94	**52.76 ± 0.22**	49.50 ± 2.42	50.65 ± 0.07	50.60 ± 0.08	50.72 ± 0.05
AM3	AM1	82.87 ± 0.01	42.24 ± 0.00	80.81 ± 1.43	**83.23 ± 0.03**	**83.22 ± 0.02**	**83.23 ± 0.03**
	AM2	48.06 ± 0.86	47.73 ± 0.12	48.14 ± 0.03	**48.21 ± 0.00**	**48.21 ± 0.00**	**48.21 ± 0.00**
	AM4	42.35 ± 0.00	42.24 ± 0.00	41.89 ± 0.20	**42.43 ± 0.01**	**42.43 ± 0.01**	**42.43 ± 0.01**
	AM5	**50.75 ± 0.00**	50.56 ± 0.00	50.60 ± 0.50	50.63 ± 0.05	50.64 ± 0.04	50.62 ± 0.05
AM4	AM1	82.73 ± 0.09	74.27 ± 0.00	66.17 ± 16.24	**83.20 ± 0.07**	**83.15 ± 0.08**	**83.12 ± 0.04**
	AM2	**46.74 ± 0.28**	46.33 ± 0.06	**48.24 ± 0.88**	**48.21 ± 0.00**	**48.21 ± 0.00**	**48.21 ± 0.00**
	AM3	52.03 ± 0.41	54.72 ± 0.00	60.29 ± 0.57	**61.04 ± 0.05**	**61.04 ± 0.05**	**61.02 ± 0.02**
	AM5	**50.77 ± 0.01**	48.47 ± 0.00	47.65 ± 2.87	50.46 ± 0.06	50.47 ± 0.04	50.43 ± 0.05
AM5	AM1	74.00 ± 11.50	76.33 ± 0.00	27.73 ± 6.04	**83.14 ± 00.02**	**83.22 ± 00.08**	**83.18 ± 00.06**
	AM2	41.07 ± 9.32	**48.50 ± 1.15**	**48.24 ± 0.15**	**48.30 ± 0.27**	**48.16 ± 0.02**	**48.22 ± 0.02**
	AM3	51.31 ± 10.91	60.37 ± 0.00	58.97 ± 1.30	**61.00 ± 00.00**	**61.00 ± 00.00**	**61.00 ± 00.00**
	AM4	42.01 ± 0.54	41.90 ± 0.00	41.71 ± 0.85	**42.23 ± 00.00**	**42.23 ± 00.00**	**42.23 ± 00.00**

TABLE 3.4: The comparison of F1 scores in weighted format for each experiment. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final F1 weighted score. The best result in each experiment is highlighted with bold font, while the second-best is highlighted with underline font.

presents an analysis of different values for α_1 and α_2.

ACDC implements internal epochs at the granularity of data chunks with a size of N_m instead of processing the entire dataset, thereby satisfying the demands of online learning. Discarded are the processed batches, and they are not re-visited. Nonetheless, enhancing the number of internal epochs κ may not always enhance the classification rate performance, as observed in most of the experiments of *Amazon@X* and, to a smaller degree, the WA↔LD experiments that involve a considerable amount of inherent noise.

Table 3.2 did not feature MSDA and COMC [66, 67] as their source codes were not accessible to the public. Despite that, the US→MN experiment results published in their paper exhibit some overlap. Specifically, COMC attains a target accuracy of 42.37%, whereas MSDA attains $28.72 \pm 0.56\%$ and $29.96 \pm 0.64\%$.[7]. ACDC still

[7]Results for the different variations of the MSDA method.

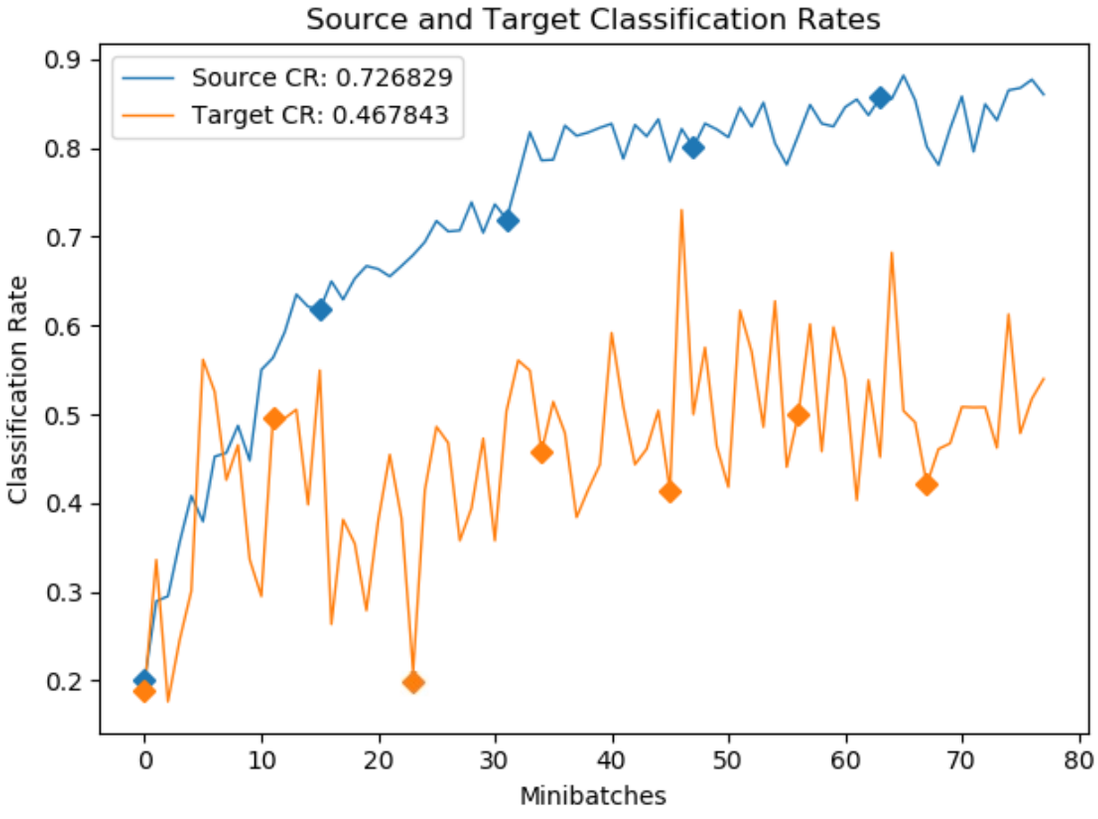


FIGURE 3.4: The progress of the classification rate (CR) for the source and target and the approximate positions of concept drift in the US→MN experiment. The top-left corner displays the final CR achieved in the experiment.

outperforms these results.

3.3.5 Hyper-parameter sensitivity

In order to better comprehend the sensitivity of ACDC's performance to its hyper-parameters, we carried out a series of experiments using ACDC-1 on the MN↔US experiments, with the results shown in Table 3.3.5.

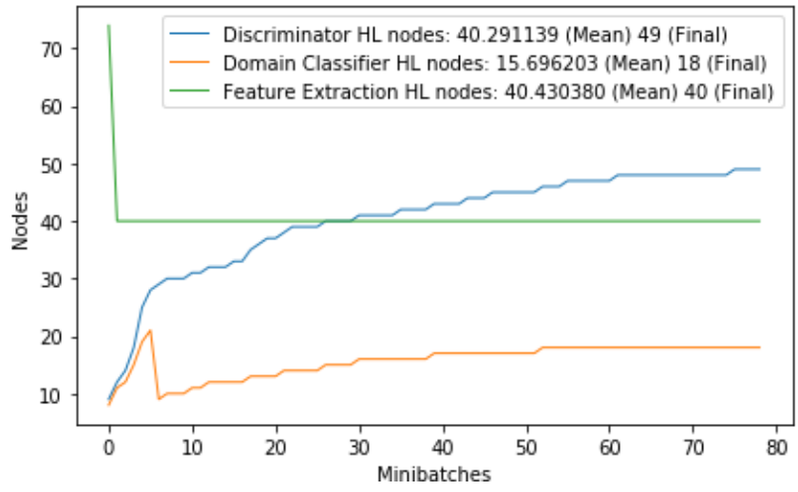
The hyper-parameter N_m governs the size of ACDC's sliding window. Although the parameter learning procedure for ACDC operates under a tuple (x_s, x_t, y_s), the samples are initially paired and permuted into new arrays W'_S and W'_T to handle contrasting throughput (Algorithm 1, line 4). As a result, a larger sliding window N_m can better adapt to streams with varying speeds.

Hyper-parameter		MN → US (%)	US → MN (%)
N_m	25	48.65 ± 2.64	20.52 ± 8.60
	100	48.76 ± 1.82	37.94 ± 1.32
	250	48.67 ± 2.07	38.33 ± 0.91
	500	47.78 ± 2.59	38 ± 1.03
	1000	53.42 ± 00.42	41.16 ± 1.54
l_{dae}	1	25.70 ± 3.24	25.75 ± 2.73
	u/5	46.93 ± 2.80	38.41 ± 1.64
	u/2	53.42 ± 00.42	41.16 ± 1.54
	u	46.93 ± 2.48	38.41 ± 1.64
$(\alpha_1; \alpha_2)$	(1.05; 0.55)	00.00 ± 00.00	00.00 ± 00.00
	(1.25; 0.75)	53.42 ± 00.42	41.16 ± 1.54
	(1.45; 0.95)	46.68 ± 2.59	33.68 ± 2.30
	(1.65; 1.15)	41.07 ± 3.22	32.31 ± 2.10

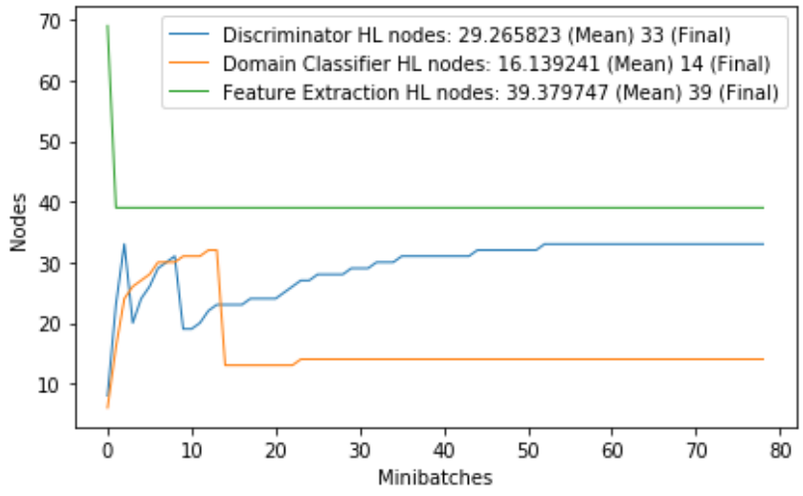
TABLE 3.5: The comparison of target accuracy for each experiment on ACDC-1, with variations in the values of hyper-parameters. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final accuracy scores.

Another critical factor in learning and adaptation is the network's initial size. DISC and DAA can efficiently function with a single initial latest latent node due to their discriminative generalization power, as shown in Equations (3.9) and (3.8). However, the choice of the DAE's latent space size, l_{dae}, is crucial for the speed at which the model learns, which is vital for incremental learning models. If l_{dae} is initialized with only a few nodes, it lacks the capacity to reconstruct the input samples, resulting in faulty features being passed to both DISC and DAA. Conversely, if DAE starts with an excessive number of nodes, extracting meaningful features takes longer, even when extra nodes are continually pruned.

The hyper-parameters α_1 and α_2 regulate the inner workings of the SPC algorithm, which underlies ACDC's drift detection and dynamically evolving structure. They modify the confidence interval within which ACDC assesses its bias-variance trade-off. Generally, higher $(\alpha_1; \alpha_2)$ values result in the network creating fewer nodes, while lower values trigger the growing condition more frequently. These hyper-parameters exhibit varying performance on each benchmark, and we discovered that the pair (1.25; 0.75) acts as a useful guideline, typically yielding average positive results across datasets.



(A) Internal epoch $\kappa = 1$



(B) Internal epoch $\kappa = 5$

FIGURE 3.5: The evolution of Hidden Layer (HL) nodes using various κ values on experiment US→MN. The plots show the average number of nodes per mini-batch, which is evaluated over a sliding window, as well as the number of nodes in the final batch.

When aiming for a specific level of precision, certain experiments may reap advantages from a lengthier training period, as evidenced by the data presented in Table 3.2. In this examination, we investigate the impact of increasing the number of internal epochs (κ) on the parameter learning process for ACDC.

Diagrams illustrating the structural progression during the US→MN experiment can be seen in Figures 3.5a and 3.5b. The former shows the evolution at the first internal epoch $\kappa = 1$, while the latter depicts the changes at $\kappa = 5$. It should be noted that module adaptation is restricted to the first internal epoch, but parameter learning occurs in all internal epochs. By setting $\kappa > 1$, ACDC

enhances its parameter learning by optimizing each batch before moving on to the next, although this increases the time complexity.

By analyzing Figures 3.5a and 3.5b, we can infer that when $\kappa > 1$, the experiments tend to have more pertinent network parameters. This is because they can achieve comparable or superior performance in the US→MN experiment while utilizing fewer hidden nodes per module.

3.3.6 Space and Time complexity

Experiment Source	Target	MSC (s)	ATL (s)	FUSION (s)	Melanie (s)	**ACDC-1** (s)	**ACDC-3** (s)	**ACDC-5** (s)
MN	US	8,498 ± 395	66,160 ± 3,210	**110 ± 3**	1,002 ± 41	4,635 ± 12	13,199 ± 1,160	23,258 ± 1,302
US	MN	11,587 ± 9,111	346 ± 5	**115 ± 4**	555 ± 26	1,917 ± 109	3,299 ± 25	4,926 ± 165
CF	ST	5,900 ± 181	1,721 ± 245	**95 ± 6**	559 ± 64	1,536 ± 169	3,636 ± 290	4,945 ± 109
ST	CF	11,437 ± 245	18,323 ± 171	**147 ± 9**	1,181 ± 119	1,564 ± 119	3,381 ± 280	4,580 ± 280
LD	WA	2,789 ± 40	254 ± 3	30 ± 0	**2 ± 0**	172 ± 7	176 ± 1	175 ± 2
WA	LD	3,124 ± 109	262 ± 3	31 ± 0	**2 ± 0**	169 ± 13	167 ± 10	152 ± 10
AM1	AM2	58,683 ± 1,846	16,936 ± 601	**960 ± 29**	1,243 ± 100	12,869 ± 561	20,780 ± 1,447	27,918 ± 2,880
	AM3	13,282 ± 1,678	238 ± 6	150 ± 18	**144 ± 4**	1,923 ± 78	3,824 ± 99	5,649 ± 53
	AM4	6,730 ± 1,480	201 ± 2	68 ± 2	**59 ± 2**	870 ± 37	1,685 ± 74	2,497 ± 185
	AM5	787 ± 43	13 ± 1	11 ± 0	**8 ± 0**	120 ± 2	242 ± 2	361 ± 3
AM2	AM1	39,527 ± 3,411	11,683 ± 527	**531 ± 37**	529 ± 13	11,305 ± 158	22,932 ± 1,957	41,645 ± 10,019
	AM3	58,406 ± 4,711	17,996 ± 1,223	**729 ± 55**	763 ± 14	11,424 ± 293	21,529 ± 606	38,908 ± 6,019
	AM4	45,749 ± 3,392	13,705 ± 792	**586 ± 36**	646 ± 10	15,512 ± 2,228	22,563 ± 1,296	35,274 ± 4,743
	AM5	34,540 ± 818	11,393 ± 403	**552 ± 34**	**554 ± 7**	17,448 ± 5,242	23,634 ± 2,335	37,271 ± 3,544
AM3	AM1	7,220 ± 551	**85 ± 2**	90 ± 8	137 ± 1	1,965 ± 89	3,853 ± 165	5,909 ± 43
	AM2	58,631 ± 1,127	16,961 ± 282	**1,067 ± 66**	**1,602 ± 47**	11,545 ± 358	27,726 ± 2,100	43,329 ± 6,129
	AM4	12,294 ± 630	170 ± 3	**143 ± 12**	179 ± 12	1,639 ± 19	3,081 ± 30	4,571 ± 36
	AM5	6,708 ± 398	**74 ± 2**	82 ± 6	116 ± 2	1,644 ± 13	3,300 ± 256	5,524 ± 320
AM4	AM1	3,281 ± 163	119 ± 2	**47 ± 2**	53 ± 0	1,090 ± 79	2,077 ± 264	3,077 ± 151
	AM2	53,670 ± 2,892	16,678 ± 607	**974 ± 58**	1,398 ± 72	15,087 ± 722	27,914 ± 1,308	37,773 ± 5,947
	AM3	14,285 ± 1,132	350 ± 16	177 ± 11	**172 ± 4**	1,633 ± 20	3,226 ± 146	4,451 ± 326
	AM5	2,906 ± 192	132 ± 6	**40 ± 2**	46 ± 0	760 ± 15	1,449 ± 23	2,337 ± 172
AM5	AM1	579 ± 22	17 ± 0	13 ± 0	**12 ± 0**	163 ± 36	369 ± 57	420 ± 69
	AM2	38,079 ± 2,262	14,433 ± 546	**961 ± 43**	1,376 ± 120	12,314 ± 1,001	21,116 ± 3,685	38,171 ± 1,390
	AM3	7,268 ± 667	218 ± 3	**137 ± 5**	150 ± 4	1,807 ± 99	3,637 ± 269	5,459 ± 438
	AM4	3,277 ± 259	172 ± 2	62 ± 2	**54 ± 0**	894 ± 53	1,587 ± 111	2,258 ± 236

TABLE 3.6: The comparison of the trainig times for each experiment. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final trainig times in secods. The best result in each experiment is highlighted with bold font, while the second-best is highlighted with underline font.

The time complexity of ACDC's training is represented by matrix multiplications involving the latent dimensions of the final layers in the three ACDC modules. These dimensions are l_{dae}, l_{disc}, and l_{daa}, respectively. The training procedure for ACDC is carried out using a tuple of samples, (x_s, x_t, y_s):

$$\underbrace{O(2(\underbrace{u^2 l_{\text{dae}}}_{\text{Eq.(3.1)}} + \underbrace{ul_{\text{dae}}}_{\text{Eq. (3.4)}})}_{\text{DAE } O(\cdot)} + \underbrace{ul_{\text{dae}}l_{\text{disc}}m}_{\text{Eq.(3.3)}} + \underbrace{ul_{\text{dae}}l_{\text{disc}}}_{\text{Eq.(3.6)}} + \underbrace{2(\underbrace{ul_{\text{dae}}l_{\text{daa}}2}_{\text{Eq.(3.2)}} + \underbrace{ul_{\text{dae}}l_{\text{daa}}}_{\text{Eq.(3.5)}})}_{\text{DAA } O(\cdot)})$$

where m and u are respectively the number of classes in the current problem, and the input feature size of the analyzed dataset.

As a result, both DAE and DAA are evaluated twice, while DISC is only computed once. This process can be illustrated in Algorithm 1.

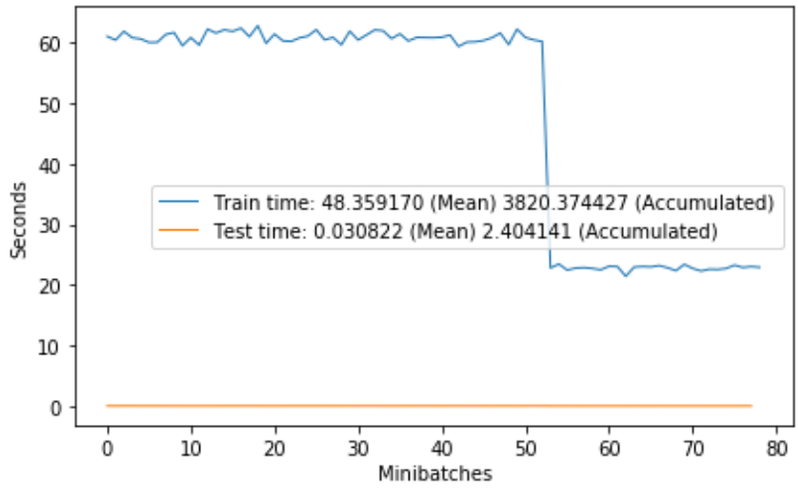
Moreover, during validation, ACDC only uses the forward-pass of the DISC module. This ensures that the validation phase focuses on the discriminative performance of the model:

$$O(ul_{\text{dae}}l_{\text{disc}}m) \tag{3.20}$$

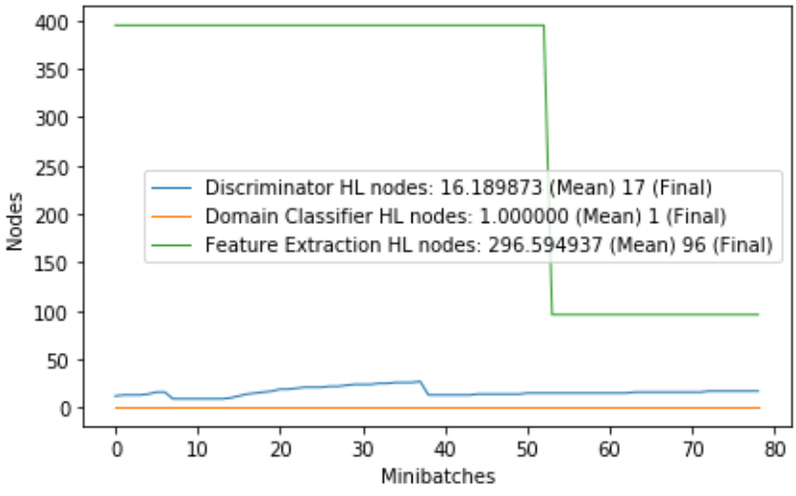
Table 3.6 presents an overview of the overall processing time of ACDC and its baselines concerning time complexity. It is also essential to explore the effect of ACDC's adaptation on both its time and space complexity during the training phase, rather than only measuring it as a final metric. To observe the influence of the network's node count on the training time more precisely, we conducted an ablation study, and this phenomenon is illustrated in Figures 3.6a and 3.6b.

ACDC's modules increase the number of nodes during underfitting and decrease the number of nodes during overfitting, resulting in direct impacts on the memory and computational needs of the model due to changes in the number of nodes. Consequently, when ACDC adds a new node, it increases its space and time complexity, while pruning a node or a group of nodes reduces these complexities. Such observation is supported by Eq. 3.20, where the time complexity is a function of the number of nodes in each module.

ACDC's adaptation conditions are assessed for each sample, causing fluctuations in training time, as seen in Figure 3.6a. However, a significant decrease in training time is observed around chunk 52 in Figure 3.6b, where ACDC prunes multiple nodes from the DAE module, thus significantly reducing its space and time complexity.



(A) Summary of two key pieces of information regarding processing times. The first is the mean, or average, processing time, while the second is the accumulated, or total, processing time.



(B) The evolution of nodes in a hidden layer (HL), with information on the average node count per minibatch (evaluated window), as well as the number of nodes in the final batch. Such steep reduction in few minibatches is possible because ACDC performs the growing and prunning operation per sample, and not per minibatch. So, if a minibatch is composed of 1000 samples, there is an opportunity to grow and prune 1000 nodes, according to the statiscal information of those samples.

FIGURE 3.6: Plots for the ablation study (A), depicting sliding windows information over the MN→US experiment are presented.

3.3.7 Performance over GPU

Time complexity is undeniably the most significant challenge faced by ACDC. The module adaptation procedure adds computational overhead to the training process, leading to increased time complexity. Furthermore, ACDC's existing parameter learning approach is not optimized for GPU utilization, meaning that all

Experiment Source	Experiment Target	Target Accuracy (%)	F1 macro (%)	F1 weighted (%)	Training Time (s)
MN	US	36.03 ± 2.75	24.34 ± 3.84	29.12 ± 3.87	283.8 ± 2.88
US	MN	24.55 ± 1.63	17.29 ± 1.00	17.68 ± 0.98	163.9 ± 2.81
CF	ST	44.02 ± 2.69	36.37 ± 2.09	36.36 ± 2.09	191 ± 1.77
ST	CF	37.85 ± 2.58	34.28 ± 4.31	34.28 ± 4.31	196.7 ± 14.69
LD	WA	62.31 ± 2.61	62.26 ± 2.70	62.26 ± 2.70	45.11 ± 0.71
WA	LD	54.12 ± 2.44	50.67 ± 8.74	50.66 ± 8.75	42.42 ± 0.31
AM1	AM2	62.54 ± 0.08	15.43 ± 0.02	48.20 ± 0.02	1255 ± 15.77
	AM3	72.22 ± 0.23	16.85 ± 0.04	60.89 ± 0.08	183.61 ± 0.57
	AM4	56.80 ± 0.45	14.86 ± 0.09	42.15 ± 0.21	86.32 ± 0.51
	AM5	62.17 ± 1.55	16.44 ± 0.37	50.20 ± 0.40	15.26 ± 0.25
AM2	AM1	88.85 ± 0.37	18.88 ± 0.15	83.47 ± 0.29	1238.48 ± 35.87
	AM3	72.44 ± 0.09	16.82 ± 0.01	60.95 ± 0.02	1347.73 ± 5.08
	AM4	57.73 ± 0.15	14.66 ± 0.02	42.35 ± 0.01	1236.45 ± 35.40
	AM5	63.61 ± 0.42	15.58 ± 0.06	49.62 ± 0.42	1178.03 ± 8.9
AM3	AM1	88.10 ± 0.35	19.12 ± 0.33	82.84 ± 0.03	180.74 ± 1.97
	AM2	62.56 ± 0.04	15.42 ± 0.01	48.21 ± 0.01	1288.66 ± 7.34
	AM4	57.66 ± 0.13	14.71 ± 0.05	42.34 ± 0.02	205.05 ± 19.48
	AM5	64.19 ± 0.92	16.11 ± 0.30	50.77 ± 0.09	189.03 ± 2.75
AM4	AM1	84.45 ± 0.75	19.72 ± 01.12	82.65 ± 0.37	92.25 ± 1.28
	AM2	62.53 ± 0.06	15.43 ± 0.02	48.20 ± 0.01	1253.19 ± 10.88
	AM3	72.26 ± 0.16	16.92 ± 0.06	60.93 ± 0.06	203.18 ± 5.03
	AM5	63.35 ± 1.73	15.92 ± 0.30	50.41 ± 0.40	76.52 ± 0.99
AM5	AM1	82.95 ± 4.89	18.45 ± 0.39	80.54 ± 2.39	14.18 ± 0.40
	AM2	62.25 ± 0.29	15.50 ± 0.05	48.14 ± 0.09	696.97 ± 48.53
	AM3	71.83 ± 0.29	16.93 ± 0.06	60.76 ± 0.10	137.17 ± 0.36
	AM4	57.17 ± 0.24	14.81 ± 0.07	42.19 ± 0.06	65.65 ± 0.27

TABLE 3.7: The comparison of the multiple metrics for each experiment on the ACDC_{GPU}-5 model. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final target accuracy scores.

computations have been executed on the CPU thus far. In this subsection, we explore some minor adjustments to the parameter learning method that would enable ACDC to parallelize specific matrix multiplications using a GPU, potentially improving its overall efficiency.

One efficient way to parallelize ACDC is by performing batch operations on the tuples in the sliding windows W'_S and W'_T. This approach, however, would result in ACDC executing its module adaptation strategy only once per batch. Consequently, there are two potential options: (a) carry out the evolution step multiple times, or (b) decrease the value of N_m to facilitate more frequent evolution. After conducting several tests, we opted for the latter option, reducing N_m from the initial 1000 samples to just 200 samples.

Since a decrease in accuracy performance is expected due to a lower sliding window size, we also increased κ from 1 to 5. This adjustment led to the creation of a model called ACDCGPU-5. The primary metrics for ACDCGPU-5 are summarized in

Table 3.7. This new model achieves similar or slightly lower accuracy rates while reducing training time by a factor of 10 or more. This is due to the fact that it no longer evaluate the present of concept drifts sample by sample, but by a batch of samples at a time - to be more precise, by the average representation of a batch of samples.

Experiments such as MN↔US experienced a considerable reduction in target accuracy when using the ACDC_{GPU}-5 model. However, it's worth noting that this model still outperforms some baselines, including FUSION and Melanie. This finding demonstrates the potential for ACDC to remain competitive even when subjected to modifications aimed at reducing time complexity.

Although the current experiment is not conclusive, it provides valuable insights into the adaptability of ACDC for use in a parallelized environment. By making these minor adjustments, it's possible to maintain ACDC's performance while significantly reducing its time complexity, making it a more viable option for various applications.

3.3.8 Ablation study

In order to gain a deeper understanding of ACDC's behavior, we conducted an ablation study on the MN↔US experiments with $\kappa = 1$. This study was performed under three distinct configurations, each aimed at isolating specific components of ACDC to evaluate their individual impacts on the overall performance. The configurations were as follows:

A. DAA is deactivated: In this scenario, we disabled the DAA module, which is responsible for adapting the model to the target domain. By deactivating DAA, we assessed the effects of its absence on the overall performance of ACDC, particularly in terms of its ability to handle domain adaptation.

B. No self-evolving mechanism: In this configuration, we removed the self-evolving mechanism that allows ACDC to grow and prune nodes dynamically in response to changes in the input data. Instead, we fixed the number of hidden nodes in the hidden layers at 100. This experiment helped us understand the importance of ACDC's self-evolving capabilities and their impact on its performance.

C. The feature of DAA to expand its nodes in DISC has been disabled: In this setup, We deliberately deactivated the DISC module's capacity to expand nodes driven by DAA. This modification allowed us to examine the significance of DAA's contribution to the growth of DISC nodes and its influence on the overall performance of the model.

By evaluating the performance of ACDC under these three different configurations, we aimed to identify the critical components of the model and understand their respective contributions to its effectiveness. This comprehensive analysis provided valuable insights into ACDC's behavior and offered a clearer understanding of the model's strengths and weaknesses.

Ablation	MN→US	US→MN
A(%)	38.74 ± 4.12	27.86 ± 3.68
B(%)	56.22 ± 1.88	37.10 ± 3.85
C(%)	48.84 ± 3.00	38.82 ± 1.43

TABLE 3.8: The results for five ablation study runs.

As an example, study (A) demonstrates that the absence of DAA results in a classification accuracy decrease of at least 5% in both the target and source domains. This demonstrates the essential role DAA plays in ACDC's performance, particularly in addressing domain adaptation challenges.

In study (B), using a fixed structure introduces an additional hyper-parameter to be tuned – the number of hidden nodes – which diminishes ACDC's ease of use. During the training phase, ACDC is capable of dynamically exploring and identifying the most suitable configuration through the process of adaptation. In the MN→US experiment, the fixed structure model outperformed the adaptive network because it was already initialized with a sufficient number of hidden nodes close to the optimal structure size. Conversely, the outcome of the US→MN trial showed a diminished effect since ACDC in this ablation experiment possesses a fixed arrangement that is both more extensive and more distant from the ideal configuration.

Finally, ablation study (C) highlights the crucial role of the interaction between DAA and DISC in achieving rapid learning. This finding emphasizes the importance of maintaining the synergy between these two components in ACDC's architecture.

In conclusion, the ablation study sheds light on the significance of ACDC's key design choices and their impact on the overall performance. These insights can be valuable for future refinements and improvements to the model, ensuring that ACDC remains effective in addressing various domain adaptation challenges.

3.3.9 Handling different concept drifts

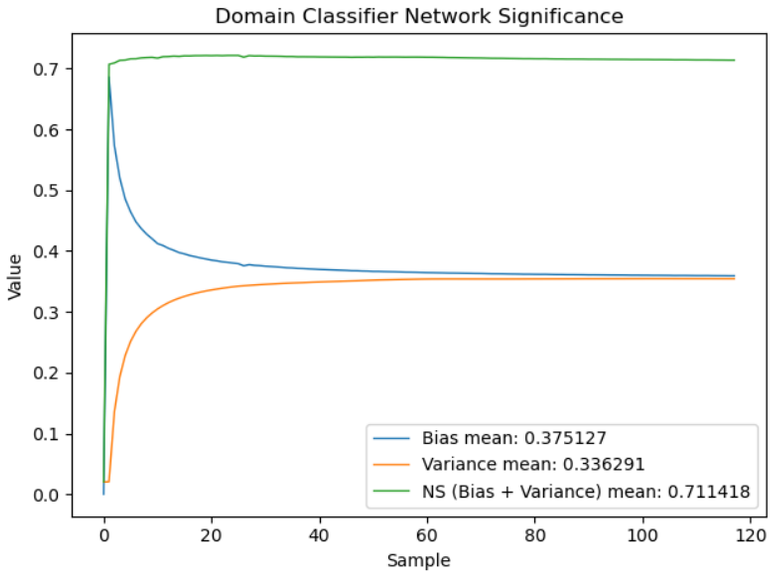
Hyper-parameter		LED Generator (%)	Random RBF Generator (%)
N_m	25	99.91 ± 0.06	99.97 ± 0.04
	100	100.00 ± 00.00	100.00 ± 00.00
	250	100.00 ± 00.00	100.00 ± 00.00
	500	100.00 ± 00.00	100.00 ± 00.00
	1000	100.00 ± 00.00	100.00 ± 00.00
l_{dae}	1	100.00 ± 00.00	100.00 ± 00.00
	u/5	100.00 ± 00.00	100.00 ± 00.00
	u/2	100.00 ± 00.00	100.00 ± 00.00
	u	100.00 ± 00.00	100.00 ± 00.00
$(\alpha_1; \alpha_2)$	(1.05; 0.55)	100.00 ± 00.00	100.00 ± 00.00
	(1.25; 0.75)	100.00 ± 00.00	100.00 ± 00.00
	(1.45; 0.95)	100.00 ± 00.00	100.00 ± 00.00
	(1.65; 1.15)	100.00 ± 00.00	100.00 ± 00.00

TABLE 3.9: The comparison of the target accuracy for each experiment involving incremental concept drift on the ACDC-1 model. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation of their final target accuracy scores.

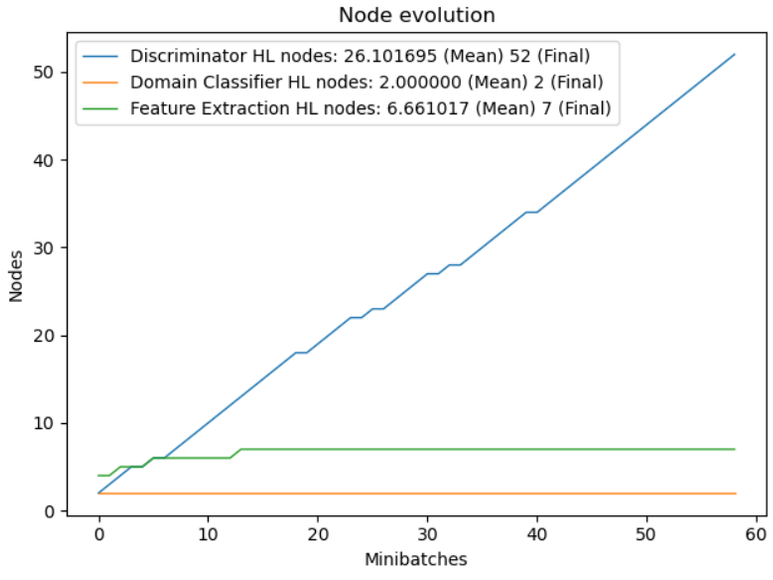
The experiments conducted so far have demonstrated ACDC's performance under sudden concept drifts. In this sub-section, we explore its performance under incremental concept drifts using two synthetic asynchronous incremental concept drift benchmarks built with the scikit-multiflow package [149].

The first benchmark is the **LED Generator**, which originates from Leo Breiman [150]. This dataset utilizes seven binary attributes to predict the digit displayed on a seven-segment LED display. Five of these attributes undergo an incremental concept drift over time.

The second benchmark is the **Random RBF Generator**. In this benchmark, a radial basis function (RBF) stream is generated by creating several centroids with random central positions, standard deviations, class labels, and weights. This process results in a normally distributed hypersphere of samples surrounding each



(A) The relationship between bias and variance. Network Significance (NS) is equal to the bias plus the variance rate.



(B) The evolution of nodes in the hidden layer (HL), showing both the average number of nodes per minibatch (which is evaluated over a specific window of time) and the total number of nodes in the final batch. Even though the bias and variance rate have stabilized, the discriminator continues to create nodes to deal with the small incremental concept drift.

FIGURE 3.7: Showcase of ACDC's bias and variance, as well as its node evolution, on a incremental concept drift experiment: LEG Generator, with a sliding window of size 500.

centroid. Drift is introduced by adding speed to certain centroids, causing their centers to change over time.

In both benchmarks, the concepts before and after the change are characterized by a weighted, gradual, and smooth combination of two pure distributions. Different random seeds are used for the source and target streams to ensure differences in their generated distributions. The source stream consists of $20,000$ samples, with the incremental concept drift introduced after $5,000$ samples, while the target stream has $10,000$ samples, with the gradual concept drift beginning after $2,500$ samples.

As shown in Table 3.9, ACDC can efficiently handle incremental concept drifts by using its sliding window to continuously adjust the network. Even with a small sliding window, ACDC can quickly recover and present near-perfect results. In fact, Table 3.9 showcases that the sliding window is probably the main factor to be considered when detecting concept drifts using ACDC. A small sliding window, as only 25 samples, might not be sufficient to accurately detect and react to concept drifts; Hence, these experiments are the only one to present a standard deviation greater than 0.0.

Figures 3.7a and 3.7b illustrate how ACDC's bias-variance trade-off is easily shifted towards the optimal model complexity as new nodes are cautiously created. This ability to adapt to incremental concept drifts highlights the flexibility and robustness of ACDC's performance in various scenarios.

In conclusion, the experiments show that ACDC performs effectively under both sudden and incremental concept drifts. While ACDC and other baselines were initially evaluated under benchmarks with sudden concept drifts, representing more challenging scenarios, this analysis demonstrates that ACDC can also excel in handling incremental drifts, further showcasing its versatility in tackling different types of domain adaptation challenges.

3.4 Summary

The ACDC framework is a dynamic approach to unsupervised cross-domain adaptation that uses adversarial techniques to counteract data drifts. It consists of three

key components that work together to address the challenge of online unsupervised cross-domain adaptation: a denoising autoencoder as a generative feature extractor, a domain-adversarial adaptation network for adjusting to different domains, and a discriminator.

In our assessment, we compared ACDC to strong baselines utilizing the prequential test-then-train protocol. The outcomes indicate impressive generalization and target accuracy performance, with ACDC consistently outperforming the other models in nearly all experiments. In some instances, ACDC even demonstrated an improvement of more than 10%. Additionally, we explored the implications of ACDC's dynamic architecture on its space and time complexity, as well as how the internal epochs κ affect the overall classifier generalization. Generally, a higher κ value provides more opportunities for model learning but incurs a substantial time complexity cost.

Nonetheless, ACDC has some limitations. Its compatibility with GPU is not optimal, leading to reduced accuracy performance when compared to the CPU version. Moreover, ACDC relies on a fully labeled source data stream, which can be challenging or impossible to obtain for certain data stream configurations. Potential enhancements to address these concerns include the development of a fully parallel-compatible learning scheme and architecture, and the capacity to function in weakly labeled situations.

As a flexible online neural network framework, ACDC presents a variety of possibilities for future expansion. Some of the potential additions could involve integrating convolutional layers, employing distributed computing infrastructures, utilizing recurrent layers, and adjusting layer depths for greater flexibility.

To conclude, ACDC's exceptional performance in minimizing target error rate has been confirmed through rigorous experimentation using real-world data and the prequential test-then-train approach, outperforming baseline models. ACDC's adaptability and responsiveness to dynamic changes make it a promising solution for tackling complex domain adaptation issues in a variety of settings.

Chapter 4

Class-Incremental Learning via Knowledge Amalgamation

Catastrophic forgetting poses a significant challenge to the implementation of deep learning algorithms in continuous learning environments. This issue arises when an agent loses its ability to generalize from previous tasks while acquiring new knowledge. Several approaches have been suggested to tackle the problem of catastrophic forgetting [43], which impairs an agent's capacity to learn from past tasks as it takes on new challenges. The CFA approach we suggest offers a resolution to this predicament by using knowledge integration to instruct a student network through diverse teacher models, each of which has expertise in a particular prior task. This technique is adaptable to current offline methodologies.

The process of combining and integrating knowledge, i.e., knowledge amalgamation, occurs through a single-head approach, requiring only a limited number of stored samples without the need for annotations. It is crucial to note that the design of the teacher network and the student network does not have to be identical, which allows for the adaptation of diverse tasks to a more compact or sparse data representation. By using this innovative strategy, we aim to effectively address the issue of catastrophic forgetting, making it possible for agents to retain their generalization power across multiple tasks in continuous learning settings. Source-code: `github.com/Ivsucram/CFA`[1]

[1]Paper published in the Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD '22

4.1 Introduction

Deep learning algorithms have made significant strides in several computational learning systems, including natural language processing, computer vision, and clustering [27]. However, while these models have shown great promise on stationary data, they suffer from *catastrophic forgetting* when applied to dynamic settings. This phenomenon occurs when new information replaces previous experiences, resulting in a significant drop in performance on previously learned tasks.

Traditional learning systems rely on batch-setting training, where all training data is available, and tasks are known in advance. However, this approach becomes impractical when new tasks are continuously introduced, requiring a time-consuming and computationally expensive retraining process of the network parameters to adapt to changes.

In order to prevent catastrophic forgetting, it is crucial for learning agents to consistently incorporate novel data into their knowledge base without compromising their prior learning. Agents that possess the ability to continuously acquire new knowledge including new categories, are referred to as class-incremental learning systems.

A solution for class-incremental learning should possess three essential characteristics to be considered effective:

A. Firstly, in class-incremental learning, it is essential for the agent to continuously update its knowledge as new classes can be introduced at any time via a stream of data. Learning from a never-ending stream of data that introduces new classes sporadically is a crucial aspect of the process. A model with this capability ensures that it adapts to new knowledge seamlessly while preserving its existing knowledge. This feature is in contrast to batch-based models that can only learn from data available at the time of training, making them unsuitable for dynamic settings.

B. Secondly, a class-incremental learning solution must also be able to provide multi-class inference for all the classes learned at any point in time. This feature is necessary for scenarios where the agent is required to recognize multiple classes simultaneously. A model that can achieve this can be used in a range of applications, including image classification and speech recognition.

C. Lastly, an effective class-incremental learning solution should manage its computational requirements. The computational demands should be bounded or increase slowly as the number of classes learned grows. This feature is essential in practical scenarios where the computational power and memory resources are limited. A model that scales well with the number of classes learned ensures that it remains efficient and practical in real-world settings.

We present the novel catastrophic forgetting solution based on knowledge amalgamation (CFA). In this solution, multiple trained teacher models are used to *teach* a single-head student model that handles all previous tasks with no annotations, using only a selected number of memorized samples. During the process of amalgamation, this method prevents catastrophic forgetting by examining the interconnection between tasks without any form of identification. CFA can be seamlessly integrated into existing learning pipelines and does not require maintaining specific network architectures for each task.

The knowledge amalgamation technique for addressing catastrophic forgetting provides several advantages. Firstly, it enables the adaptation of diverse tasks to a unified model without requiring multiple final models. Additionally, this technique investigates task correlations during amalgamation, making it easy to integrate with current learning pipelines without additional identification procedures. This methodology is a form of post-processing continual learning, where a unique teacher model is created for each task and flexibly merged into a streamlined model for inference. This approach also eliminates the need for task-specific network architectures.

In summary, knowledge amalgamation is a promising solution for the catastrophic forgetting problem in the class-incremental learning setting. It integrates the knowledge from multiple trained teacher models into a single-head student model and maintains a bounded or slow growth in computational requirements to the number of learned classes. Basically, the process of combining knowledge can be viewed as an ongoing learning method that occurs after initial data processing. Specifically, a unique instructional model is created for each task, and these models are adaptively merged into a unified, condensed model for use during inference. Furthermore, the approach is highly flexible and can be easily incorporated into existing learning pipelines.

Contributions:

- Presenting a novel method of class-incremental learning that involves combining and integrating different forms of knowledge, known as knowledge amalgamation.
- Permitting educators and student models to showcase diverse formats and assignments.
- Developing a methodology that can seamlessly assimilate into current learning workflows, encompassing both non-continual and continual environments.

4.2 Problem Formulation

In the context of continuous learning, specifically in the scenario of learning with incremental classes, we are presented with a continuous flow of data pairs (x_i, y_i) that follow the distribution $(x_i, y_i) \overset{iid}{\sim} P_{t_i}(X, Y)$. These data pairs consist of an input x_i and a corresponding target y_i, and they are organized into a sequence of tasks denoted as $t_i \in \mathcal{T} = 1, ...T$, where the total number of tasks T is initially unknown but arrives in a controllable manner. The primary objective here is to train a predictive model $f : X \times \mathcal{T} \to Y$, which can be queried "at any point" to make predictions about the target vector "y" for a given test sample x, where $(x, y) \sim P_t$. It's worth noting that this test pair may belong to a task that has been encountered in the past or the current task.

Now, let's define the task of knowledge amalgamation in the following manner. Suppose we have N pre-trained teacher models $t_{i=1}^{N}$, each of which is specialized in handling a specific task T. Let $\mathcal{D}_i$ represent the set of classes that a particular model t_i is designed to classify. Without loss of generality, we assume that for any pair of models t_i and t_j, their sets of classes, $\mathcal{D}_i$ and $\mathcal{D}_j$, are distinct (i.e., $\mathcal{D}_i \neq \mathcal{D}j$ for all $i \neq j$). In simpler terms, we assume that each model is focused on classifying different tasks. The ultimate objective of knowledge amalgamation is to create a concise single-headed student model that has the capability to infer all tasks, essentially being able to classify all the classes within the combined set $\mathcal{D} = \cup_{i=1}^{N} \mathcal{D}_i$. To put it differently, the process of knowledge amalgamation occurs after the fact, where all teacher

Algorithm 2: CFA

Input: Teacher models T_N, Task Memory M, Student model $\mathcal{S}$, number of epochs

Output: Amalgamated Model $\mathcal{S}$

for *epoch in epochs* **do**
- M ← shuffle(M); `// Optional`
- **for** *sample m in M* **do**
  - **begin** Joint Representation Learning:
    - $f_S \leftarrow 1 \times 1conv(F_S)$; `// Student adaptation layer via 1×1 conv`
    - $\hat{f}_S \leftarrow 1 \times 1conv(3 \times 3conv(3 \times 3conv(f_S)))$; `// Student encoder via residual blocks of 1 stride`
    - $\mathcal{L}_M \leftarrow \mathcal{L}_R \leftarrow 0$; `// Loss initialization`
    - **for** T_i *in* T_N **do**
      - $f_{Ti} \leftarrow 1 \times 1(F_{Ti})$; `// Teacher adaptation layer via 1×1 conv`
      - $\hat{f}_{Ti} \leftarrow 1 \times 1conv(3 \times 3conv(3 \times 3conv(f_{Ti})$; `// Teacher encoder via residual blocks of 1 stride`
      - $\mathcal{L}_M \leftarrow \mathrm{L}_M + H(\hat{f}_S, \hat{f}_{T_i}) - H(\hat{f}_S)$; `// Eq.` 4.2
      - $\mathcal{L}_R \leftarrow \mathrm{L}_R + ||F'_{T_i}$ - $F_{T_i}||$; `// Teacher decoder, Eq.` 4.3
  - **begin** Soft Domain Adaptation:
    - $y_T \leftarrow T_N(m)$; `// Stacked teachers' soft output`
    - $D_{\mathrm{KL}_{\mathrm{soft}}} \leftarrow H(\hat{y}_S, y_T) - H(\hat{y}_S)$; `// Eq.` 4.4
  - $\mathcal{L} \leftarrow \alpha D_{\mathrm{KL}_{\mathrm{soft}}} + (1-\alpha)(\mathcal{L}_M + \mathcal{L}_R)$; `// Eq.` 4.5
  - $S_\theta \leftarrow S_\theta - \lambda \nabla \mathcal{L}$; `// Parameter learning`

models that have been trained for specific tasks are merged into a single model to perform comprehensive classification, consistent with their respective teacher models. This approach offers flexibility compared to existing continuous learning methods because a teacher model can be independently constructed for a particular task, and their knowledge can later be merged into a student model without sacrificing generalization capabilities.

4.3 Proposed method

CFA is composed of three macro stages - the join representation learning (JRL) and knowledge amalgamation (KA) - , and one micro stage - soft domain adaptation (SDA). The JRL component allows teachers and students

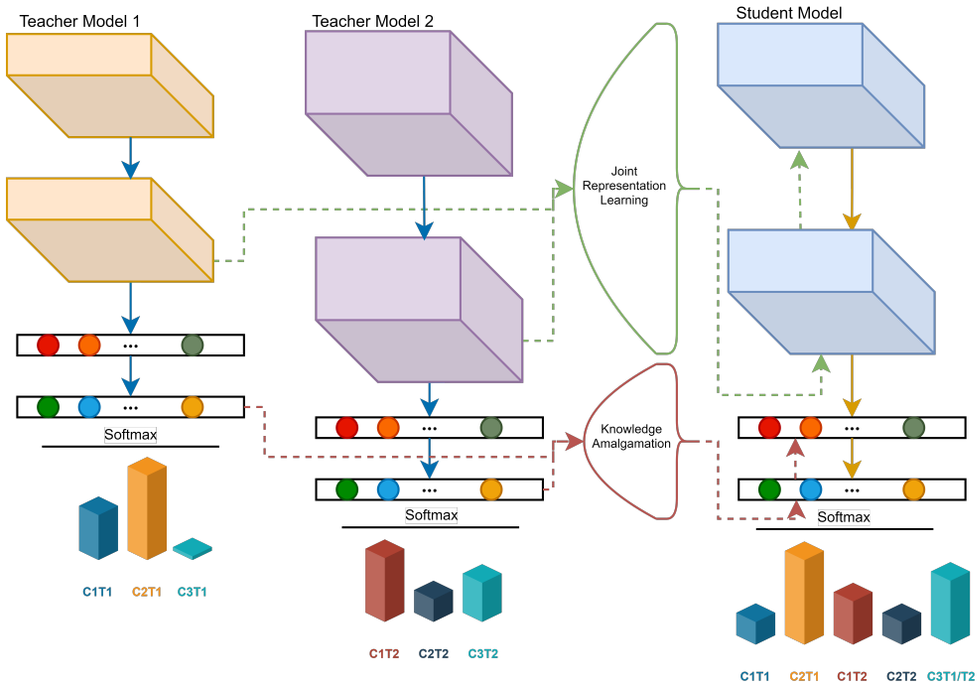


FIGURE 4.1: An overview of the proposed CFA through a summarized workflow. The CFA comprises the macro stages joint representation learning (JRL), and the knowledge amalgamation (KA). In the JRL stage, the features of both the teachers and the students (only two are shown here) are transformed into a shared space. The micro stage of between teachers and student made by KA.

to have different structures and adapt to heterogeneous tasks, while KA performs alignment between source (teachers) and target (student). The SDA stage ensures that the learned knowledge through KA is transferable across domains by aligning the source and target domains' distributions.

4.3.1 Joint Representation Learning

Figure 4.1 depicts the JRL scheme, which involves transforming the features of the teachers and students into a common feature space. In this shared space, two types of loss terms are minimized to achieve the learning objectives.

The feature ensemble loss term, denoted by $\mathcal{L}_M$, aims to align the features of the students with those of the teachers in the joint space. Meanwhile, the reconstruction loss term, denoted by $\mathcal{L}_R$, ensures that the transformed features can be accurately mapped back to the original space with minimal errors. The combined effect of these loss terms facilitates effective knowledge

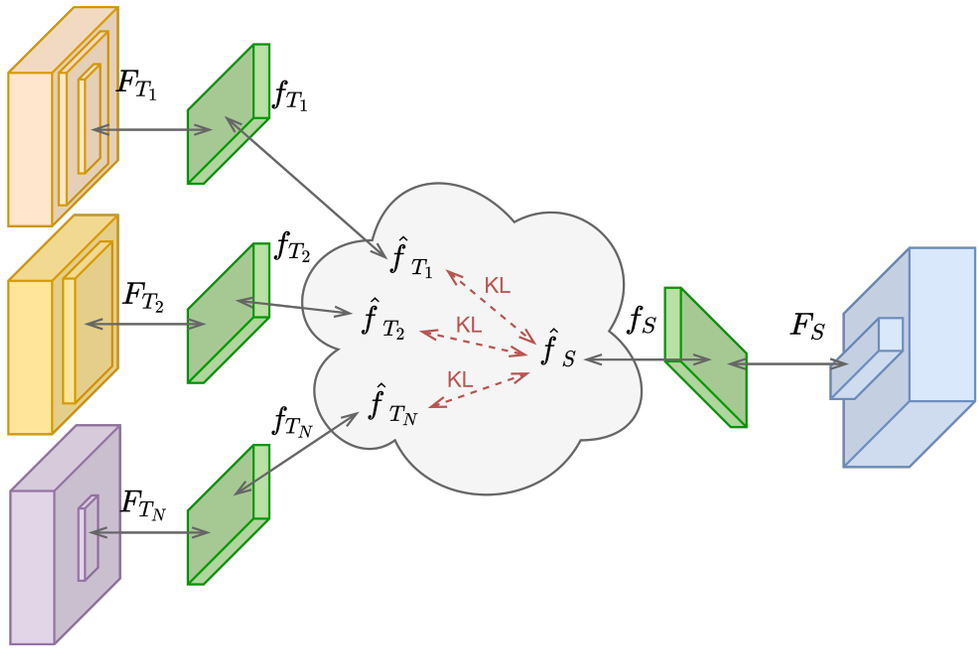


FIGURE 4.2: A depiction of CFA's shared extractor sub-network. In the joint representation learning (JRL) process, a sub-network is utilized by both the teachers and the students. The objective of this shared extractor is to generate a domain-invariant space by means of several KL operations. The resulting space is then reconstructed into the student model via decompression, i.e. a decoder is put forward to guarantee that the shared latent space is still related to the original data distribution.

transfer from the teachers to the students, enabling successful learning and the acquisition of new skills.

4.3.1.1 Adaptation Layer

The adaptation layer is responsible for aligning the output feature dimensions of the teachers and students. This alignment is achieved using a 1×1 convolution kernel [151], which can generate a predefined output length even for inputs of different sizes. More precisely, assuming F_S and F_{T_i} stand for the initial characteristics of the student and teacher T_i, correspondingly, their synchronized features are indicated by f_S and f_{T_i}.

Our approach involves setting the dimensions of f_S and f_{T_i} to match those of F_S and F_{T_i}, correspondingly. This ensures that the features of the student and teacher are appropriately aligned and that knowledge transfer can occur efficiently between the two parties.

4.3.1.2 Shared Extractor

Once the aligned features of both teachers and students have been obtained, a direct averaging method may initially appear to be a logical approach for obtaining the student's feature. Nevertheless, as a result of variances in the teacher network architectures and domain disparities in the training data, the aligned features may continue to be dissimilar. To tackle this problem, teachers and students utilize a small trainable sub-network with shared parameters, as illustrated in Figure 4.2, to achieve consistency.

This sub-network, called the shared extractor, comprises three consecutive residual blocks of 1 stride, which convert the aligned features f_{T_i} and f_S into common representation spaces denoted by $\hat{f}_{T_i}$ and $\hat{f}_S$, respectively. In our implementation, the size of $\hat{f}_{T_i}$ and $\hat{f}_S$ is half that of f_{T_i} and f_S, respectively. By sharing the extractor network, the student's features can be more accurately and effectively learned from the teachers, enabling successful knowledge transfer, as demonstrated by auto-encoder domain adaptation models [51, 152]

4.3.1.3 Knowledge Amalgamation

To combine knowledge from diverse teachers with varying attributes, we establish a feature space that remains consistent across domains for both the student and every teacher. This is achieved through the use of the KL divergence, which encourages the common feature space to remain consistent across the heterogeneous teachers:

$$D_{KL_i}(\hat{f}_S||\hat{f}_{T_i}) = H(\hat{f}_S, \hat{f}_{T_i}) - H(\hat{f}_S), \tag{4.1}$$

where $H(\hat{f}_S, \hat{f}_{T_i})$ is the cross entropy of $\hat{f}_{T_i}$ and $\hat{f}_S$ and $H(\hat{f}_S)$ is the entropy of $\hat{f}_S$.

After computing the KL divergence loss between the student's feature space and each teacher's feature space, we proceed to aggregate all the pairwise KL losses between them. Figure 4.2 depicts this aggregation process, which results in an overall discrepancy measure, denoted as $\mathcal{L}_M$, in the shared feature space.

$$\mathcal{L}_M = \sum_{i=1}^{N} D_{KLi}. \tag{4.2}$$

This discrepancy measure enables us to quantify the extent to which the student's features are aligned with those of the teachers in the shared feature space. By minimizing the $\mathcal{L}_M$ loss, our method can effectively enable knowledge transfer from the heterogeneous teachers to the student. This approach is particularly useful when dealing with diverse sources of knowledge, allowing for successful acquisition of new skills and knowledge.

In order to improve the joint representation learning between the teachers and the student, we incorporate an auto-encoder [153] reconstruction loss that operates on the original feature space of the teachers. Specifically, we define the reconstructed feature of teacher T_i as F'_{Ti} and introduce the reconstruction loss $\mathcal{L}_R$ as a means of evaluating the accuracy of the reconstruction:

$$\mathcal{L}_R = \sum_{i=1}^{N} ||F'_{Ti} - F_{Ti}||. \tag{4.3}$$

By minimizing the $\mathcal{L}_R$ loss, our method can ensure that the original feature space of the teachers is effectively preserved, enabling accurate knowledge transfer from the teachers to the student. This reconstruction approach complements the KL divergence loss, which ensures domain invariance between the teacher and student feature spaces. Together, these loss terms provide a comprehensive framework for joint representation learning, facilitating the effective transfer of knowledge across different domains and teacher characteristics.

4.3.2 Soft Domain Adaptation

In addition to familiarizing themselves with the teacher's characteristics, students are also expected to generate inferences that are identical or similar to those made by their instructors. To facilitate this process, we employ a method in which we input unlabelled samples to the teacher models and supervise the training of the student based on the output generated by the teachers.

Given the assumption that the teacher exemplifies dealing with classes that do not overlap, we can merely concatenate their scores vectors and employ them as the desired output for the student model. However, in cases where teachers have overlapping classes, we can opt to sum or average the logits of repeating classes. In contrast to traditional methods of knowledge distillation, which rely on cross-entropy loss for comparing the student model's output to the soft output of the teacher, we propose incorporating a domain invariant space into the fully connected layers of the student. This is accomplished by utilizing KL divergence to compare the student output to the soft output of the teachers, which is stacked together.

The given scenario involves utilizing the stacked teachers' soft output, denoted as y_T, and the corresponding soft output of the student, represented by $\hat{y}_S$, for implementing KL divergence in the following manner:

$$D_{KL_{soft}}(\hat{y}_S||y_T) = H(\hat{y}_S, y_T) - H(\hat{y}_S). \tag{4.4}$$

To demonstrate that such operation achieves a domain invariant latent space, we need to remember how the Kullback-Leibler (KL) divergence, also known as the relative entropy [104]. Given two probability distributions $\mathbf{P} \in \mathrm{R}^{k+1}$ and $\mathbf{Q} \in \mathrm{R}^{k+1}$, the KL divergence of $\mathbf{Q}$ from $\mathbf{P}$ is the information lost when $\mathbf{Q}$ is used to approximate $\mathbf{P}$ [154], defined as $D_{KL}(\mathbf{P}||\mathbf{Q}) = \sum_{i=1}^{k} \mathbf{P}(i) \ln \frac{\mathbf{P}(i)}{\mathbf{Q}(i)}$. As the KL divergence measures the difference between two data domains, embedding them into the same latent space and decreasing its divergence outputs a domain invariant latent space.

4.3.3 Final Loss

Our final loss function incorporates the loss terms from Eqs. 4.2, 4.3, and 4.4. Our approach involves optimizing the objective function to train the complete framework from end to end:

$$\mathcal{L} = \alpha D_{KL_{soft}}(\hat{y}_S||y_T) + (1-\alpha)(\mathcal{L}_M + \mathcal{L}_R). \tag{4.5}$$

In this context, the hyper-parameter α, which belongs to the interval $[0, 1]$, plays a crucial role in balancing the three components of the loss function.

Through the minimization of this loss function, the student network can be effectively trained, leveraging the knowledge provided by its teachers, without necessitating any annotations.

4.3.4 Theoretical Analysis

To understand how knowledge amalgamation works, we first need to understand the case of a knowledge amalgamation with a single student, which is the same as knowledge distillation [119].

4.3.4.1 Knowledge Distillation

We introduce the concept of knowledge distillation within the framework of binary classification. Let $X \subseteq \mathbb{R}^u$ represent the input space, $Y = 0, 1$ denote the label space, and $P(X)$ signify the probability density distribution of inputs.

The teacher network, denoted as $T^* : X \to Y$, and the student network, represented by $S : X \to Y$, both function as linear classifiers. The teacher model is a fixed entity described by $h^*(X) = \mathbb{1}\{\theta^T X \geq 0\}$. Both classifiers are equipped with internal parameters, referred to as weights, which are parameterized as a product of matrices: $\theta^T = \mathcal{W}_N \mathcal{W}_{N-1} \cdot \cdot \cdot \mathcal{W}_1$, where $N \geq 1$ denotes the number of layers within the model.

The knowledge distillation process commences with the collection of soft labels, often referred to as logits, from the teacher network, given by $\hat{Y} = \sigma(\theta^T X)$. Here, the inputs X are sampled independently and identically distributed (i.i.d) from $P(X)$, and σ represents a non-linear function, such as the sigmoid function $\sigma(x) = 1/(1+\exp(-x))$. These soft values can be perceived as a more informative counterpart to the traditional hard (0/1-valued) labels found in standard classification settings. Subsequently, the student network undergoes training by minimizing the cross-entropy loss function:

$$\mathcal{L}^*(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \left[\hat{Y} \log(\sigma(\theta^T X)) + (1 - \hat{Y}) \log(1 - \sigma(\theta^T X)) \right] \tag{4.6}$$

The objective is to minimize $\mathcal{L}^*$ until it reaches zero. The student network monitors this loss as a function of its parameters, specifically the individual weight matrices:

$$\mathcal{L}(\mathcal{W}_1, \cdots, \mathcal{W}_N) := \mathcal{L}^*((\mathcal{W}_N \mathcal{W}_{N-1} \cdots \mathcal{W}_1)^T) \tag{4.7}$$

The optimization process then proceeds via gradient descent.

4.3.4.2 Knowledge Amalgamation

Knowledge amalgamation can be viewed as an extension of knowledge distillation involving multiple teachers, thus allowing us to adapt the previous analyses accordingly.

The teacher models, denoted as $T^1 : X \to Y^1$, $T^2 : X \to Y^2$, ..., $T^K : X \to Y^K$, represent fixed linear classifiers defined as follows: $h^1(X) = \mathbb{1}\{\theta_1^T \geq 0\}$, $h^2(X) = \mathbb{1}\{\theta_2^T \geq 0\}$, ..., $h^K(X) = \mathbb{1}\{\theta_K^T \geq 0\}$, where $K \geq 2$ denotes the number of available teachers. Concurrently, the student model, denoted as $S : X \to (Y^1, Y^2, \cdots, Y^K)$, functions as a multi-class classifier with the capacity to handle all classes known by the teachers. Each of these classifiers possesses internal parameters referred to as weights, which are parameterized as products of matrices: $\theta_1^T = \mathcal{W}_N^1 \mathcal{W}_{N-1}^1 \cdots \mathcal{W}_1^1$, $\theta_2^T = \mathcal{W}_N^2 \mathcal{W}_{N-1}^2 \cdots \mathcal{W}_1^2$, ..., $\theta_K^T = \mathcal{W}_N^K \mathcal{W}_{N-1}^K \cdots \mathcal{W}_1^K$. The student's internal parameters are denoted as $\theta_S^T = \mathcal{W}_N^S \mathcal{W}_{N-1}^S \cdots \mathcal{W}_1^S$, with $N \geq 1$ representing the number of layers within these models.

The knowledge amalgamation process commences with the collection and concatenation of soft labels obtained from the teachers. This entails computing $\hat{Y}_1 = \sigma(\theta_1^T X)$, $\hat{Y}_2 = \sigma(\theta_2^T X)$, ..., $\hat{Y}_K = \sigma(\theta_K^T X)$, resulting in $\hat{Y} = (\hat{Y}_1, \hat{Y}_2, \cdots, \hat{Y}_K)$[2], where the input samples X are independently and identically distributed (i.i.d) according to $P(X)$. Here, σ represents a non-linear function, such as the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$.

The student model is trained by minimizing the multi-class loss function:

$$\mathcal{L}^*(\theta_S) = \sum -\hat{Y} \log(\theta_S^T X) \tag{4.8}$$

[2]Non-overlapping classes

The objective is to minimize $\mathcal{L}^*$ until it reaches zero. The student network optimizes the following observable loss via gradient descent:

$$\begin{aligned}\mathcal{L}(\mathcal{W}_1^S, \cdots, \mathcal{W}_N^S) := \mathcal{L}^*(((\mathcal{W}_N^1 \mathcal{W}_{N-1}^1 \cdots \mathcal{W}_1^1)^T, \\ (\mathcal{W}_N^2 \mathcal{W}_{N-1}^2 \cdots \mathcal{W}_1^2)^T, \\ . \\ . \\ . \\ (\mathcal{W}_N^K \mathcal{W}_{N-1}^K \cdots \mathcal{W}_1^K)^T))\end{aligned} \tag{4.9}$$

This loss is optimized through the iterative process of gradient descent.

4.4 Experiments

To assess the effectiveness of the CFA approach and its corresponding baselines, we subject them to evaluation using four different benchmarks. In addition to the benchmark evaluations, we also conduct an ablation study to gain a deeper understanding of the memory utilization and internal processes of the CFA approach.

To ensure consistency and fair comparison, we execute all experiments related to the CFA approach using identical teacher and student structures. Specifically, we employ a ResNet18 architecture [28] as the backbone feature extractor, and two fully-connected layers are incorporated in front of it. By using this standard structure for all CFA experiments, we can more accurately assess the impact of the CFA method and its corresponding baselines, as well as any observed differences or variations.

4.4.1 Replay Memory

The CFA approach utilizes the replay memory strategy to obtain logits from the teachers that are related to the original data distribution. To implement this approach, a collection of past instances is stored for future use during the merging phase. The method used to generate the replay memory was the nearest-mean-of-exemplars technique [80], although alternative strategies for selecting samples could also be utilized.

4.4.1.1 Nearest-Mean-of-Exemplars strategy

The mean exemplar for each class in class y is introduced by Rebuffi et al. [80], and computed as $\mu_y = \frac{1}{||\mathcal{D}i||} \sum_{x \in \mathcal{D}_i} t_i(x)$, where $t_i(x)$ represents the logits generated by a teacher t_i for a specific task i. This calculation allows us to obtain a representative exemplar for each class that encapsulates the knowledge of all teachers for that specific class.

To ensure that this exemplar is accurate, the CFA approach employs a sample selection strategy for the replay memory. Specifically, a sample x is added to the memory if there is available space. If the memory is full, the sample x and the existing samples in memory are sorted in descending order based on their L_2 distance. The sample with the farthest distance from the existing samples is then replaced with x, ensuring that the memory is populated with the most informative and relevant samples. By utilizing this sample selection strategy, the CFA approach can achieve better performance in the amalgamation process, and the resulting exemplars can more accurately represent the knowledge of all teachers.

4.4.2 Baselines setup

In our experiments, we utilized a Windows machine with an Intel Core i9-9900K 5.0 GHz processor, 32GB of main memory, and an Nvidia GeForce 2080 Ti to evaluate all methods. To ensure consistency and a fair comparison, all teachers and students had the same architecture, consisting of a pre-trained ResNet18 feature extractor [28] followed by two fully-connected layers with dimensions of $\mathbb{R}^{1000 \times 500}$ and $\mathbb{R}^{500 \times \text{output}}$, respectively. During optimization, we used the Adam optimizer with a learning rate of $\lambda = 10^{-4}$, a hyper-parameter value of $\alpha = 0.5$, and a training period of 100 epochs.

In the CFA approach, we evaluated two different configurations: $\text{CFA}_{\text{fixed}}$ and CFA_{grow}. $\text{CFA}_{\text{fixed}}$ utilized the nearest-mean-of-exemplars replay memory strategy with a fixed memory footprint of 1000 samples, while CFA_{grow} employed the teachers' confidence replay memory strategy (i.e., similar to the nearest-mean-of-exemplars introduced by [80], but where the confidence rate is used instead of the mean to the exemplars of a class), which allows for a growing memory size of up to 1000 samples per task. As a result, the

memory footprint of $\text{CFA}_{\text{fixed}}$ remained the same regardless of the number of classes learned so far[3], while CFA_{grow} had a memory footprint that slowly increased as more classes were introduced.

In addition to the CFA approach, we also evaluated other baselines in our experiments based on the source-code release by [82]. Like CFA_{grow}, these baselines had a memory budget of 1000 samples per task, making them comparable in memory usage.

4.4.3 Benchmarks

To assess the efficacy and performance of the CFA approach and its corresponding baselines in various continual learning scenarios, our experiments utilized four different benchmarks. Each benchmark features an incremental class problem, and the datasets are divided into sequential tasks that gradually increase in complexity.

The first benchmark, known as *SplitMNIST*, is a standard continual learning benchmark that transforms the original MNIST problem [142] into five sequential tasks, comprising a total of ten classes. In this benchmark, the aim is to maintain high accuracy across all tasks, while also ensuring that new tasks do not cause catastrophic forgetting of previously learned tasks.

The second benchmark, *SplitCIFAR10*, is also an incremental class problem that divides the full CIFAR10 problem [143] into five sequential tasks, each consisting of ten classes. The aim in this benchmark is similar to that of *SplitMNIST*, which is to achieve high accuracy while preventing catastrophic forgetting.

The third benchmark, *SplitCIFAR100*, also features an incremental class problem and divides the entire CIFAR100 problem [143] into ten sequential subsets, totaling one hundred classes. In this benchmark, the difficulty level gradually increases as more classes are introduced, making it more challenging to maintain high accuracy across all tasks.

Finally, the *SplitTinyImageNet* benchmark also features the incremental class problem. However, instead of using the full ImageNet dataset, only 200

[3]It should be noted that storage of the original teacher models parameters is still required, typically in secondary memory such as an HDD or SSD.

Baseline	Metric (%)	*SplitMNIST*	*SplitCIFAR10*	*SplitCIFAR100*	*SplitTinyImageNet*
EWCo [87, 88]	ACC	19.13 ± 0.02	18.57 ± 2.04	7.91 ± 0.79	7.39 ± 0.03
	BWT	-97.37 ± 0.26	-88.51 ± 5.18	-83.80 ± 1.56	-71.79 ± 0.62
	FWT	-13.36 ± 1.38	-10.09 ± 5.64	-1.02 ± 0.15	-0.44 ± 0.16
LWF [4]	ACC	19.20 ± 0.06	16.27 ± 4.55	9.13 ± 0.43	0.41 ± 0.20
	BWT	-96.52 ± 0.94	-89.24 ± 9.60	-83.34 ± 5.57	-50.33 ± 3.25
	FWT	-11.92 ± 2.01	-11.05 ± 1.88	0.16 ± 0.71	-0.25 ± 0.03
ER [155]	ACC	23.41 ± 0.60	69.07 ± 3.31	27.41 ± 2.94	12.33 ± 1.23
	BWT	-93.83 ± 0.92	-24.15 ± 14.17	-66.35 ± 1.22	-71.07 ± 2.35
	FWT	-8.83 ± 3.14	-11.84 ± 0.40	-0.98 ± 0.08	-0.5 ± 0.05
AGEM [81]	ACC	9.19 ± 0.65	13.49 ± 4.12	0.94 ± 0.32	1.55 ± 0.32
	BWT	-40.52 ± 46.02	-47.26 ± -47.26	2.99 ± 5.74	-14.93 ± 2.12
	FWT	-8.97 ± 3.54	-6.06 ± -6.06	0.74 ± 1.74	-0.55 ± 0.20
DER [82]	ACC	60.85 ± 2.87	72.99 ± 6.43	**32.60 ± 9.77**	23.62 ± 3.30
	BWT	-42.80 ± 3.81	-22.71 ± 5.00	-44.64 ± 8.54	-52.19 ± 3.66
	FWT	-12.25 ± 2.71	-9.36 ± 8.93	-0.93 ± 0.09	-0.46 ± 2.12
DER++ [82]	ACC	72.86 ± 0.95	**77.86 ± 7.59**	**38.82 ± 8.28**	23.94 ± 2.52
	BWT	-24.64 ± 1.21	-16.27 ± 5.71	-49.03 ± 7.60	-43.82 ± 5.95
	FWT	-12.59 ± 0.48	-6.26 ± 8.81	-0.91 ± 0.07	-0.26 ± 2.16
FDR [156]	ACC	78.08 ± 3.41	48.00 ± 5.36	**32.26 ± 5.51**	13.30 ± 1.64
	BWT	-21.73 ± 4.36	-86.58 ± 4.37	-62.87 ± 5.83	-67.08 ± 1.69
	FWT	-10.10 ± 1.13	-11.41 ± 2.95	-0.87 ± 7.29	-0.67 ± 0.22
GSS [157]	ACC	24.69 ± 0.80	43.96 ± 2.86	13.94 ± 0.30	9.60 ± 0.84
	BWT	-91.69 ± 1.04	-55.71 ± 2.57	-78.21 ± 0.32	-69.36 ± 0.28
	FWT	-10.31 ± 1.96	-10.56 ± 3.30	-0.39 ± 0.55	-0.53 ± 0.05
HAL [158]	ACC	**88.25 ± 0.46**	50.11 ± 1.18	11.00 ± 2.87	3.23 ± 0.11
	BWT	-13.61 ± 0.62	-47.01 ± 2.14	-44.74 ± 1.84	-32.68 ± 4.10
	FWT	-8.81 ± 3.28	-11.70 ± 1.69	-0.97 ± 0.26	-0.24 ± 0.31
CFA_{fixed} (Ours)	ACC	83.51 ± 1.35	74.96 ± 0.46	27.76 ± 2.28	23.44 ± 2.55
	BWT	-7.95 ± 1.53	-14.25 ± 1.76	-16.41 ± 1.49	-17.58 ± 1.89
	FWT	69.46 ± 9.41	54.28 ± 6.57	26.91 ± 3.17	20.49 ± 5.50
CFA_{grow} (Ours)	ACC	**89.25 ± 3.66**	**79.40 ± 1.15**	**38.74 ± 3.26**	**32.50 ± 3.35**
	BWT	69.77 ± 1.31	49.00 ± 2.78	11.49 ± 2.65	23.33 ± 3.45
	FWT	91.77 ± 5.18	65.67 ± 8.22	21.84 ± 4.26	32.58 ± 5.12

TABLE 4.1: A comparison of average accuracy rate (ACC), backward transfer (BWT) and forward transfer (FWT) for each experiment. Specifically, we conducted five executions of each experiment and present the results here, in terms of mean and standard deviation. The best result is each experiment is highlighted with bold font.

classes were selected and resized to 64×64 colored pixels. The dataset was then divided into ten sequential tasks, with the aim being to maintain high accuracy across all tasks while also handling the increased complexity caused by the larger dataset size.

4.4.4 Numerical results

To evaluate the performance of CFAfixed and CFAgrow in comparison to existing approaches, we compared them against one regularization-based approach (EWCo), one knowledge distillation approach (LWF), and six memory-based approaches (AGEM, DER, DER++, FDR, GSS, HAL) [81, 82, 156–158].

The results of our study, which are presented in Table 4.1, show that $\text{CFA}_{\text{fixed}}$ and CFA_{grow} exhibit comparable or even superior performance in comparison to existing state-of-the-art methods. It is noteworthy that this is particularly significant when taking into account the fact that $\text{CFA}_{\text{fixed}}$ has a fixed memory footprint. Both $\text{CFA}_{\text{fixed}}$ and CFA_{grow} achieved impressive BWT and FWT metrics, with CFA_{grow} being the only model to achieve positive values across all metrics.

These results suggest that CFA has the potential to enable zero-shot learning [45], even though the study did not specifically target this. Zero-shot learning refers to the ability to recognize novel objects, classes, or attributes without requiring any training samples [159]. The fact that CFA exhibits characteristics of zero-shot learning may indicate that it has the ability to generalize well to unseen tasks or classes.

The strong performance of CFA can be attributed to its utilization of pre-existing knowledge from individual teachers trained on specific tasks. This approach enables CFA to overcome the problem of catastrophic forgetting, while also reducing the need for continual retraining. Moreover, the flexibility of CFA allows for its integration into existing learning pipelines, making it a useful tool for organizations looking to adopt incremental learning techniques. Overall, our study demonstrates the effectiveness of the CFA approach in continual learning scenarios and highlights its potential for broader applications in the field of machine learning.

4.4.5 Ablation study

4.4.5.1 Memory Analysis

Table 4.2 offers an insightful comparison of the strongest baselines, showing how their accuracies vary across different memory budgets. Notably, our

Benchmark	Memory budget	CFA_{fixed}	CFA_{grow}	DER	DER++	FDR
SplitCIFAR10	100	48.87 ± 5.26	**61.36 ± 2.02**	46.77 ± 3.12	51.91 ± 4.21	39.60 ± 4.54
	200	61.20 ± 4.35	**69.33 ± 1.54**	58.41 ± 3.23	64.92 ± 6.15	44.49 ± 4.31
	500	69.53 ± 3.30	**74.63 ± 1.20**	65.63 ± 5.95	**72.45 ± 6.85**	48.20 ± 5.30
	1000	74.96 ± 0.46	**79.40 ± 1.15**	72.99 ± 6.43	**77.86 ± 7.59**	41.91 ± 6.42
	2000	76.45 ± 1.90	**82.21 ± 2.52**	73.81 ± 5.12	77.44 ± 8.90	47.39 ± 7.01
SplitCIFAR100	100	7.75 ± 1.20	**21.34 ± 2.30**	13.23 ± 0.00	**22.88 ± 4.90**	12.23 ± 4.50
	200	12.87 ± 2.12	**26.01 ± 2.53**	19.98 ± 0.00	23.78 ± 5.20	14.74 ± 2.24
	500	21.23 ± 2.23	**30.59 ± 3.54**	26.53 ± 5.23	**31.45 ± 6.43**	22.26 ± 4.21
	1000	27.76 ± 2.28	**38.74 ± 3.26**	32.60 ± 9.77	**38.82 ± 8.28**	32.26 ± 5.51
	2000	41.50 ± 3.45	**47.54 ± 3.18**	36.78 ± 9.88	43.45 ± 8.54	33.12 ± 5.40

TABLE 4.2: The average accuracy metrics expressed as a percentage over varying memory budgets are displayed.

analysis demonstrates that CFA_{fixed} maintains a competitive performance despite presenting a fixed memory footprint. In fact, it performs comparably to other strong baselines such as DER, DER++, and FDR while utilizing less memory. Additionally, CFA_{fixed} can be easily incorporated into existing offline learning pipelines, making it a practical and cost-effective solution for organizations looking to adopt incremental learning techniques.

However, it is important to note that while CFA has many strengths, it also has one significant weakness. The approach requires secondary memory to record previous teacher models. In practice, this means that organizations must allocate storage resources, such as hard disk drives (HDD) or solid-state drives (SSD), to store the original teacher models. Although this may not be a significant issue for some organizations, it can be a concern for those with limited storage capacity or strict data privacy regulations.

Despite this limitation, the flexibility and effectiveness of the CFA approach make it a promising tool for organizations seeking to adopt incremental learning techniques. By utilizing pre-existing knowledge from individual teachers trained on specific tasks, CFA can overcome the problem of catastrophic forgetting and reduce the need for continual retraining. Additionally, the fixed memory footprint of CFA_{fixed} provides a practical and cost-effective solution for resource-constrained organizations. Overall, our study highlights the potential of the CFA approach to revolutionize the field of machine learning by enabling continual learning in a practical and efficient manner.

Description	*Split CIFAR10*
$\alpha = 1.0$ — JRL(×)SDA(✓)	35.78 ± 10.78
$\alpha = 0.5$ — JRL(✓)SDA(✓)	74.96 ± 0.46
$\alpha = 0.0$ — JRL(✓)SDA(×)	69.68 ± 2.23

TABLE 4.3: The average accuracy (in percentage) outcomes with different hyper-parameter values for α are shown for the $\text{CFA}_{\text{fixed}}$ model with a memory budget of 1000. The impact of Joint Representation Learning (JRL) and Soft Domain Adaptation (SDA) on the primary loss function is modulated to evaluate the performance of the model.

4.4.5.2 Joint representation learning analysis

Table 4.3 provides a detailed analysis of the impact of hyper-parameter α on the performance of CFA. According to our findings, joint representation learning (JRL) plays a crucial role in guiding the student's latent space to accurately represent various tasks. If JRL is deactivated, the model experiences difficulty in learning high-level feature representations through the soft domain adaptation (SDA) method, leading to significant catastrophic forgetting.

Moreover, our examination shows that selecting α is particularly crucial when working with complete diverse arrangements, supported by Luo et al. [121]. Our research employs a uniform framework for both teachers and students models, resulting in the reduced importance of the distinction between $\alpha = 0.0$ and $\alpha = 0.5$.

Overall, these findings demonstrate the importance of the JRL in driving the adaptation of the student's latent space and the significance of the choice of α in achieving optimal performance. By incorporating both the JRL and appropriate values of α to control SDA, CFA can effectively address the problem of catastrophic forgetting, enabling efficient continual learning.

4.5 Summary

CFA is a novel approach to mitigate catastrophic forgetting in class-incremental learning scenarios. The issue is addressed by combining the expertise of various teacher models, each trained on a prior undertaking, into a solitary

student model with a single head capable of managing all the tasks. We assessed the efficacy of CFA by contrasting it with a collection of competitive baselines in class-incremental learning situations. Our experimental findings proved that CFA accomplished favorable generalization, displaying remarkable average accuracy and knowledge transfer capabilities, as demonstrated by its elevated backward and forward knowledge transfer metrics.

One of the unique features of CFA is its ability to handle multiple classes simultaneously, making it an attractive option for real-world applications where a large number of classes need to be handled. Moreover, CFA demonstrated some zero-shot learning abilities, which indicate that it can learn from classes it has never seen before. This is a significant advantage over traditional methods that require labeled data for every class.

However, CFA does have some limitations. One of the main limitations of CFA is the requirement for secondary memory to record the previous teacher models, which can be a challenge in resource-constrained environments. Additionally, while our evaluation demonstrated the effectiveness of CFA in comparison to current state-of-the-art methods, further studies are necessary to determine its generalizability across a wider range of datasets and tasks.

Despite these limitations, CFA presents a unique and innovative approach to continual learning through knowledge amalgamation, allowing for the creation of a single-headed student model that can handle all tasks. By incorporating both the JRL and appropriate values of α to control SDA, CFA can effectively address the problem of catastrophic forgetting, enabling efficient continual learning. The promising results obtained by CFA in our evaluation suggest that knowledge amalgamation approaches such as CFA can offer a significant contribution to the field of continual learning and inspire further research in this direction.

Chapter 5

Towards Cross-Domain Continual Learning

The ultimate goal of continual learning is to develop learning agents that can sequentially handle a sequence of tasks/classes without revisiting past data. This requires the agent to utilize previously learned experiences to better and faster learn new tasks while simultaneously preventing catastrophic forgetting. Unfortunately, existing methods have been limited to handling only a single domain, thereby restricting their applicability to specific problems.

To address this limitation, we propose an innovative approach that incorporates an inter- and intra-task cross-attention mechanism into a compact convolution network. Our intuition is that this mechanism implicitly maintains previous task domain feature alignments, thereby delaying the data drift of important attention scores between the tasks. To ensure accurate input pairs between labeled and unlabeled samples, we employ an intra-task-specific pseudo-label method. The proposed framework represents a significant step toward Cross-Domain Continual Learning (CDCL), as it can address the unsupervised domain adaptation (UDA) problem in a task-incremental learning setup.

Our experimental analysis demonstrates that the proposed method performs well on cross-domain continual learning problems under public UDA datasets. Moreover, the results highlight incremental ideas that could be applied to the field, such as the use of a compact convolution network and an intra-task-specific pseudo-label method. Our approach is also suitable for applications

where limited computational resources are available, as it does not require a large neural network architecture. We made the source-code public available at `github.com/Ivsucram/CLCD`[1]

5.1 Introduction

The area of UDA is well-explored [99, 111–114], but most existing methods focus solely on single-domain Continual Learning (CL) [45, 81, 82, 158, 160], which is insufficient to capture all the available knowledge in the world. These methods are not equipped to handle changes in the task definition due to drifts in the conditional distributions of the given labels (*task drift*) as well as changes in the marginal distributions of the given domains (*domain drift*). Hence, single-domain CL methods are prone to *feature-alignment catastrophic forgetting* when applied to multi-domain problems because not only are past experiences being overwritten, but their domain invariance acquaintance is lost.

We propose the novel Cross-Domain Continual Learning (CDCL) framework. CDCL puts the cross-attention mechanism into the domain-adaptation CL setup by introducing a *inter- intra-task cross-attention* mechanism. Within continual configurations, the latter enables a category-level feature alignment between a labeled source domain and an unlabeled target domain. Moreover, CDCL proposes an *intra-task center-aware pseudo-label* procedure, which arranges similar source and target domain samples with reduced noise. Furthermore, CDCL's rehearsal memory records not only previous source and target data samples but previously outputted logits, forcing the model to mimic acquired knowledge when learning new tasks.

To address these limitations, we propose the novel Cross-Domain Continual Learning (CDCL) framework. CDCL introduces an *inter- intra-task cross-attention* mechanism into the domain-adaptation CL setup. The mechanism is a transformer-based approach that maintains category-level feature alignment between a labeled source domain and an unlabeled target domain.

[1] Paper submited to the Proceedings of the IEEE International Conference on Data Engineering, IEEE ICDE '24

Moreover, CDCL incorporates an *intra-task center-aware pseudo-labeling* procedure, which arranges similar source and target domain samples with reduced noise. The process of *intra-task center-aware pseudo-labeling* ensures a trade-off between exploration and exploitation by offering the model an adequate amount of diverse training data.

CDCL is a significant step toward addressing the cross-domain continual learning problem, being able to solve the UDA problem in the task-incremental learning setup.

We present a three-fold contribution in this work

- A new framework named Cross-Domain Continual Learning (CDCL) that addresses the unsupervised cross-domain (UDA) task-incremental learning problem while preserving a balanced stability-plasticity trade-off.
- An *inter- intra-task cross-attention* mechanism that aligns domain-specific features in current tasks while consolidating previously acquired alignment knowledge. This approach mitigates the feature-alignment catastrophic forgetting of related knowledge domains.
- An *intra-task-based center-aware pseudo-labeling* method to identify similar task-specific samples between domains and decrease the noise.

5.2 Problem formulation

Consider an incremental data flow denoted by $(\mathcal{D}_{\mathcal{S}_i}, \mathcal{D}_{\mathcal{T}_i})$, where the data pairs $(X_{\mathcal{S}_i}, Y_{\mathcal{S}_i})$ follow the distribution $(X_{\mathcal{S}_i}, Y_{\mathcal{S}_i}) \overset{iid}{\sim} \mathcal{D}_{\mathcal{S}_i}(X_{\mathcal{S}}, Y_{\mathcal{S}})$, and the unlabeled inputs $X_{\mathcal{T}_i}$ follow the distribution $(X_{\mathcal{T}_i}) \overset{iid}{\sim} \mathcal{D}_{\mathcal{T}_i}(X_{\mathrm{T}})$. These data are organized into a series of sequential tasks denoted as $t_i \in \mathcal{T} = 1, \cdots, T$, where the total number of tasks T is initially unknown. The primary objective here is to train a predictive model $f : (X_{\mathcal{S}} \cup X_{\mathcal{T}}) \times \mathcal{T} \to Y_{\mathcal{T}}$, which can be queried "at any point" to predict the target vector $y_{\mathcal{T}t}$ associated with a sample from the target domain $x_{\mathcal{T}t}$, where $(x_{\mathcal{T}t}, y_{\mathcal{T}t}) \sim P_t$.

Solving this problem is not straightforward due to the challenges of dealing with "limited labeled data in the target domain." Additionally, the model must handle "potential shifts in tasks," where the distributions $P_t(X_{\mathcal{S}}, Y_{\mathcal{S}})$

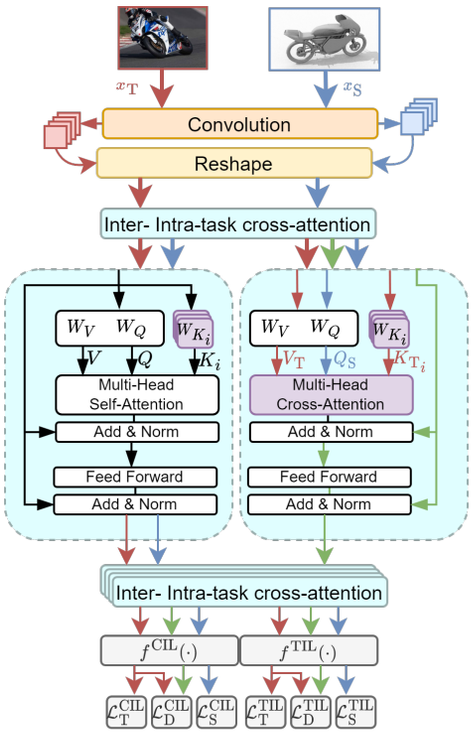


FIGURE 5.1: The proposed framework. The main contribution is highlighted with purple: The inter- intra-task cross attention, responsible for aligning the source and target feature domains and mitigating its catastrophic forgetting when new tasks arrive. The first inter- intra-task cross-attention block propagates the target attention signal, as no cross-attention signal is available. $\mathbf{b}_i$ is omitted for simplicity. The $f^{\text{CIL}}(\cdot)$ is a single-head output used for class incremental learning (CIL) scenarios along with the latest $\mathbf{K}_T$ and $\mathbf{b}_T$ instantiated. Meanwhile, the $f^{\text{TIL}}(\cdot)$ is a multi-head output used for task incremental learning (TIL) scenarios with the respective $\mathbf{K}_i$ and $\mathbf{b}_i$, as the task-identifier t_i is provided.

are not equal to $P_{t+1}(X_\mathcal{S}, Y_\mathcal{S})$ and $P_t(X_\mathcal{T}, Y_\mathcal{T})$ are not equal to $P_{t+1}(X_\mathcal{T}, Y_\mathcal{T})$, and "changes in data domains" where $P_t(X_\mathcal{S})$ is not equal to $P_t(X_\mathcal{T})$ and $P_t(Y_\mathcal{S}|X_\mathcal{S}) = P_t(Y_\mathcal{T}|X_\mathcal{T})$ during the training process.

5.3 Proposed method

The Cross-Domain Continual Learning (CDCL) framework effectively mitigates the catastrophic forgetting of domain-invariant features. An overview of the framework is presented in Figure 5.1. The major contributions of the CDCL framework are as follows:

1) *Inter- intra-task cross-attention*: This is an incremental transformer block that implicitly retains information from previous tasks into the attention

maps. The cross-attention mechanism aligns domain features in current tasks and consolidates previous alignment knowledge, alleviating the feature-alignment catastrophic forgetting of related knowledge domains.

2) *Intra-task center-aware pseudo-labeling*: This component generates synthetic labels for the incoming unlabeled target inputs based on the intra-task information, integrating the same into the inter- intra-task cross-attention maps. It identifies similar task-specific samples between domains, resulting in accurate input pairs between labeled and unlabeled samples. The proposed center-aware pseudo-labeling method reduces the noise in similar source and target domain samples.

5.3.1 Inter- intra-task cross-attention mechanism

The proposed cross-attention technique is founded upon the Compact Convolutional Transformer (CCT) [161] sequential pooling approach, a method that employs attention to perform pooling over the convolutional tokenizer. Consider an image $x_i \in \mathbb{R}^{H\times W\times C}$ that is a part of task t_i:

$$x_{ct} = \text{MaxPool}(\text{ReLU}(\text{Conv2d}(x_i))) \tag{5.1}$$

In this case, Conv2D is equipped with d filters, which matches the number of dimensions in the transformer backbone's embedding—thus preserving local spatial details. The resulting convolution, x_{ct}, takes the place of patch tokens in the Vision Transformer (ViT) [162]. Subsequently, x_{ct} is projected into global query vectors $\mathbf{Q} \in \mathbb{R}^{n\times d}$ and global value vectors $\mathbf{V} \in \mathbb{R}^{n\times d}$. Additionally, x_{ct} is also projected into task-specific key vectors $\mathbf{K}_i \in \mathbb{R}^{n\times d}$ and a task-specific bias vector $\mathbf{b}_i \in \mathbb{R}^{1\times n}$.

The self-attention mechanism is subsequently employed:

$$x_{Li} = \frac{\mathbf{Q}\mathbf{K}_i^T + \mathbf{b}_i}{\sqrt{d}}\mathbf{V} \tag{5.2}$$

Here, x_{L_i} represents the output of a transformer encoder consisting of L layers, and d denotes the overall embedding dimension. The cross-attentionn component is derived from the self-attention module [163]. The key distinction lies in the input for the cross-attention mechanism, which consists of

image pairs from the source and target domains, specifically x_{S_i} and x_{T_i}[2]. The computation for the cross-attention component is as follows:

$$x_{Li} = \frac{\mathbf{Q}_S \mathbf{K}_i^T + \mathbf{b}_i}{\sqrt{d}} \mathbf{V}_\mathrm{T} \tag{5.3}$$

Here, $\mathbf{Q}_S$ refers to the global queries derived from image x_{S_i}, and $\mathbf{V}_T$ signifies the global values originating from image x_{T_i}. It is essential to note that the convolution token of x_{T_i} is also projected into task-specific $\mathbf{K}_i$ and $\mathbf{b}_i$, resulting in an inter-task projection. A new pair, $\mathbf{K}_{i+1}$ and $\mathbf{b}_{i+1}$, is generated when a new task emerges, while previously learned $\mathbf{K}_{1,\cdots,i}$ and $\mathbf{b}_{1,\cdots,i}$ remain fixed, conserving the knowledge from prior feature-aligned tasks and constructing the current feature-aligned knowledge based on the global $\mathbf{Q}$ and $\mathbf{V}$.

The attention maps are utilized to produce an importance weighting for each input convolution token:

$$x'_{L_i} = \mathrm{softmax}(g(x_{L_i})^T) \tag{5.4}$$

$$z_i = x'_{L_i} x_{L_i} = \mathrm{softmax}(g(x_{L_i})^T) \times x_{L_i} \tag{5.5}$$

In the subsequent portions of the chapter, this entire procedure is streamlined using the following notation:

$$z_i = a(x_i) \tag{5.6}$$

The ultimate feature output is produced by flattening $z_i \in \mathbb{R}^{b \times d}$. These features can then be passed through a classifier for both training and assessment. CDCL examines z_i in terms of both inter-task (CIL) and intra-task (TIL) approaches.

$$\hat{y}^{\mathrm{TIL}} = f^{\mathrm{TIL}}(z_i) \tag{5.7}$$

$$\hat{y}^{\mathrm{CIL}} = f^{\mathrm{CIL}}(z_i) \tag{5.8}$$

[2]Such input is omitted from most further notations, but it should be clear to readers that when the attention mechanism receives two inputs, it is indeed the cross-attention mechanism that is being displayed

From here, the following objective functions are applied:

$$\mathcal{L}_S^{\text{CIL}} = -\sum y_S \log(\hat{y}_S^{\text{CIL}}) \tag{5.9}$$

$$\mathcal{L}_T^{\text{CIL}} = -\sum y_S \log(\hat{y}_T^{\text{CIL}}) \tag{5.10}$$

$$\mathcal{L}_D^{\text{CIL}} = \sum \hat{y}_{S+T}^{\text{CIL}} \log(\hat{y}_T^{\text{CIL}}) \tag{5.11}$$

In this context, y_S represents the label vector from the source domain, while $\hat{y}_S^{\text{CIL}}$ and $\hat{y}_T^{\text{CIL}}$ are the output vectors of the source input and target input, respectively, as processed by the intra-task classifier. Additionally, $\hat{y}_{S+T}^{\text{CIL}}$ denotes the output vector of the inter-intra-task cross-attention applied to both the source and target image vectors via the intra-task classifier. Analogous objectives are optimized using the inter-task classifier:

$$\mathcal{L}_S^{\text{TIL}} = -\sum y_S \log(\hat{y}_S^{\text{TIL}}) \tag{5.12}$$

$$\mathcal{L}_T^{\text{TIL}} = -\sum y_S \log(\hat{y}_T^{\text{TIL}}) \tag{5.13}$$

$$\mathcal{L}_D^{\text{TIL}} = \sum \hat{y}_{S+T}^{\text{TIL}} \log(\hat{y}_T^{\text{TIL}}) \tag{5.14}$$

We will refer to the inter-task and intra-task losses as a single objective throughout the chapter:

$$\mathcal{L}^{\text{CIL}} = \mathcal{L}_S^{\text{CIL}} + \mathcal{L}_T^{\text{CIL}} + \mathcal{L}_D^{\text{CIL}} \tag{5.15}$$

$$\mathcal{L}^{\text{TIL}} = \mathcal{L}_S^{\text{TIL}} + \mathcal{L}_T^{\text{TIL}} + \mathcal{L}_D^{\text{TIL}} \tag{5.16}$$

5.3.2 Intra-task center-aware pseudo-label

We construct a set $\mathbb{P}$ that comprises pairs of similar inputs from both the source and target domains to guarantee feature alignment on samples that

look similar. The formation of $\mathbb{P}$ takes into account both feature and label similarities. We extend the work of Liang et al. [132] to enhance inter-task alignment by incorporating intra-task knowledge into their center-aware pseudo-label mechanism. Upon the arrival of a new task, a warm-up phase commences in which the classifiers train exclusively on source domain inputs. Once the warm-up stage concludes, we establish category centroids in the target domain via weighted *k-means* clustering, based on intra-task classification information and updating it with every training epoch[3]:

$$\mathbf{c}_k = \frac{\sum_{\mathcal{D}_{T_i}} \hat{y}_T^{\text{TIL}} a(x_T)}{\sum_{\mathcal{D}_{T_i}} \hat{y}_T^{\text{TIL}}} \tag{5.17}$$

In this case, $\hat{y}_T^{\text{TIL}}$ corresponds to the intra-task prediction on target input samples, while k denotes the number of classes present in the current task. Target data pseudo-labels are generated using the nearest neighbor classifier:

$$\hat{y}_T = \arg\min_k d(\mathbf{c}_k, a(x_T)) \tag{5.18}$$

with $d(\,\cdot\,,\,\cdot\,)$ denoting a distance metric, such as cosine similarity or Euclidean distance.

Finally, the set $\mathbb{P}$ is updated based on the pseudo-labels, eliminating noise and solely permitting paired instances that align the source input label with the target-generated pseudo-label:

$$\mathbb{P} = \{(x_S, y_S, x_T) | \forall x_S \in a(X_S), \forall x_T \in a(X_T), \arg\min_{(x_S, y_S, x_T)} d(x_S, x_T) \wedge y_S = \hat{y}_T\} \tag{5.19}$$

5.3.3 Sample rehearsal

Even though neural networks remain vulnerable to catastrophic forgetting, it is postulated that remnants of prior learning experiences persist within the network's weights. As such, the practice of rehearsing samples serves as an efficient method for guiding these remnants back towards the representation of earlier knowledge. Nevertheless, the memory capacity is limited, either due

[3]We reconstruct the centroids at each training epoch.

to a fixed size or an exceedingly slow expansion rate in relation to the influx of new tasks. Consequently, it becomes crucial to select samples that more accurately reflect the previously acquired feature alignment knowledge, thus facilitating superior regeneration of the remnants. Upon completion of the current task t_i training, $\lfloor |M|/t_i \rfloor$ records[4] possessing the greatest intra-task confidence, given by $max(\hat{Y}_S^{TIL})$, or $\max(\hat{Y}_T^{TIL})$ if drawn, are retained in the memory M.

In order to retain information about earlier tasks, our objective is to minimize the subsequent loss functions:

$$\mathcal{L}_R^{ST} = -\sum y^R \log(f^{\text{CIL}}(a(x_S^R)) f^{\text{CIL}}(a(x_T^R))) \tag{5.20}$$

$$\mathcal{L}_R^{D} = \sum f^{\text{CIL}}(a(x_S^R, x_T^R)) f^{\text{CIL}}(a(x_T^R)) \tag{5.21}$$

In these equations, x_S^R, x_T^R, y^R denote the source image, target image, and the true source labels documented in the memory M, respectively. The intuition here is that, while learning a new task, the inter-task connections (CIL) of the model should try to preserve previous intra-task behaviors (TIL). This is done to forcing the model to replay previous intra-task (TIL) representations recorded in memory through the inter-task parameters (CIL). Such equations can be seen as KL divergences between different sets of previously learned distributions.

Simultaneously, CDCL strives to safeguard the demarcations of previously learned tasks through the utilization of intra-task logit replay:

$$\mathcal{L}_R^{Z} = \sum y_S^R \log(\frac{\hat{y}_R^R}{f^{\text{CIL}}(a(x_T^R))} \frac{\hat{y}_S^R}{f^{\text{CIL}}(a(x_S^R))}) \tag{5.22}$$

where $\hat{y}_S^R, \hat{y}_T^R$ correspond to the source logits and target logits archived in memory M.

Ultimately, we combine the rehearsal objectives as follows:

$$\mathcal{L}_R = \mathcal{L}_R^{ST} + \mathcal{L}_R^{D} + \mathcal{L}_R^{Z} \tag{5.23}$$

[4]Each record is comprised of the tuple $(x_S, x_T, y_S, \hat{y}_S^{\text{CIL}}, \hat{y}_T^{\text{CIL}})$, stored in memory with corresponding notation $(x_S^R, x_T^R, y^R, \hat{y}_S^R, \hat{y}_T^R)$, where R signifies "Rehearsal".

A concise depiction of the manner and order in which CDCL optimizes each of its objectives can be observed in Algorithm 3.

5.3.4 Time Complexity Analysis

The runtime complexity of the proposed technique can be broken down into two primary steps: the convolution-based tokenization process and the inter- and intra-task cross-attention mechanism.

The convolutional tokenizer, as seen in Equation (5.1), carries out $O(n)$ operations for each layer L_c, with n representing the size of the input feature space for x_i. A cross-attention mechanism involves a linear transformation of the input into the three projections $\mathbf{Q}, \mathbf{V}$, and $\mathbf{K}$, followed by a subsequent multiplication of dimensions $(d \times d)$, leading to a complexity of $O(nd^2)$. To compute the output of the attention mechanism as per Equation (5.3), the complexity is $O(dn^2)$ for evaluating $\frac{\mathbf{Q}\mathbf{K}_i^T + \mathbf{b}_i}{\sqrt{d}}$, and $O(nd^2)$ for the post-multiplication involving $\mathbf{V}$, with this process repeated across L_a layers. As a result, the single forward-pass time complexity for CDCL can be expressed as:

$$O(\underbrace{nL_c}_{\text{Conv Token}} + \underbrace{(dn^2 + nd^2)L_a}_{\text{Cross-attentions}}) \tag{5.24}$$

This complexity analysis does not account for the influence of general-purpose accelerated hardware architectures, such as GPGPUs or TPUs, on the algorithm's performance. Ultimately, due to parallelism, transformers are still capable of exhibit fast execution times despite its higher computational complexity [163].

5.3.5 Theoretical analysis

Following Theorem 2.1, we can compute the error bound for the cross-domain continual learning problem.

Theorem 5.1. *(Lao et al. [164]) For every new sequential task $t_i \in \mathcal{T} = \{1, \cdots, T\}$ and incoming domain distributions $\mathcal{D}_{S_i}(X_S)$ and $\mathcal{D}_{T_i}(X_T)$, let $d_{\mathcal{H}\Delta\mathcal{H}_i} = d_{\mathcal{H}\Delta\mathcal{H}}(Z_{S_i}, Z_{T_i})$ be the domain discrepancy of the feature distributions Z_{S_i} and Z_{T_i}. The target domain error ε_{T_i} at task t_i is bound by:*

$$\varepsilon_{T_i} \leq \varepsilon_{S_i} + d_{\mathcal{H}\Delta\mathcal{H}_i} + C_i^* \tag{5.25}$$

where $C_i^* = \min_\theta(\varepsilon_{S_i} + \varepsilon_{T_i})$ *is the error of an optimal classifier for both source domain* $\mathcal{D}_{S_i}$ *and target domain* $\mathcal{D}_{T_i}$ *at task* t_i*.*

Theorem 5.2. *(Lao et al. [164]) For every new sequential task* $t \in \mathcal{T} = \{1, \cdots, T\}$ *and incoming domain distributions* $\mathcal{D}_{S_i}(X_S)$ *and* $\mathcal{D}_{T_i}(X_T)$*, let* $d_{\mathcal{H}\Delta\mathcal{H}_i} = d_{\mathcal{H}\Delta\mathcal{H}}(Z_{S_i}, Z_{T_i})$ *be the domain discrepancy of the feature distributions* Z_{S_i} *and* Z_{T_i}*, and* P_{R_i}, P_{M_i} *denote respectively the conditional distributions of labels* Y_{S_i} *given the raw features* Z_{S_i} *and memory stored features* Z_{M_i}*. The total error of the target domain error* ε_T *at task* t *is bound by:*

$$\varepsilon_T \leq \sum_{i=1}^{t}(\varepsilon_{S_i} + d_{\mathcal{H}\Delta\mathcal{H}_i}) + \sum_{i=1}^{t-1} D_{KL}(P_{M_i}||P_{R_i}) + C^* \tag{5.26}$$

where $C^* = \min_\theta \sum_{i=1}^{t}(\varepsilon_{S_i} + \varepsilon_{T_i})$ *is the error of an optimal classifier for both source domain* $\mathcal{D}_{S_\mathcal{T}}$ *and target domain* $\mathcal{D}_{T_\mathcal{T}}$ *across all tasks.*

Proof. At task t, the error of previous learned target experiences $\mathcal{D}_{T_i}(X_T)$ for $i \in \{1, \cdots, t-1\}$ is estimated by $\hat{\varepsilon}_{T_i}$ since we rehearse Z_{M_i} to mimic Z_{S_i} during training. The total error of the target domain error ε_T at task t_i is:

$$\begin{aligned} \varepsilon_T &= \hat{\varepsilon}_{T_i} + \sum_{i=1}^{t-1} \hat{\varepsilon}_{T_i} \\ &\leq \hat{\varepsilon_{S_t}} + d_{\mathcal{H}\Delta\mathcal{H}_t} + \sum_{i=1}^{t-1}(\varepsilon_{S_i} + d_{\mathcal{H}\Delta\mathcal{H}_i}) + C^* \cdot \end{aligned} \tag{5.27}$$

Additionally, the divergence between P_{R_i} and P_{M_i} can be understood as:

$$\begin{aligned} \mathrm{KL}\big(P_{Mi}(Y_{S_i}|Z_{M_i})||P_{R_i}(Y_{S_i}|Z_{S_i})\big) &= \sum_{y_S \in Y_S} P_{M_i}(y_{S_i}|Z_{M_i}) \log \frac{P_{M_i}(y_{S_i}|Z_{M_i})}{P_{R_i}(y_{S_i}|Z_{S_i})} \\ &= \mathbb{E}\Big[\log \frac{P_{M_i}(Y_{S_i}|Z_{M_i})}{P_{R_i}(Y_{S_i}|Z_{S_i})}\Big] \\ &= \mathbb{E}[P_{M_i}(Y_{S_i}|Z_{M_i})] - \mathbb{E}[P_{R_i}(Y_{S_i}|\mathbf{Z}_{\mathrm{S}_i})] \\ &= \hat{\varepsilon}_{S_i} - \varepsilon_{S_i} \cdot \end{aligned} \tag{5.28}$$

Hence, we can estimate the error for each source domain ε_{S_i} in a specific task t as:

$$\varepsilon_{S_i} = D_{\mathrm{KL}}\big(P_{M_i}(Y_{S_i}|Z_{M_i})||P_{R_i}(Y_{S_i}|Z_{S_i})\big) - \hat{\varepsilon}_{S_i} \tag{5.29}$$

Finally, we can estimate the total error bound for the target domain ε_T at a task t by:

$$\begin{aligned}
\varepsilon_T &\leq \varepsilon_{S_t} + d_{\mathcal{H}\Delta\mathcal{H}_t} + \sum_{i=1}^{t-1}(\hat{\varepsilon}_{S_i} + d_{\mathcal{H}\Delta\mathcal{H}_i}) + C^* \\
&= \varepsilon_{S_t} + d_{\mathcal{H}\Delta\mathcal{H}_t} + \sum_{i=1}^{t-1}(\varepsilon_{S_i} + D_{\mathrm{KL}}\big(P_{M_i}||P_{R_i}\big) + d_{\mathcal{H}\Delta\mathcal{H}_i}) + C^* \\
&= \sum_{i=1}^{t}(\varepsilon_{S_i} + d_{\mathcal{H}\Delta\mathcal{H}_i}) + \sum_{i=1}^{t-1} D_{\mathrm{KL}}\big(P_{M_i}||P_{R_i}\big) + C^* \cdot
\end{aligned} \tag{5.30}$$

□

In conclusion, we can employ Theorem 5.2 to elucidate the rationale behind the objective selections in CDCL, based on the aforementioned proof: $\mathcal{L}_R^{\mathrm{CIL}}, \mathcal{L}_D^{\mathrm{CIL}}, \mathcal{L}_R^{\mathrm{TIL}}, \mathcal{L}D^{\mathrm{TIL}}$ contribute to the formation of a feature-aligned domain-invariant latent space, associated with $d\mathcal{H}\Delta\mathcal{H}_i$; $\mathcal{L}_R$ corresponds to the KL term, minimized by the samples chosen through the intra-task center-aware pseudo-label strategy; and lastly, $\mathcal{L}_S^{\mathrm{CIL}}, \mathcal{L}S^{\mathrm{TIL}}$ adapt CDCL to the source domain, reducing the incremental source domain error rate εS_t. C^* signifies the model architecture's aptitude for discovering an optimal solution across both data domains.

5.4 Experiments

5.4.1 Benchmarks

The proposed approach is evaluated on five widely-used UDA benchmarks, including VisDA-2017 [165], Office-Home [166], Office-31 [167], DomainNet [168], and MNIST↔USPS [141, 169].

Algorithm 3: CDCL learning pseudo-algorithm

Input: $\mathcal{D}_{\mathrm{S}}, \mathcal{D}_{\mathrm{T}}$
Output: $f : (X_S \cup X_T) \times \mathcal{T} \to Y_T$
for t_i *in* $\mathcal{T}$ **do**
  $X_S, Y_S \leftarrow \mathcal{D}_{S_i}$;
  $X_T \leftarrow \mathcal{D}_{T_i}$;
  Random initialize K_i and b_i;
  for *epoch in epochs* **do**
    $\mathcal{L}^{\mathrm{CIL}} \leftarrow \mathcal{L}^{\mathrm{TIL}} \leftarrow \mathcal{L}_R \leftarrow 0$;
    if *epoch* $<$ *warm-up-epochs* **then**
      $\mathcal{L}_S^{\mathrm{CIL}} \leftarrow -\sum Y_S \log(\hat{Y}_S^{\mathrm{CIL}})$; # Eq.5.9
      $\mathcal{L}_S^{\mathrm{TIL}} \leftarrow -\sum Y_S \log(\hat{Y}_S^{\mathrm{TIL}})$; # Eq.5.12
    else
      Compute $\mathbf{c}_k$ and $\hat{Y}_T$; # Eqs. 5.17 and 5.18
      Compute $\mathbb{P}$; # Eqs. 5.19 and 5.19
      $\mathcal{L}^{\mathrm{CIL}} \leftarrow \mathcal{L}_S^{\mathrm{CIL}} + \mathcal{L}_T^{\mathrm{CIL}} + \mathcal{L}_D^{\mathrm{CIL}}$; # Eq.5.15
      $\mathcal{L}^{\mathrm{TIL}} \leftarrow \mathcal{L}_S^{\mathrm{TIL}} + \mathcal{L}_T^{\mathrm{TIL}} + \mathcal{L}_D^{\mathrm{TIL}}$; # Eq.5.16
      if $t_i > 1$ **then**
        $\mathcal{L}_R \leftarrow \mathcal{L}_R^{ST} + \mathcal{L}_R^{D} + \mathcal{L}_R^{Z}$; # Eq.5.23
    $\mathcal{L} \leftarrow \mathcal{L}^{\mathrm{CIL}} + \mathcal{L}^{\mathrm{TIL}} + \mathcal{L}_R$;
    $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}$; # Update model parameters
  Stores $\lfloor |M|/t_i \rfloor$ records per task in memory M;

MNIST↔USPS: This benchmark comprises gray-scale images of handwritten digits, representing a classical problem in computer vision and machine learning. The dataset includes 10 distinct classes, corresponding to the digits from 0 to 9. These classes are divided into 5 tasks, with each task containing 2 classes. The objective of this benchmark is to evaluate the performance of domain adaptation methods when dealing with different styles of handwritten digits.

VisDA-2017: This dataset is designed for visual domain adaptation, containing a source dataset of synthetic 2D renderings created from 3D models, generated from various angles and under different lighting conditions. The objective dataset comprises of lifelike pictures captured in authentic situations. The 12 classes in the dataset represent objects such as cars, bicycles, and horses. These classes are divided into 4 tasks, with each task containing 3 classes. The main challenge in this benchmark is to adapt the learning from synthetic to real-world images.

DomainNet: This comprehensive benchmark features 345 classes of common objects, represented across six distinct domains: Clipart (clp), Infographics (inf), Painting (pnt), Quickdraw (qdr), Real (rel), and Sketch (skt). The dataset's 345 classes are split into 15 tasks, with each task encompassing 23 classes. The aim of this benchmark is to assess the domain adaptation performance when confronted with varying artistic styles and visual representations of objects.

Office31: This dataset contains 31 object categories distributed across three domains: Amazon (A), DSLR (D), and Webcam (W). The 31 categories consist of objects typically encountered in office settings, such as keyboards, laptops, and chairs. To simplify the baseline setup, the "trash can" category was excluded, resulting in a dataset with 30 classes divided into 5 tasks of 6 classes each. The main challenge of this benchmark is to evaluate the domain adaptation capabilities when dealing with different image qualities and acquisition devices.

Office-Home: This benchmark includes 65 categories across four domains: Art (Ar), Clipart (Cl), Product (Pr), and Real-World (Re). The dataset's 65 classes are partitioned into 13 tasks, each containing 5 classes. The categories cover a wide range of everyday objects found in homes and offices, such as furniture, kitchenware, and electronic devices. The goal of this benchmark is to assess the performance of domain adaptation techniques when confronted with varying levels of visual abstraction and different image sources.

5.4.2 Baselines

We evaluate our approach against state-of-the-art methods in UDA tasks, such as CDTrans-S and CDTrans-B [113], as well as cutting-edge techniques in continual learning, including DER [82], DER++ [82], and HAL [158]. We also compare our method against MLS [170], which perform supervised cross-domain continual learning.

For the experiments, we designed two variants of CDCL: a smaller instance tailored for the MNIST↔USPS experiments and a larger one suitable for all other benchmarks. The compact CDCL version comprises 7 transformer encoder layers, a 2-layer convolution tokenizer with a 7x7 kernel size, and a convolution input feature space measuring 28x28x1. In contrast, the more

Method		A→D	A→W	D→A	D→W	W→A	W→D	MN→US	US→MN	VisDA-2017
DER	TIL	4.45	4.20	3.99	11.50	4.29	9.14	62.79	73.16	11.43
DER++	TIL	4.21	4.18	4.21	8.51	3.74	8.42	79.81	73.24	10.24
HAL	TIL	4.62	5.02	3.93	5.77	3.70	5.28	80.97	73.38	9.84
MSL	TIL	4.75	4.10	3.89	11.65	4.67	9.23	63.01	71.43	12.21
CDTrans-S	TIL	2.50	5.20	4.50	5.20	4.50	2.50	10.78	10.68	8.90
CDTrans-B	TIL	5.96	3.90	4.70	3.60	4.70	5.80	8.77	9.42	7.85
Ours (ACC)	TIL	**26.22**	**22.43**	**28.74**	**55.44**	**26.54**	**53.21**	**91.91**	**81.48**	**40.80**
Ours (FGT)	TIL	5.95	6.86	0.33	6.38	3.41	9.38	7.38	10.22	7.94
DER	CIL	2.07	3.62	5.23	24.81	7.34	26.50	29.71	29.34	10.09
DER++	CIL	2.07	3.75	6.36	11.11	4.43	19.25	40.82	33.06	9.32
HAL	CIL	4.35	5.81	3.23	5.43	4.43	11.18	29.08	25.94	8.50
MSL	CIL	3.23	4.42	4.43	25.23	**8.01**	24.20	30.22	30.56	**12.23**
Ours (ACC)	CIL	**9.68**	**10.98**	**7.02**	**27.97**	6.73	**28.98**	**66.73**	**52.50**	10.16
Ours (FGT)	CIL	4.14	4.37	1.19	8.30	1.54	7.20	37.03	27.81	26.20

TABLE 5.1: Comparison with SoTA methods' ACC on Office-31, MNIST↔USPS, and VisDA-2017. The best result in each experiment is highlighted with bold font, while the second-best is highlighted with underline font.

Method		Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr
DER	TIL	2.26	2.38	2.93	2.29	3.37	3.09	2.31	2.62	3.18	3.75	1.77	1.72
DER++	TIL	2.25	2.40	2.77	2.60	3.34	3.26	2.32	2.51	3.12	4.02	3.41	3.56
HAL	TIL	2.08	2.10	2.53	2.36	2.84	2.83	2.07	2.42	2.55	3.43	3.07	3.40
MSL	TIL	2.26	2.35	4.01	4.02	3.94	3.65	3.34	3.32	3.72	4.02	2.24	2.14
CDTrans-S	TIL	1.50	1.80	1.50	1.50	1.80	1.50	1.50	1.50	1.50	1.50	1.50	1.80
CDTrans-B	TIL	1.00	1.20	1.40	1.40	1.20	1.40	1.40	1.00	1.65	1.70	1.12	1.20
Ours (ACC)	TIL	**24.44**	**25.18**	**26.20**	**21.25**	**26.64**	**23.54**	**22.89**	**24.21**	**29.44**	**26.25**	**26.27**	**31.25**
Ours (FGT)	TIL	7.33	8.17	10.97	10.90	9.93	11.06	9.05	10.32	13.89	12.75	11.05	19.17
DER	CIL	**4.40**	4.75	**8.45**	4.86	10.63	9.30	4.45	**6.58**	**9.55**	5.00	2.27	2.21
DER++	CIL	3.89	4.93	6.47	5.52	**11.11**	**11.06**	**4.49**	5.50	8.45	**6.84**	4.00	5.23
HAL	CIL	4.01	3.27	6.24	4.70	7.16	7.28	3.38	5.18	4.66	3.54	3.03	4.74
MSL	CIL	4.02	4.35	6.02	4.70	6.44	6.82	4.40	5.10	7.12	4.42	3.64	3.66
Ours (ACC)	CIL	4.15	**4.77**	6.01	**5.56**	8.32	6.35	3.92	5.02	6.29	6.45	**6.26**	**8.35**
Ours (FGT)	CIL	10.36	8.15	13.52	10.15	12.76	11.78	11.96	11.48	21.80	19.45	13.06	27.14

TABLE 5.2: Comparison with SoTA methods' ACC on Office-Home. The best result in each experiment is highlighted with bold font, while the second-best is highlighted with underline font.

extensive version features 14 transformer encoder layers, a 2-layer convolution tokenizer with a 7×7 kernel size, and a convolution input feature space of 224×224×3.

All baseline models were trained for 125 epochs, consisting of 25 warm-up epochs and 25 cooldown epochs, with a fixed memory capacity of 1000 records. CDTrans-S and CDTrans-B were implemented using the same hyper-parameters outlined in the original paper [113]. DER, DER++, and HAL were executed with the optimal hyper-parameters detailed in the Mammoth

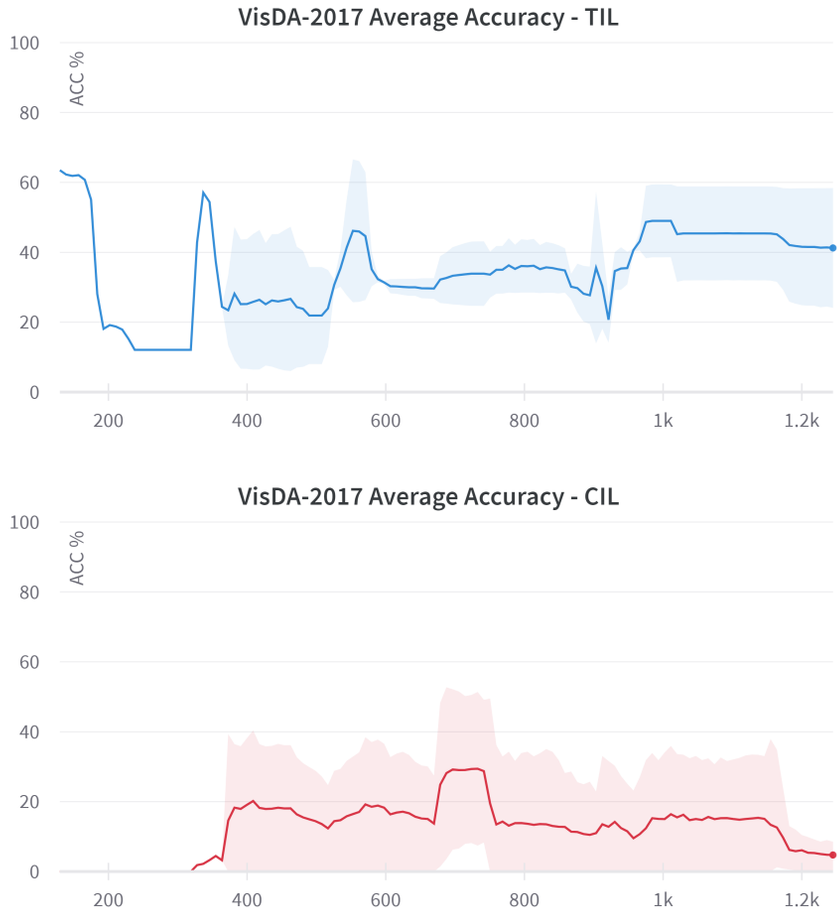


FIGURE 5.2: The evolution of CDCL's ACC in the VisDA-2017 for both TIL and CIL scenarios. The shared area represents the standard deviation of $R_{j,i}, j \in [1, i]$, the accuracy on such a task by a model that learned only previous tasks.

library [171]. CDCL employs the AdamW optimizer [172], featuring a warm-up learning rate of $\lambda = 10^{-5}$, a cosine annealing learning rate that starts at $\lambda = 5 * 10^{-5}$, and a minimum learning rate of $\lambda = 10^{-6}$. All experiments were conducted using an NVIDIA GeForce RTX 2080Ti with 11GB of VRAM.

5.4.3 Numerical results

We examined the ability of all baseline models to address the feature alignment catastrophic forgetting problem. The results for both the TIL and CIL scenarios can be found in Tables 5.1, 5.2, and 5.3, which compare the average accuracy of all baselines and the additional performance gains of CDCL's average forgetting.

	DER	clp	inf	pnt	qdr	rel	skt		DER	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	0.53	0.55	0.49	0.51	0.53	**CIL**	clp	-	1.29	1.32	**1.21**	1.32	1.31
	inf	0.55	-	0.56	0.52	0.52	0.51		inf	1.26	-	**1.32**	1.32	1.24	**1.32**
	pnt	0.54	0.52	-	0.54	0.49	0.50		pnt	1.32	1.33	-	**1.27**	1.31	**1.26**
	qdr	0.53	0.52	0.51	-	0.49	0.52		qdr	1.26	**1.30**	1.29	-	1.19	1.26
	rel	0.53	0.53	0.50	0.49	-	0.51		rel	1.29	1.27	1.22	1.23	-	1.23
	skt	0.51	0.52	0.51	0.52	0.53	-		skt	1.32	1.29	1.20	**1.30**	1.32	-
	DER++	clp	inf	pnt	qdr	rel	skt		DER++	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	0.55	0.55	0.48	0.56	0.57	**CIL**	clp	-	1.30	**1.34**	1.18	1.24	1.27
	inf	0.55	-	0.53	0.55	0.52	0.52		inf	1.23	-	1.31	**1.34**	1.28	1.28
	pnt	0.52	0.52	-	0.56	0.57	0.54		pnt	1.35	1.30	-	1.24	1.26	1.23
	qdr	0.57	0.58	0.57	-	0.54	0.52		qdr	1.32	1.27	**1.35**	-	**1.25**	**1.35**
	rel	0.55	0.54	0.54	0.55	-	0.56		rel	1.36	1.28	1.25	**1.31**	-	**1.24**
	skt	0.55	0.52	0.54	0.57	0.53	-		skt	1.29	1.29	1.20	1.22	1.24	-
	HAL	clp	inf	pnt	qdr	rel	skt		HAL	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	0.48	0.48	0.43	0.48	0.47	**CIL**	clp	-	0.88	0.87	0.81	0.90	0.86
	inf	0.48	-	0.50	0.45	0.49	0.45		inf	0.91	-	0.87	0.85	0.85	0.85
	pnt	0.45	0.47	-	0.51	0.49	0.48		pnt	0.92	0.86	-	0.84	0.86	0.86
	qdr	0.48	0.46	0.46	-	0.48	0.46		qdr	0.88	0.91	0.88	-	0.84	0.85
	rel	0.47	0.45	0.45	0.48	-	0.47		rel	0.87	0.85	0.88	0.87	-	0.92
	skt	0.46	0.46	0.46	0.49	0.45	-		skt	0.94	0.91	0.89	0.90	0.88	-
	MSL	clp	inf	pnt	qdr	rel	skt		MSL	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	0.53	0.53	0.48	0.53	0.53	**CIL**	clp	-	1.30	1.32	1.20	1.30	1.30
	inf	0.54	-	0.55	0.54	0.54	0.50		inf	1.24	-	1.30	1.30	1.26	1.30
	pnt	0.55	0.52	-	0.54	0.52	0.52		pnt	1.35	1.30	-	1.26	1.30	**1.26**
	qdr	0.55	0.55	0.55	-	0.50	0.52		qdr	1.30	1.30	1.30	-	1.20	1.24
	rel	0.55	0.53	0.52	0.53	-	0.55		rel	1.32	1.27	1.23	1.27	-	1.23
	skt	0.53	0.52	0.53	0.55	0.53	-		skt	1.30	1.29	1.20	1.25	1.30	-
	CDTrans-S	clp	inf	pnt	qdr	rel	skt		CDTrans-B	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	0.00	0.00	0.00	0.00	0.00	**TIL**	clp	-	0.00	0.00	0.00	0.00	0.00
	inf	0.30	-	0.40	0.30	0.30	0.30		inf	0.20	-	0.20	0.30	0.20	0.20
	pnt	0.20	0.30	-	0.30	0.0	0.40		pnt	0.20	0.40	-	0.30	0.20	0.20
	qdr	0.20	0.30	0.40	-	0.30	0.30		qdr	0.20	0.40	0.20	-	0.20	0.20
	rel	0.00	0.00	0.00	0.00	-	0.00		rel	0.00	0.00	0.00	0.00	-	0.00
	skt	0.30	0.30	0.40	0.30	0.30	-		skt	0.20	0.40	0.20	0.30	0.20	-
	Ours (ACC)	clp	inf	pnt	qdr	rel	skt		Ours (ACC)	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	**10.86**	**5.70**	**4.68**	**7.48**	**3.40**	**CIL**	clp	-	**1.33**	0.63	0.45	**2.06**	**2.17**
	inf	**21.10**	-	**10.12**	**2.03**	**13.50**	**8.77**		inf	**2.59**	-	1.26	0.25	**1.64**	1.02
	pnt	**22.80**	**10.84**	-	**2.52**	**14.61**	**9.96**		pnt	**2.98**	**1.31**	-	0.28	**1.77**	1.18
	qdr	**16.43**	**2.32**	**3.85**	-	**6.96**	**4.64**		qdr	**1.94**	0.29	0.48	-	0.89	0.58
	rel	**26.22**	**12.26**	**11.85**	**2.57**	-	**10.23**		rel	**3.24**	**1.50**	**1.49**	0.31	-	1.20
	skt	**27.62**	**11.58**	**11.07**	**5.15**	**13.93**	-		skt	**3.49**	**1.35**	**1.31**	0.58	**1.64**	-
	Ours (FGT)	clp	inf	pnt	qdr	rel	skt		Ours (FGT)	clp	inf	pnt	qdr	rel	skt
TIL	clp	-	4.22	6.42	1.44	10.51	2.44	**CIL**	clp	-	5.19	8.12	1.25	11.39	2.54
	inf	7.73	-	3.93	0.81	4.96	3.45		inf	9.46	-	4.59	0.96	6.27	4.07
	pnt	9.04	4.21	-	0.97	5.42	3.88		pnt	11.18	5.22	-	1.17	6.73	4.61
	qdr	6.19	0.89	1.50	-	2.83	1.91		qdr	7.37	1.08	1.83	-	3.41	2.32
	rel	10.25	4.70	4.59	1.06	-	3.60		rel	12.36	5.89	5.79	1.24	-	4.49
	skt	11.14	4.32	4.38	1.85	5.52	-		skt	13.20	5.17	5.51	2.47	6.70	-

TABLE 5.3: Comparison with SoTA methods' ACC on DomainNet. The rows represent the source dataset, while the columns represent the target dataset. The best result in each experiment is highlighted with bold font.

Our analysis shows that CDCL outperforms all baselines in the TIL setup. This superior performance can be attributed to CDCL's inter- and intra-task cross-attention mechanism, which preserves prior task information in the attention maps (queries and values projections) while maximizing intra-task knowledge in the key vectors. Additionally, CDCL excels in situations where

the target domain closely resembles the source domain, as observed in the MNIST↔USPS experiments. This is primarily due to its intra-task center-aware pseudo-labeling mechanism, which enables successful semi-supervised prediction of the target domain. The results obtained by CDCL in the D→W and W→D settings provide further evidence supporting this claim. Lastly, CDCL is the only method capable of demonstrating learning and mitigation of catastrophic forgetting in the DomainNet experiments.

Nevertheless, certain limitations should be acknowledged. All methods, including CDCL, require further improvement in addressing the CIL scenario. CDCL adopts the experimental replay strategy introduced by DER and DER++ to rapidly retrieve forgotten information; consequently, all three baselines exhibit similar performance in such demanding experiments. The underlying hypothesis for this approach is that the forgotten knowledge still exists in some residual form within the network parameters and can be partially recovered through experience replay. Future research should focus on developing an enhanced mechanism to preserve inter-task common knowledge while adapting to new tasks.[5]

5.4.4 Ablation study

In our investigation, we delved into the influence of the three losses-block on the final ACC performance of CDCL. Additionally, we explored the impact of the *inter- intra-task cross-attention*, which involves evaluating how a standard simple attention mechanism on the source domain enables the network to generalize to the target domain. The findings from this investigation are presented in Table 5.4.

In terms of overall impact, the intra-task loss has the most significant effect. This highlights the importance of intra-task loss in maintaining the performance of the model across various tasks. Consequently, focusing on the optimization of intra-task loss can lead to further improvements in the CDCL approach.

Following the intra-task loss, the rehearsal loss has the second most substantial impact on the model's performance. This suggests that the rehearsal loss

[5]The work presented by Lao et al. [164] appears to address the CIL scenario. However, the authors have not released their method's source code, and their experimental setup deviates significantly from the standard.

Experiment	Loss / Module $\mathcal{L}^{\mathbf{CIL}}$	$\mathcal{L}^{\mathbf{TIL}}$	$\mathcal{L}_{\mathrm{R}}$	MN→US **TIL**	MN→US **CIL**	US→MN **TIL**	US→MN **CIL**
-	✓	✓	✓	91.91	66.73	81.48	52.50
A	-	✓	✓	81.88	63.71	67.31	47.86
B	✓	-	✓	59.17	46.33	60.20	34.65
C	✓	✓	-	68.71	19.59	71.72	15.83
Simple attention	✓*	✓*	✓*	62.72	29.82	73.20	29.50

TABLE 5.4: Ablation study analyzing the effect of each loss module into CDCL's ACC. Additionally, a study where the inter- intra-task cross-attention mechanism is substitute by a simple attention mechanism is presented. All losses are present in the simple attention mechanism, but they only contains information about the source domain.

plays a crucial role in preserving previously learned knowledge and preventing catastrophic forgetting.

Interestingly, the minimal impact on performance caused by the absence of inter-task loss indicates that the majority of knowledge is stored in the $\mathbf{K}_i$ and $\mathbf{b}_i$ projections. This observation implies that these projections are vital for retaining and transferring information across tasks.

Moreover, in the context of the CIL scenario, rehearsal loss plays a more critical role in maintaining and adapting inter-task information. This finding underscores the importance of developing strategies that effectively balance the different loss components to enhance CDCL's performance in various scenarios

Finally, it was noted that when utilizing the standard simple attention mechanism, CDCL's main contribution is compromised, leading to a performance that closely resembles that of DER (Domain-Adversarial Network for Domain-Adaptive Visual Recognition) and DER++. This outcome underscores the unique and significant impact of CDCL's approach to cross-domain continual learning and the importance of its inter-intra-task cross-attention mechanism in achieving superior performance over alternative methods

5.5 Conclusion

In this work, we introduce CDCL, a novel framework aimed at cross-domain continual learning that tackles the unsupervised cross-domain problem within the task-incremental learning framework. CDCL employs two primary mechanisms to achieve this goal.

First, the inter- and intra-task cross-attention block enables the model to effectively learn and retain knowledge across multiple tasks. This is achieved by utilizing the transformer architecture, which has shown great success in various domains.

Second, the intra-task center-aware pseudo-labeling procedure allows CDCL to further enhance its performance by making semi-supervised predictions in the target domain. This mechanism aids the model in generalizing to new tasks and domains while maintaining its performance on previous tasks.

By combining these two mechanisms, CDCL leverages the strengths of the transformer architecture to learn multiple domains in a continual learning setup, thereby addressing the challenges posed by unsupervised cross-domain problems in task-incremental learning.

Chapter 6

Conclusions and Future Directions

The ultimate objective of machine learning is to emulate human-like learning by continuously and automatically acquiring knowledge across various domains. As the field of artificial intelligence experiences a renaissance and machine learning algorithms continue to evolve, lifelong learning is becoming increasingly significant in realizing this objective. Indeed, machine learning has already outperformed humans in numerous specialized tasks, such as image and object recognition [28, 29], video games [30, 31], as well as speech generation and recognition [32, 33]. Nevertheless, to achieve general intelligence, a system must possess the capability to accumulate and learn diverse types of knowledge over time.

We anticipate that the field of lifelong learning will attract a substantial amount of research effort in the coming years for several compelling reasons. First, traditional machine learning approaches are reaching a point of diminishing returns, as further advancements become increasingly complex and resource-intensive. Leveraging past knowledge to facilitate learning is a natural method to mimic human learning processes, thereby offering a potential avenue for growth in the field.

Second, the vast and ever-growing amount of data available across different domains presents an opportunity for lifelong learning agents to concurrently acquire multiple types of knowledge. This data explosion enables machine

learning models to become more versatile and adaptive, handling a variety of tasks that span numerous domains.

Third, continuous lifelong learning has become necessary due to the increasing prevalence of intelligent systems that interact with humans and the physical world. Examples of such systems include robots and autonomous devices operating in dynamic environments, chatbots that engage with users in natural language, and virtual assistants that assist in various tasks. The development of these applications requires lifelong learning to be an indispensable element due to the crucial nature of being able to acquire new knowledge and adjust to diverse situations and obstacles.

Lastly, the pursuit of lifelong learning also aligns with the broader goal of creating more human-like AI systems. By integrating the ability to learn continually across diverse domains, AI models can become more robust, adaptable, and capable of tackling complex real-world challenges. This ongoing pursuit will undoubtedly fuel the growth and innovation within the field of AI and machine learning, driving us closer to achieving true general intelligence.

6.1 Conclusions

The research studies presented in this report tackle the problem of lifelong learning, precisely sequential learning (online learning) and incremental learning (continual learning) upon an optics of transfer learning, i.e., on how these fields can enhance from the exploration of techniques that includes learning from multiple data sources simultaneously.

The major conclusions from the research studies presented in this report are as follows:

- **Online Unsupervised Cross-Domain Adaptation**: We proposed ACDC, an innovative framework designed to tackle the challenges of online unsupervised cross-domain adaptation. This problem is commonly encountered in transfer learning scenarios where there are insufficient labeled samples in the target data distribution. ACDC is able to overcome this constraint and handle several other challenges through its various mechanisms and components.

ACDC can handle the following challenges through its various mechanisms:

A. *Scarcity of labeled samples*: To address this challenge, ACDC builds a common latent space representation using a denoising autoencoder between the source and target domains. This shared latent space is then learned by a discriminator, treating the two domains as a single data distribution. By doing so, ACDC can effectively use the labeled samples from the source domain to enhance the classification performance on the target domain.

B. *Different marginal probability distributions*: ACDC uses a gradient reversal layer in its adversarial domain adaptation mechanism to force a common latent space in the denoising autoencoder, even if the marginal probability distributions differ. By aligning the two domains in the same latent space, ACDC can transfer knowledge from the source to the target domain.

C. *Covariate shift*: ACDC's adversarial domain adaptation mechanism adjusts the distribution of the input features in the latent space to match the distribution in the target domain. By mitigating the impact of dissimilar input feature distributions across source and target domains, this technique effectively minimizes bias within the model's predictions.

D. *Asynchronous drift*: ACDC is composed of three modules that can adapt dynamically to changes in the incoming samples from the source and target domains. The denoising autoencoder module grows and prunes nodes based on their reconstruction error, while the adversarial domain adaptation network and the multilayer-perceptron discriminator module grow and prune nodes based on their classification error. This enables ACDC to handle the asynchronous drift in the data streams effectively and adapt to the changing patterns of the data in real-time.

E. *Contrasting throughput*: ACDC can accommodate contrasting throughput between the source and target data streams. This is achieved by building a sliding window that contains pairs of source and target samples. ACDC can then permute and pair these incoming samples to learn multiple streams generated at different speeds. This feature

enables ACDC to adapt to various input rates, ensuring that it can learn from all the available data, regardless of the speed at which it is generated.

– **Class-Incremental Learning via Knowledge Amalgamation**: We propose CFA, a new framework designed to address the issue of catastrophic forgetting in incremental learning. CFA utilizes a powerful technique known as *knowledge amalgamation*, which is derived from knowledge distillation. Unlike knowledge distillation, which involves transferring knowledge from a single model (teacher) to another model (student) without requiring labeled information, knowledge amalgamation goes a step further by combining knowledge from multiple teachers into a single student.

  To address the problem of catastrophic forgetting, CFA builds multiple teachers for each task and transfers their knowledge to a final student. This allows the student to retain its previously learned knowledge while also learning new tasks incrementally. Moreover, CFA supports heterogeneous knowledge amalgamation, meaning that the teachers and students do not need to have the same structure. Additionally, the teachers can have interpolated tasks with each other. This simple yet powerful solution makes CFA easy to integrate into existing learning pipelines that are primarily focused on static learning agents.

– **Towards Cross-Domain Continual Learning**: Finally, we propose CDCL, a new framework that brings unsupervised cross-domain capabilities into a continual learning model. CDCL is designed to address the feature-alignment catastrophic forgetting problem that can occur in task-incremental learning scenarios while maintaining a good stability-plasticity trade-off.

  To solve the feature-alignment catastrophic forgetting problem, CDCL introduces two techniques. The first is an inter- and intra-task cross-attention mechanism that aligns domain features in current tasks and consolidates previous alignment knowledge. This mechanism ensures that the model retains its previously learned knowledge while also adapting to new domains and tasks. The second technique is an intra-task-based center-aware pseudo-labeling strategy that identifies similar task-specific samples between the domains. This strategy enables the model

to effectively transfer knowledge between domains and tasks without requiring any labeled information.

In a clear and accessible manner, this thesis presents a series of significant discoveries made throughout the course of the current research, as detailed in the following chapters

* Chapter 3 explore the remarkable capabilities of Deep Neural Networks (DNNs) in the context of Autonomous Cross Domain Conversion (ACDC). This research demonstrates how DNNs excel in learning across multiple interconnected domains in an online setting, even when annotations are absent. ACDC serves as a pioneering approach to addressing continual domain changes and enhancing the adaptability of neural networks.
* Chapter 4 introduces the innovative framework known as Catastrophic Forgetting Solution via Knowledge Amalgamation (CFA). This framework has the potential to reshape the landscape of machine learning. CFA seamlessly transforms traditional offline machine learning environments into a unified, continual learning framework. It ensures that models remain adaptable and capable of mitigating the issue of catastrophic forgetting, thereby enhancing their real-world utility.
* Chapter 5 delves into the intricacies of the Cross-Domain Continual Learning (CDCL) framework, providing insights into the architecture and methodologies that facilitate efficient knowledge transfer and retention across diverse domains. CDCL stands as a testament to the pursuit of agile and versatile machine learning systems.

These chapters collectively represent the culmination of our research endeavors, offering valuable contributions to the ever-evolving landscape of machine learning and lifelong learning methodologies.

6.2 Future Directions

In addition to the contributions of this research thesis, there are several challenging problems and future directions in lifelong learning, machine learning, and artificial intelligence that warrant attention. Addressing

these issues has the potential to fundamentally impact the field and pave the way for future breakthroughs.

Direct future research directions developed from this research thesis are:

* **Cross-Domain in the Class-Incremental Learning Scenario**: In the context of this research thesis, we have successfully introduced a cross-domain continual learning framework. However, it is imperative to acknowledge that there remains considerable scope for refinement, particularly in addressing the challenging class-incremental learning scenario. This scenario, which is prevalent in most real-world applications, warrants heightened attention due to its practical significance. Developing a continual learning model capable of seamlessly integrating knowledge from multiple data sources represents a pivotal step toward creating agents that can learn in a manner akin to human adaptability.

  Recent efforts aimed at constructing comprehensive class-incremental learning scenarios have explored orthogonal increments, as elucidated by Farajtabar et al. [173]. In this approach, the gradients associated with new incoming classes are constrained in specific directions, thereby preserving previously-acquired knowledge. This promising line of research contributes to the ongoing pursuit of effective class-incremental learning strategies.

  Moreover, when considering superior alternatives for cross-domain continual learning, as discussed in Chapter 5, the domain of reinforcement learning has emerged as particularly auspicious. Recent advancements in reinforcement learning techniques, exemplified by [30, 31, 34], offer greater promise in handling unknown scenarios and novel classes. This approach surpasses the self-supervised and unsupervised techniques employed in our research [132]. Furthermore, it exhibits the ability to adapt and continue learning over time, aligning more closely with the requirements of life-long learning scenarios.

* **Cross-Domain on Online Continual Learning**: The demarcation between continual learning and online learning is progressively blurring, foreshadowing a future in which computational agents will possess the capacity to acquire new knowledge in a streaming

fashion while safeguarding previously assimilated knowledge. Although extant endeavors like "Lambda Learner" [174] have made strides in addressing the challenge of incremental learning within data streams, they are generally predicated on the utilization of a single data source at any given time. We envision a forthcoming paradigm where computational agents, operating in a cross-domain milieu, will seamlessly glean insights from multiple data sources concurrently.

This evolving avenue of research closely intersects with the domains of novel class discovery (NCD) [175] and domain generalization (DG) [176], research fields that have remained relatively uncharted in the context of the present thesis. In brief, NCD is concerned with the identification and accurate classification of previously unknown classes within the learned data distribution of a model, whereas DG signifies a significant advancement beyond domain adaptation. DG necessitates that machine learning systems exhibit exceptional generalization capabilities when confronted with unforeseen data distributions during the inference phase.

The evolving landscape of continual and online learning, i.e. fully life-long learning, heralds an era of knowledge acquisition marked by fluidity and adaptability. The integration of multiple data sources, coupled with the burgeoning domains of NCD and DG, promises to significantly enrich the capabilities of computational agents, further bridging the chasm between machine and human learning paradigms.

To further advance the contributions put forth in this thesis, it is imperative to embark on a trajectory of research aimed at enhancing each of these accomplishments:

- **CFA**: The augmentation of CFA demands focused efforts towards the refinement of its core principles. Specifically, there is a need to establish a framework that facilitates incremental knowledge amalgamation among two or more models, with minimal error propagation. This entails achieving a high degree of both backward and forward knowledge transfer during the amalgamation process. Presently, CFA necessitates the retention of not only past data samples but also previous model

states. An incremental CFA approach would alleviate the space complexity associated with the storage of previous model states. Addressing this challenge is pivotal to the evolution of CFA.

- **ACDC** and **CDCL**: While ACDC and CDCL exhibit commonalities, they operate within distinct learning settings. To unlock the potential of a holistic, long-life cross-domain learning paradigm — one characterized by the continuous assimilation of incoming classes from related domains in real-time — it is imperative to amalgamate the theories underpinning ACDC and CDCL into a unified model. However, this ambitious endeavor is accompanied by substantial challenges. Realizing this vision necessitates the development and application of novel techniques that not only excel in incremental learning but also exhibit low time complexity. Achieving such a synthesis represents a formidable task that requires pioneering solutions to bridge the gap between the complementary strengths of ACDC and CDCL.

Additionally, Transfer Learning and Continual Learning researchers should collaborate and integrate existing solutions within other machine learning subfields, such as contrastive learning [177], multi-task learning [41], multi-target learning [42], reinforcement learning with human feedback [178], and few/zero-shot learning [159]. This would enable us to leverage the strengths of different machine learning techniques and create more robust and effective models that can address the challenges of lifelong learning in a more comprehensive manner.

General future research directions along the fields of Transfer Learning and Continual Learning are:

- **Multi-modal Learning**: Multi-modal learning involves processing and integrating information from various sensory modalities such as audio, video, and text. Combining transfer learning and lifelong learning approaches can enable computational agents to learn and adapt across different modalities, making them more versatile and capable of handling complex real-world scenarios.

- **Self-Supervised Learning**: Self-supervised learning is an emerging technology that enables agents to learn from unlabeled data, leveraging the intrinsic structure and relationships within the data. Combining this technique with transfer and lifelong learning approaches can enhance the agent's ability to learn continuously from diverse data sources without requiring explicit supervision.
- **Explainable AI**: Explainable AI is becoming increasingly important as models become more complex and data-driven. Transfer learning and lifelong learning approaches that produce interpretable models can help to increase transparency and accountability, enabling users to better understand the model's decision-making process.
- **Human-AI Collaboration**: The ability of humans and AI to work together is a critical factor for the success of AI in real-world applications. Transfer learning and lifelong learning approaches that enable agents to learn from and collaborate with humans can create more effective and trustworthy AI systems.
- **Active Learning**: Active learning involves selecting the most informative samples from a large unlabeled dataset for labeling by an expert. Combining transfer learning and lifelong learning approaches with active learning can improve the efficiency of labeling and reduce the amount of labeled data required, enabling agents to learn more effectively and efficiently.
- **Real-Time Learning**: Real-time learning involves updating the model in real-time as new data becomes available. Transfer learning and lifelong learning approaches that enable agents to learn continuously and incrementally can enhance their ability to learn in real-time and adapt to changing environments.

In summary, there are many exciting challenges and opportunities beyond this research thesis that can significantly impact lifelong learning, machine learning, and artificial intelligence. By addressing these challenges and collaborating across subfields, we can develop more effective and robust models that can learn continuously and adapt to changing environments.

List of Author's Publications

Journal Articles

- **Marcus de Carvalho**, Mahardhika Pratama, Jie Zhang and Edward Yapp Kien Yee "ACDC: Online Unsupervised Cross-Domain Adaptation," in *Knowledge-Based Systems, 2022*.
- Weiwei Weng, Mahardhika Pratama, Choiru Za'in, **Marcus de Carvalho**, Rakaraddi Appan, Andri Ashfahani and Edward Yapp Kien Yee "Autonomous Cross Domain Adaptation Under Extreme Label Scarcity," in *IEEE TNNLS, 2022*.

Conference Proceedings

- Mahardhika Pratama, **Marcus de Carvalho**, Renchunzi Xie, Edwin Lughofer and Jie Lu, "ATL: Autonomous Knowledge Transfer from Many Streaming Processes," in *ACM CIKM, 2019*.
- **Marcus de Carvalho**, Mahardhika Pratama, Jie Zhang and Yajuan Sun, "Class-Incremental Learning via Knowledge Amalgamation," in *ECML PKDD, 2022*.
- Appan Rakaraddi, Lam Siew Kei, Mahardhika Pratama and **Marcus de Carvalho**, "Reinforced Continual Learning for Graphs," in *ACM CIKM, 2022*.
- **Marcus de Carvalho**, Mahardhika Pratama, Jie Zhang, Chua Haoyan and Edward Yapp, "Towards Cross-Domain Continual Learning," in *Submitted to IEEE ICDE, 2024*.

Bibliography

[1] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1180–1189. JMLR.org, 2015. xvii, 23, 36, 40, 42, 46

[2] James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102 3:419–457, 1995. 1

[3] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165, January 1989. ISSN 0079-7421. doi: 10.1016/S0079-7421(08)60536-8. Funding Information: The research reported in this chapter was supported by NIH grant NS21047 to Michael McCloskey, and by a grant from the Sloan Foundation to Neal Cohen. We thank Sean Purcell and Andrew Olson for assistance in generating the figures, and Alfonso Caramazza, Walter Harley, Paul Macaruso, Jay McClelland, Andrew Olson, Brenda Rapp, Roger Ratcliff, David Rumelhart, and Terry Sejnowski for helpful discussions. 20

[4] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081. 1, 16, 17, 19, 20, 82

[5] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.09.116. URL `https://www.sciencedirect.com/science/article/pii/S0925231215017634`. Recent Developments on Deep Big Vision. 1

[6] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015. 1

[7] Ronald Kemker, Angelina Abitino, Marc McClure, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2018. 1

[8] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116, 04 2019. doi: 10.1016/j.neunet.2019.03.010. 1

[9] Angelo Cangelosi and Matthew Schlesinger. From babies to robots: The contribution of developmental robotics to developmental psychology. *Child Development Perspectives*, 12(3): 183–188, September 2018. ISSN 1750-8592. doi: 10.1111/cdep.12282. 1, 2, 8

[10] Jun Tani. *Exploring Robotic Minds: Actions, Symbols, and Consciousness as Self-Organizing Dynamic Phenomena*. Oxford University Press, Inc., USA, 1st edition, 2016. ISBN 0190281065. 1, 2

[11] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015. doi: 10.1109/MCI.2015.2471196. 2

[12] Andrew Bremner, David Lewkowicz, and Charles Spence. Multisensory development, 11 2013. 2, 3, 7

[13] Micah M. Murray, David J. Lewkowicz, Amir Amedi, and Mark T. Wallace. Multisensory processes: A balancing act across the lifespan. *Trends in Neurosciences*, 39(8):567–579, 2016. ISSN 0166-2236. doi: https://doi.org/10.1016/j.tins.2016.05.003. URL `https://www.sciencedirect.com/science/article/pii/S0166223616300480`. 2

[14] David Lewkowicz. Early experience and multisensory perceptual narrowing. *Developmental psychobiology*, 56, 02 2014. doi: 10.1002/dev.21197. 3, 7

[15] Jonathan Power and Bradley Schlaggar. Neural plasticity across the lifespan. *Wiley Interdisciplinary Reviews: Developmental Biology*, 6, 11 2016. doi: 10.1002/wdev.216.

[16] Friedemann Zenke, Wulfram Gerstner, and Surya Ganguli. The temporal paradox of hebbian learning and homeostatic plasticity. *Current Opinion in Neurobiology*, 43:166–176, 04 2017. doi: 10.1016/j.conb.2017.03.015. 2

[17] Kai A. Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009. ISSN 0010-0277. doi: https://doi.org/10.1016/j.cognition.2008.11.014. URL `https://www.sciencedirect.com/science/article/pii/S0010027708002850`. 2, 8

[18] Stefan Wermter, Günther Palm, and Mark Elshaw. *Biomimetic Neural Learning for Intelligent Robots - Intelligent Systems, Cognitive Robotics, and Neuroscience*. 01 2005. ISBN ISBN-10 3-540-27440-5.

[19] B. Skinner. Reinforcement today. *American Psychologist*, 13: 94–99, 03 1958. doi: 10.1037/h0049039. 2, 8

[20] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10): 1345–1359, October 2010. 2, 8, 11, 21, 22, 23, 34

[21] Susan Barnett and Stephen Ceci. When and where do we apply what we learn? a taxonomy for far transfer. *Psychological bulletin*, 128:612–37, 08 2002. doi: 10.1037/0033-2909.128.4.612. 3, 7

[22] Barry Stein, Terrence Stanford, and Benjamin Rowland. Development of multisensory integration from the perspective of the individual neuron. *Nature reviews. Neuroscience*, 15: 520–35, 07 2014. doi: 10.1038/nrn3742. 3

[23] Charles Spence. Crossmodal spatial attention. *Annals of the New York Academy of Sciences*, 1191:182–200, 03 2010. doi: 10.1111/j.1749-6632.2010.05440.x. 3, 7

[24] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2017.06.011. URL `https://www.sciencedirect.com/science/article/pii/S0896627317305093`. 3, 8

[25] Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1):25–46, 1995. ISSN 0921-8890. doi: https://doi.org/10.1016/0921-8890(95)00004-Y. URL `https://www.sciencedirect.com/science/article/pii/092188909500004Y`. The Biology and Technology of Intelligent Autonomous Agents. 3, 8

[26] Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting, 2022. URL `https://arxiv.org/abs/2206.00388`. 3

[27] João Gama. *Knowledge discovery from data streams*. CRC Press, Boca Raton, Fl., 2010. 7, 25, 34, 68

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 7, 49, 79, 80, 107

[29] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJWLfGWRb`. 107

[30] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. ISSN 0028-0836. doi: 10.1038/nature16961. 107, 112

[31] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017. URL `http://arxiv.org/abs/1712.01815`. 107, 112

[32] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016. URL `http://arxiv.org/abs/1609.03499`. cite arxiv:1609.03499. 107

[33] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. Lipnet: Sentence-level lipreading. *CoRR*, abs/1611.01599, 2016. URL `http://arxiv.org/abs/1611.01599`. 7, 107

[34] Kathryn Tunyasuvunakool, Jonas Adler, Zachary Wu, Tim Green, Michal Zielinski, Augustin Žídek, Alex Bridgland, Andrew Cowie, Clemens Meyer, Agata Laydon, Sameer Velankar, Gerard Kleywegt, Alex Bateman, Richard Evans, Alexander Pritzel, Michael Figurnov, Olaf Ronneberger, Russ Bates, Simon Kohl, and Demis Hassabis. Highly accurate protein structure prediction for the human proteome. *Nature*, 596:1–9, 08 2021. doi: 10.1038/s41586-021-03828-1. 7, 112

[35] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207: 117921, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117921. URL `https://www.sciencedirect.com/science/article/pii/S0957417422011654`.

[36] Rong Jin. Deep learning at alibaba. In *IJCAI*, pages 11–16, 2017.

[37] Prajyot Mane, Shubham Sonone, Nachiket Gaikwad, and Jyoti Ramteke. Smart personal assistant using machine learning. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 368–371, 2017. doi: 10.1109/ICECDS.2017.8390128. 7

[38] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis Machine Intelligence*, (01):1–1, may 5555. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3079209. 8

[39] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52, 07 2010. 8

[40] John M. Giorgi, Xindi Wang, Nicola Sahar, Won Young Shin, Gary D. Bader, and Bo Wang. End-to-end named entity recognition and relation extraction using pre-trained language models. *CoRR*, abs/1912.13415, 2019. URL `http://arxiv.org/abs/1912.13415`. 10

[41] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73:243–272, 12 2008. doi: 10.2139/ssrn.1031158. 11, 114

[42] Donna Xu, Yaxin Shi, Ivor Tsang, Yew Ong, Chen Gong, and Xiaobo Shen. Survey on multi-output learning. *IEEE Transactions on Neural Networks and Learning Systems*, PP: 1–21, 11 2019. doi: 10.1109/TNNLS.2019.2945133. 11, 114

[43] S. Thrun. A lifelong learning perspective for mobile robot control. In V. Graefe, editor, *Intelligent Robots and Systems*. Elsevier, 1995. 11, 67

[44] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Min. Knowl. Discov.*, 30(4):964–994, July 2016. 12, 25, 34, 35

[45] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V.

Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf. 12, 16, 19, 83, 88

[46] J. Kivinen, A.J. Smola, and R.C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8): 2165–2176, 2004. doi: 10.1109/TSP.2004.830991. 12, 25

[47] Matthew Hoffman, Francis Bach, and David Blei. Online learning for latent dirichlet allocation. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf. 12

[48] Mark Herbster, Massimiliano Pontil, and Lisa Wainer. Online learning over graphs. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 305–312, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102390. URL https://doi.org/10.1145/1102351.1102390. 13

[49] Mahardhika Pratama, Choiru Za'in, Andri Ashfahani, Yew Soon Ong, and Weiping Ding. Automatic construction of multi-layer perceptron network from streaming examples. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1171–1180, New York, NY, USA, 2019. Association for Computing Machinery. 13

[50] Mahardhika Pratama, Andri Ashfahani, Yew-Soon Ong, Savitha Ramasamy, and Edwin Lughofer. Autonomous deep learning: Incremental learning of denoising autoencoder for evolving data streams. *CoRR*, abs/1809.09081, 2018. 13

[51] Mahardhika Pratama, Marcus de Carvalho, Renchunzi Xie, Edwin Lughofer, and Jie Lu. Atl: Autonomous knowledge transfer from many streaming processes. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 269–278, New York, NY, USA, 2019. Association for Computing Machinery. 13, 26, 35, 50, 74

[52] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Scholkopf. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 601–608, Cambridge, MA, USA, 2006. MIT Press. 13, 25

[53] Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015. doi: 10.1109/TKDE.2014.2345382. 25

[54] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *In SBIA Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer Verlag, 2004. 13, 14, 44

[55] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016. 13, 18

[56] Albert Bifet, Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Indrė Žliobaitė. Cd-moa: Change detection framework for massive online analysis. In Allan Tucker, Frank Höppner, Arno Siebes, and Stephen Swift, editors, *Advances in Intelligent Data Analysis XII*, pages 92–103, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 13

[57] Hang Yu, Jie Lu, and Guangquan Zhang. Morstreaming: A multioutput regression system for streaming data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–13, 2021. doi: 10.1109/TSMC.2021.3102978. 13, 25

[58] Hang Yu, Jie Lu, Anjin Liu, Bin Wang, Ruimin Li, and Guangquan Zhang. Real-time prediction system of train carriage load based on multi-stream fuzzy learning. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2022. doi: 10.1109/TITS.2021.3137446. 13, 25

[59] Mohammed Oualid Attaoui, Hanene Azzag, Mustapha Lebbah, and Nabil Keskes. Improved multi-objective data stream clustering with time and memory optimization. *CoRR*, abs/2201.05079, 2022. URL `https://arxiv.org/abs/2201.05079`. 13, 25

[60] Swarup Chandra, Ahsanul Haque, Latifur Khan, and Charu Aggarwal. An adaptive framework for multistream classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 1181–1190, New York, NY, USA, 2016. Association for Computing Machinery. 13, 25, 35, 50

[61] Ahsanul Haque, Zhuoyi Wang, Swarup Chandra, Bo Dong, Latifur Khan, and Kevin W. Hamlen. Fusion: An online method for multistream classification. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 919–928, New York, NY, USA, 2017. Association for Computing Machinery. 13, 25, 35, 50

[62] Bo Dong, Yang Gao, Swarup Chandra, and Latifur Khan. Multistream classification with relative density ratio estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3478–3485, Jul. 2019. doi: 10.1609/aaai.v33i01.33013478. URL `https://ojs.aaai.org/index.php/AAAI/article/view/4225`. 13, 25

[63] Honghui Du, Leandro L. Minku, and Huiyu Zhou. Multi-source transfer learning for non-stationary environments. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8852024. URL `https://doi.org/10.1109/IJCNN.2019.8852024`. 13, 25, 50

[64] Honghui Du, Leandro L. Minku, and Huiyu Zhou. Marline: Multi-source mapping transfer learning for non-stationary environments. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 122–131, 2020. doi: 10.1109/ICDM50108.2020.00021. 13, 25

[65] Mark Dredze and Koby Crammer. Online methods for multi-domain learning and adaptation. pages 689–697, 01 2008. doi: 10.3115/1613715.1613801. 13

[66] Yi-Fan Li, Yang Gao, Gbadebo Ayoade, Hemeng Tao, Latifur Khan, and Bhavani Thuraisingham. Multistream classification for cyber threat data with heterogeneous feature space. In *The World Wide Web Conference*, WWW '19, page 2992–2998, New York, NY, USA, 2019. Association for Computing Machinery. 13, 26, 35, 53

[67] H. Tao, Z. Wang, Y. Li, M. Zamani, and L. Khan. Comc: A framework for online cross-domain multistream classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi: 10.1109/IJCNN.2019.8851931. 13, 26, 35, 53

[68] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, apr 1996. ISSN 0885-6125. doi: 10.1023/A:1018046501280. URL `https://doi.org/10.1023/A:1018046501280`. 14

[69] Anjin Liu, Yiliao Song, Guangquan Zhang, and Jie Lu. Regional concept drift detection and density synchronized drift adaptation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2280–2286, 2017. doi: 10.24963/ijcai.2017/317. URL `https://doi.org/10.24963/ijcai.2017/317`. 14, 15

[70] Jeffrey C. Schlimmer and Richard H. Granger. Incremental learning from noisy data. *Mach. Learn.*, 1(3):317–354, mar 1986. ISSN 0885-6125. doi: 10.1023/A:1022810614389. URL `https://doi.org/10.1023/A:1022810614389`. 14

[71] Anjin Liu, Guangquan Zhang, and Jie Lu. Fuzzy time windowing for gradual concept drift adaptation. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2017. doi: 10.1109/FUZZ-IEEE.2017.8015596. 14

[72] Ning Lu, Jie Lu, Guangquan Zhang, and Ramon Lopez de Mantaras. A concept drift-tolerant case-base editing technique. *Artif. Intell.*, 230(C):108–133, jan 2016. ISSN 0004-3702. doi: 10.1016/j.artint.2015.09.009. URL `https://doi.org/10.1016/j.artint.2015.09.009`. 14

[73] Ning Lu, Guangquan Zhang, and Jie Lu. Concept drift detection via competence models. *Artif. Intell.*, 209:11–28, apr 2014. ISSN 0004-3702. doi: 10.1016/j.artint.2014.01.001. URL `https://doi.org/10.1016/j.artint.2014.01.001`. 14, 15

[74] Bartosz Krawczyk, Leandro Minku, João Gama, Jerzy Stefanowski, and Michal Wozniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 09 2017. doi: 10.1016/j.inffus.2017.02.004. 14

[75] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michal Wozniak, and Francisco Herrera. A survey on data

preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239, 02 2017. doi: 10.1016/j.neucom.2017.01.078. 14, 15

[76] Amos Storkey. *When Training and Test Sets Are Different: Characterizing Learning Transfer*, pages 3–28. 01 2009. ISBN 9780262170055. doi: 10.7551/mitpress/9780262170055.003.0001. 14

[77] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Hamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46, 04 2014. doi: 10.1145/2523813. 15

[78] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Software Engineering*, PP, February 2021. ISSN 0098-5589. doi: 10.1109/TPAMI.2021.3057446. 16

[79] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, page 556–572, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-01251-9. doi: 10.1007/978-3-030-01252-6_33. URL `https://doi.org/10.1007/978-3-030-01252-6_33`. 16

[80] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: incremental classifier and representation learning. In *CVPR*, 2017. 16, 79, 80

[81] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420, 2019. 19, 82, 83, 88

[82] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930. Curran Associates, Inc., 2020. 16, 81, 82, 83, 88, 100

[83] Gido M. van de Ven and Andreas Savas Tolias. Three scenarios for continual learning. *ArXiv*, abs/1904.07734, 2019. 17, 19

[84] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4655–4665, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. 17, 20

[85] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf`. 17, 19

[86] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1803839115. URL `https://www.pnas.org/content/115/44/E10467`. 18, 20

[87] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, and Tiago Ramalho. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/content/114/13/3521`. 18, 82

[88] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress amp; compress: A scalable framework for continual learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4528–4537. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/schwarz18a.html`. 18, 82

[89] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International*

Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/zenke17a.html. 18

[90] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1): 23–63, 1987. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(87)80025-3. URL https://www.sciencedirect.com/science/article/pii/S0364021387800253. 19

[91] G.A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21 (3):77–88, 1988. doi: 10.1109/2.33. 19, 20

[92] J. Hertz, John, Krough, Anders Flisberg, Palmer, and Richard G. *Introduction To The Theory Of Neural Computation*, volume 44. 12 1991. doi: 10.1063/1.2810360. 20

[93] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/content/114/13/3521.

[94] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *ArXiv*, abs/1607.00122, 2016. 20

[95] ROBERT M. FRENCH. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 4(3-4):365–377, 1992. doi: 10.1080/09540099208946624. URL https://doi.org/10.1080/09540099208946624. 20

[96] Robert French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. *Proceedings of the 16th Annual Cognitive Science Society Conference*, 08 1994. 20

[97] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International*

Conference on International Conference on Machine Learning, ICML'11, page 513–520, Madison, WI, USA, 2011. Omnipress. 21

[98] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1081–1090. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/chen19i.html`. 21

[99] Sinno Jialin Pan, James T. Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, page 677–682. AAAI Press, 2008. ISBN 9781577353683. 21, 24, 88

[100] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. 21

[101] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. *Journal of Machine Learning Research*, 22:1453–1461, 2012. ISSN 1532-4435. 21

[102] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Mach. Learn.*, 79(1–2): 151–175, May 2010. 22, 23, 24

[103] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13 (25):723–773, 2012. URL `http://jmlr.org/papers/v13/gretton12a.html`. 23

[104] James M. Joyce. *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_327. URL `https://doi.org/10.1007/978-3-642-04898-2_327`. 23, 76

[105] Wei Wang, Hao Wang, Zhi-Yong Ran, and Ran He. Learning robust feature transformation for domain adaptation. *Pattern Recognition*, 114:107870, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.107870. URL `https://www.sciencedirect.com/science/article/pii/S0031320321000571`. 24

[106] Luís A.M. Pereira and Ricardo da Silva Torres. Semi-supervised transfer subspace for domain adaptation. *Pattern Recognition*, 75:235–249, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.04.011. URL `https://www.sciencedirect.com/science/article/pii/S0031320317301632`. Distance Metric Learning for Pattern Recognition. 24

[107] Lin Zuo, Mengmeng Jing, Jingjing Li, Lei Zhu, Ke Lu, and Yang Yang. Challenging tough samples in unsupervised domain adaptation. *Pattern Recognition*, 110:107540, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107540. URL `https://www.sciencedirect.com/science/article/pii/S0031320320303435`. 24

[108] Behzad Bozorgtabar, Dwarikanath Mahapatra, and Jean-Philippe Thiran. Exprada: Adversarial domain adaptation for facial expression analysis. *Pattern Recognition*, 100:107111, 2020. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2019.107111. URL `https://www.sciencedirect.com/science/article/pii/S0031320319304121`.

[109] Baoyao Yang and Pong C. Yuen. Learning adaptive geometry for unsupervised domain adaptation. *Pattern Recognition*, 110:107638, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107638. URL `https://www.sciencedirect.com/science/article/pii/S0031320320304416`.

[110] Jing Wang, Jiahong Chen, Jianzhe Lin, Leonid Sigal, and Clarence W. de Silva. Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by gaussian-guided latent alignment. *Pattern Recognition*, 116:107943, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.107943. URL `https://www.sciencedirect.com/science/article/pii/S0031320321001308`. 24

[111] Marcus de Carvalho, Mahardhika Pratama, Jie Zhang, and Edward Yapp. Acdc: Online unsupervised cross-domain adaptation, 2021. 24, 88

[112] Yun-Chun Chen, Yen-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang. Crdoco: Pixel-level domain transfer with cross-domain consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 24

[113] Tongkun Xu, Weihua Chen, Pichao Wang, Fan Wang, Hao Li, and Rong Jin. Cdtrans: Cross-domain transformer for unsupervised domain adaptation, 2021. URL `https://arxiv.org/abs/2109.06165`. 24, 100, 101

[114] Jichang Li, Guanbin Li, Yemin Shi, and Yizhou Yu. Cross-domain adaptive clustering for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2505–2514, June 2021. 24, 88

[115] Sheng Wu, Ancong Wu, and Wei-Shi Zheng. Online deep transferable dictionary learning. *Pattern Recognition*, 118: 108007, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108007. URL `https://www.sciencedirect.com/science/article/pii/S0031320321001941`. 25, 35

[116] Shiliang Sun, Honglei Shi, and Yuanbin Wu. A survey of multi-source domain adaptation. *Information Fusion*, 24:84–92, 2015. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2014.12.003. URL `https://www.sciencedirect.com/science/article/pii/S1566253514001316`. 25, 35

[117] Peilin Zhao and Steven C.H. Hoi. Otl: A framework of online transfer learning. 2010. 26

[118] Renchunzi Xie and Mahardhika Pratama. Automatic online multi-source domain adaptation. *Inf. Sci.*, 582:480–494, 2022. 26

[119] Geoffrey Hinton, Jeff Dean, and Oriol Vinyals. Distilling the knowledge in a neural network. pages 1–9, 03 2014. 28, 77

[120] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=Uuf2q9TfXGA`. 29

[121] Sihui Luo, Xinchao Wang, Gongfan Fang, Yao Hu, Dapeng Tao, and Mingli Song. Knowledge amalgamation from heterogeneous networks by common feature learning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao,*

China, August 10-16, 2019, pages 3087–3093. ijcai.org, 2019. doi: 10.24963/ijcai.2019/428. URL https://doi.org/10.24963/ijcai.2019/428. 29, 85

[122] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3068–3075, Jul. 2019. doi: 10.1609/aaai.v33i01.33013068. URL https://ojs.aaai.org/index.php/AAAI/article/view/4165.

[123] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. *ArXiv*, abs/1811.02796, 2019. 29

[124] Weiwei Weng, Mahardhika Pratama, Choiru Za'in, Marcus de Carvalho, Rakaraddi Appan, Andri Ashfahani, and Edward Yapp Kien Yee. Autonomous cross domain adaptation under extreme label scarcity. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2022. doi: 10.1109/TNNLS.2022.3183356. 30

[125] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013. 30

[126] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. 30, 49

[127] Chang'an Yi, Haotian Chen, Yonghui Xu, Yong Liu, Lei Jiang, and Haishu Tan. Atpl: Mutually enhanced adversarial training and pseudo labeling for unsupervised domain adaptation. *Knowledge-Based Systems*, 250:108831, 2022. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2022.108831. URL https://www.sciencedirect.com/science/article/pii/S0950705122003963. 30

[128] Hyeonwoo Cho, Kazuya Nishimura, Kazuhide Watanabe, and Ryoma Bise. Effective pseudo-labeling based on heatmap for unsupervised domain adaptation in cell detection. *Medical Image Analysis*, 79:102436, 2022. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2022.102436. URL https://www.sciencedirect.com/science/article/pii/S1361841522000871. 30

[129] Can Zhang and Gim Hee Lee. Ca-uda: Class-aware unsupervised domain adaptation with optimal assignment and pseudo-label refinement, 2022. URL `https://arxiv.org/abs/2205.13579`. 30

[130] Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14153–14172. PMLR, 17–23 Jul 2022. 30

[131] Kento Nishi, Yi Ding, Alex Rich, and booktitle = Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) month = June year = 2021 pages = 8022-8031 Höllerer, Tobias, title = Augmentation Strategies for Learning With Noisy Labels. 30

[132] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 30, 94, 112

[133] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. Symmetric cross entropy for robust learning with noisy labels. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 322–330, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society. doi: 10.1109/ICCV.2019.00041. URL `https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00041`. 30

[134] Mohammad M. Masud, Clay Woolam, Jing Gao, Latifur Khan, Jiawei Han, Kevin W. Hamlen, and Nikunj C. Oza. Facing the reality of data stream classification: Coping with scarcity of labeled data. *Knowl. Inf. Syst.*, 33(1):213–244, October 2012. 35

[135] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Mach. Learn.*, 90 (3):317–346, March 2013. 35

[136] Peilin Zhao, Steven C.H. Hoi, Jialei Wang, and Bin Li. Online transfer learning. *Artificial Intelligence*, 216:76–102, 2014. 35

[137] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 38

[138] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press. 42

[139] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020. 43

[140] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010. 45

[141] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. 49, 98

[142] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *ATT Labs*, 2010. 49, 81

[143] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 49, 81

[144] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `http://proceedings.mlr.press/v15/coates11a.html`. 49

[145] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. 49

[146] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. 49

[147] Hristo Mavrodiev. London bike sharing dataset, Oct 2019. URL `https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset`. 49

[148] Hadi Fanaee-T and João Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2:113–127, 2013. 49

[149] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018. URL `http://jmlr.org/papers/v19/18-251.html`. 63

[150] Charles J. Stone R.A. Olshen Leo Breiman, Jerome Friedman. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. 63

[151] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594. 73

[152] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning with double encoding-layer autoencoder for transfer learning. *ACM Trans. Intell. Syst. Technol.*, 9(2), October 2017. 74

[153] David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987. 75

[154] Andrew R. Liddle, Pia Mukherjee, and David Parkinson. *Model selection and multi-model inference*, page 79–98. Cambridge University Press, 2009. doi: 10.1017/CBO9780511802461.005. 76

[155] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97 2:285–308, 1990. 82

[156] Ari S. Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. 2018. doi: 10.48550/ARXIV.1805.08289. URL `https://arxiv.org/abs/1805.08289`. 82, 83

[157] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. *Gradient Based Sample Selection for Online Continual*

Learning. Curran Associates Inc., Red Hook, NY, USA, 2019. 82

[158] Arslan Chaudhry, Albert Gordo, Puneet K. Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning, 2020. URL `https://arxiv.org/abs/2002.08165`. 82, 83, 88, 100

[159] Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019. ISSN 2157-6904. doi: 10.1145/3293318. URL `https://doi.org/10.1145/3293318`. 83, 114

[160] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 171–181, 2022. doi: 10.1109/CVPR52688.2022.00027. 88

[161] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. 2021. URL `https://arxiv.org/abs/2104.05704`. 91

[162] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`. 91

[163] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. 91, 96

[164] Qicheng Lao, Xiang Jiang, Mohammad Havaei, and Yoshua Bengio. A two-stream continual learning system with variational domain-agnostic feature replay. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4466–4478, 2022. doi: 10.1109/TNNLS.2021.3057453. 96, 97, 104

[165] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017. 98

[166] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. 98

[167] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. volume 6314, pages 213–226, 09 2010. ISBN 978-3-642-15560-4. doi: 10.1007/978-3-642-15561-1_16. 98

[168] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019. 98

[169] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *ATT Labs*, 2010. 98

[170] Christian Simon, Masoud Faraki, Yi-Hsuan Tsai, Xiang Yu, Samuel Schulter, Yumin Suh, Mehrtash Harandi, and Manmohan Chandraker. On generalizing beyond domains in cross-domain continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9265–9274, June 2022. 100

[171] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 102

[172] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`. 102

[173] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning, 2019. 112

[174] Rohan Ramanath, Konstantin Salomatin, Jeffrey D. Gee, Kirill Talanine, Onkar Dalal, Gungor Polatkan, Sara Smoot, and Deepak Kumar. Lambda learner: Fast incremental learning on data streams. *CoRR*, abs/2010.05154, 2020. URL `https://arxiv.org/abs/2010.05154`. 113

[175] Colin Troisemaine, Vincent Lemaire, Stéphane Gosselin, Alexandre Reiffers-Masson, Joachim Flocon-Cholet, and Sandrine Vaton. Novel class discovery: an introduction and key concepts, 2023. 113

[176] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. Generalizing to unseen domains: A survey on domain generalization, 2022. 113

[177] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 114

[178] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis insights from training gopher, 2022. 114