UAB
Universitat Autònoma de Barcelona

Understanding the Embedding Space in Continual and Federated Learning

Dipam Goswami

UAB
Universitat Autònoma
de Barcelona

Understanding the Embedding Space in Continual and Federated Learning

A dissertation submitted by **Dipam Goswami**
at Universitat Autònoma de Barcelona to fulfill
the degree of **Doctor of Philosophy**.

Bellaterra, January 19, 2026

Director	**Dr. Joost van de Weijer** Centre de Visió per Computador Universitat Autònoma de Barcelona **Dr. Bartłomiej Twardowski** Centre de Visió per Computador Universitat Autònoma de Barcelona IDEAS Research Institute
Thesis committee	**Dr. Rahaf Aljundi** AI Core Research Toyota Motor Europe, Belgium **Dr. Dimosthenis Karatzas** Department of Computer Science Universitat Autònoma de Barcelona, Spain **Dr. Adrian Popescu** Laboratory for Integration of Systems and Technologies The French Alternative Energies and Atomic Energy Commission (CEA), France

This document was typeset by the author using LaTeX 2ε.

The research described in this book was carried out at the Centre de Visió per Computador, Universitat Autònoma de Barcelona.

ISBN: 978-84-128712-9-6

Printed by Imprima Sabadell

To the loving memory of my parents...

Acknowledgements

I would like to thank my supervisors Joost van de Weijer and Bartłomiej Twardowski for the opportunity to perform my PhD research at CVC and their constant support and guidance all throughout my PhD. I am extremely thankful to Joost for considering me for remote collaboration during my Masters in 2022 and then an in-person internship at CVC which helped me to take the decision of doing my PhD research in Spain. Right from the very beginning of my PhD, I realized the privilege of having both Joost and Bartek for research discussions and all sorts of things. While their expertise, feedback, and insights helped me grow as a researcher, spending the last few years with them inspired me well beyond my academic life. I am also thankful to them for giving me the opportunity to attend international conferences and the freedom to collaborate and spend time in international research labs and industry during my PhD. I have been fortunate to work with several incredible researchers during my PhD - Andrew D. Bagdanov, Tinne Tuytelaars, Gido van de Ven and Joan Serrà and I am so thankful to them for all the learnings.

The decision to do a PhD was not an easy one and would not have been possible without the constant backing of my family who stood with me through thick and thin. I dedicate my thesis to the memory of my mother and father whose love remains my foundation. I cannot imagine coming this far in my life without the sacrifices of my parents, brother, sister, uncle, and aunt, and the love and support of my entire family which kept me going.

I am thankful to all my co-authors - Yuyang, Albin, Simone, Sandesh, Kai, Alex, Liying, Marco and others with whom I enjoyed working so much. I still cherish my initial days of starting continual learning with Albin who introduced me to the field and inspired me a lot throughout my time here. To Yuyang, it has always been a pleasure to work with you right from the start of my journey in CVC. To Simone, thank you for all the motivation and sharing the pain of paper rejections, you are an admiration and I am so blessed to have you by my side. To Alex and Kai, thanks for all the guidance, it was always a pleasure to have you guys for any discussions. To Hector, thanks for all the help with the servers and for bearing with my impatience. To Shiqi, thanks for all the inspiration. I always enjoyed the research discussions with the members of the LAMP group during reading group meetings and lunches.

I am thankful to all my friends from CVC with whom I shared countless memories over the last three years. To my friend, philosopher and guide - Sounak, I am so blessed to have you in my life. I truly don't think I could

have stayed sane without your guidance and love. To Ayan and Khanh, thanks for the friendship and for sharing the struggles of expat life. I will always cherish the time we spent together. To Sanket, thanks for everything you did for me and for being an amazing friend. To Moha, thanks for all the love, inspiration, wisdom and support. To Ali and Andres, thanks for the friendship and all the memories we have shared, especially those lunches in CVC. To Sergi and Beatriz, thanks for all the calmness, love, friendship and memories. To Andrea, thanks for your friendship and all the cheerful vibes. To Pau, Artemis, Adri and Carlos, thanks for all the memories we had in CVC and in the padel court. To Andrey, thanks for the amazing time we spent hanging out in the city. To Kunal and Souparni, thanks for being amazing friends. To my CVC friends - Emanuele, German, Sandesh, Atif, David, Diego, Ramzy, Christos, Danna, Chuanming, Chunmeng, Lei, Fei, Laura, thank you for all the love and friendship. I am thankful to the CVC Professors - Javier, Dimos and Bogdan for all the inspiration and love. I deeply appreciate the constant support from all the CVC staff especially Gisele, Montse, Mireia, Joan, Marc and Kevin who helped me with so many things over the years.

I am thankful to all my friends from Poland for the wonderful memories I have with you guys - Filip, Daniel, Dawid, Stanislaw, Sebastian, Damian, Alicja, Marcin, Kamil, Jan and others. To Glen and Aniello, thanks for all the memories and friendship. To my colleagues from Sony - Alain, Eleonora, Uche, Fabio, Marc, it was a pleasure to share so much memories with you all. To my buddy Iustin, you are incredible and I am so lucky to have a friend like you. To Alexandra, thank you for all the love, friendship, support and for Dembele who is pawsitively the best. To Ayush, thanks for all the friendship and inspiration — from our time at BITS to this day. To Ankit, thanks for your support and for sharing so much of the journey and its struggles together. To Rishav, thanks for all the encouragement, love and friendship. To my friends with whom I started my journey in Spain - Manisha, Aarya, Soumya, Esha and George, thanks for the incredible time we had together. To my friends from BITS - Mayank, Bhuvesh, Nikhil and many more, thanks for all the countless memories and love. Finally, I want to thank everyone who has influenced me in so many ways over the years in my academic and personal life.

Abstract

Neural networks are extensively used throughout computer vision for a wide range of tasks, where the networks are typically trained on datasets in a single training session. However, enabling these networks to learn continually is challenging as new data arrive over time and models need to adapt accordingly. A critical aspect of continual learning is forgetting of previously learned knowledge after learning on new data. Another challenging paradigm for training neural networks is federated learning, where the data is distributed between multiple clients instead of a single source leading to class imbalance and data heterogeneity across clients.

This thesis investigates continual learning in an exemplar-free setting, which prohibits storing data from previous tasks. We explore the use of class prototypes for classification and investigate the use of Euclidean distance assuming isotropic feature representations. We demonstrate that the feature distributions of classes are not isotropic in a continual setting owing to the stability-plasticity dilemma. We discuss two major continual training practices: training on a bigger dataset in first task and freezing the network after the first task to continually learn only the classifier, and training the entire network at every new task. In the former setting, we propose to exploit the feature covariances to take into account the anisotropic distributions of the new classes which the model has not been trained on. In the latter setting, we discuss the issue of semantic drift in the embedding space and propose to generate adversarial samples from the new task data which can behave similar to the old task data. These generated samples are like pseudo-exemplars which are then used to track the movement of the class prototypes in the evolving embedding space.

Beyond image classification, we also study continual learning for information retrieval where the goal is to retrieve the most related document from a large corpus given a query document. We discuss the non-compatibility issue which stems from storing the corpus embeddings of old tasks beforehand and using the latest model to encode the queries during retrieval. We propose to move the query embeddings to the old embedding space during retrieval to compensate for the drift in the embedding space, thereby avoiding the need to re-index or compute the embeddings of the entire corpus every time the model is updated.

For federated learning, we explore the use of pre-trained feature extractors which achieve superior performance by only exploiting feature distributions

from clients in a training-free setting. We discuss how sharing second-order client statistics increases the communication budget and propose to estimate the second-order statistics at the server using only first-order statistics from clients with a provably unbiased covariance estimator. The proposed training-free approach dramatically reduces the communication cost while achieving a stable and more effective classifier initialization.
Keywords: *deep learning, computer vision, continual learning, federated learning, transfer learning, incremental learning, catastrophic forgetting, information retrieval*

Resumen

Las redes neuronales se utilizan ampliamente en la visión artificial para una amplia gama de tareas, en las que las redes suelen entrenarse con conjuntos de datos en una sola sesión de entrenamiento. Sin embargo, permitir que estas redes aprendan continuamente es un reto, ya que con el tiempo llegan nuevos datos y los modelos deben adaptarse en consecuencia. Un aspecto crítico del aprendizaje continuo es el olvido de los conocimientos adquiridos anteriormente tras el entrenamiento con nuevos datos. Otro paradigma desafiante para el entrenamiento de redes neuronales es el aprendizaje federado, en el que los datos se distribuyen entre múltiples clientes en lugar de una única fuente, lo que da lugar a un desequilibrio de clases y a una heterogeneidad de los datos entre clientes.

Esta tesis investiga el aprendizaje continuo en un entorno sin ejemplos, que prohíbe almacenar datos de tareas anteriores. Exploramos el uso de prototipos de clase para la clasificación e investigamos el uso de la distancia euclidiana asumiendo representaciones isotrópicas de características. Demostramos que las distribuciones de características de las clases no son isotrópicas en un entorno continuo debido al dilema entre estabilidad y plasticidad. Analizamos dos prácticas principales de entrenamiento continuo: el entrenamiento con un conjunto de datos más grande en la primera tarea y la congelación de la red después de la primera tarea para aprender continuamente solo el clasificador, y el entrenamiento de toda la red en cada nueva tarea. En el primer caso, proponemos aprovechar las covarianzas de las características para tener en cuenta las distribuciones anisotrópicas de las nuevas clases con las que el modelo no ha sido entrenado. En el segundo caso, analizamos el problema de la deriva semántica en el espacio de incrustación y proponemos generar muestras adversarias a partir de los datos de la nueva tarea que puedan comportarse de manera similar a los datos de la tarea anterior. Estas muestras generadas son como pseudoejemplos que luego se utilizan para rastrear el movimiento de los prototipos de clase en el espacio de incrustación en evolución.

Más allá de la clasificación de imágenes, también estudiamos el aprendizaje continuo para la recuperación de información, cuyo objetivo es recuperar el documento más similar a un documento de referencia dentro de un corpus

grande de documentos. Analizamos el problema de incompatibilidad que surge al almacenar previamente las incrustaciones del corpus de tareas antiguas y utilizar el modelo más reciente para codificar las consultas durante la recuperación. Proponemos trasladar las incrustaciones de la consulta al espacio de incrustación antiguo durante la recuperación para compensar la deriva en el espacio de incrustación, evitando así la necesidad de reindexar o calcular las incrustaciones de todo el corpus cada vez que se actualiza el modelo.

Para el aprendizaje federado, exploramos el uso de extractores de características preentrenados que logran un rendimiento superior al aprovechar únicamente las distribuciones de características de los clientes en un entorno sin entrenamiento. Analizamos cómo el intercambio de estadísticas de clientes de segundo orden aumenta la capacidad de comunicación y proponemos estimar las estadísticas de segundo orden en el servidor utilizando únicamente estadísticas de primer orden de los clientes con un estimador de covarianza demostrablemente imparcial. El enfoque sin entrenamiento propuesto reduce drásticamente el coste de comunicación, al tiempo que logra una inicialización del clasificador estable y más eficaz.

Palabras clave: *aprendizaje profundo, visión artificial, aprendizaje continuo, aprendizaje federado, aprendizaje por transferencia, aprendizaje incremental, olvido catastrófico, recuperación de información*

Resum

Les xarxes neuronals s'utilitzen àmpliament en la visió per computador per a una àmplia gamma de tasques, on les xarxes s'entrenen habitualment amb conjunts de dades en una única sessió d'entrenament. No obstant això, fer que aquestes xarxes aprenguin de manera contínua és un repte, ja que en aquest paradigma els models s'han d'adaptar a noves dades d'entrenament incorporades al llarg del temps. Un aspecte crític de l'aprenentatge continu és l'oblit del coneixement anteriorment après després d'entrenar amb noves dades. Un altre desafiament per a l'entrenament de xarxes neuronals és el paradigma de l'aprenentatge federat, on les dades estan distribuïdes entre diversos clients en lloc d'una única font, la qual cosa provoca desequilibri de classes i heterogeneïtat de dades entre clients.

Aquesta tesi investiga l'aprenentatge continu en un entorn sense exemples, un paradigma que prohibeix emmagatzemar dades de tasques anteriors. Explorem l'ús de prototips de classes per a la classificació i investiguem l'ús de la distància euclidiana assumint representacions de característiques isotròpiques. Demostrem que les distribucions de característiques de les classes no són isotròpiques en un entorn continu a causa del dilema estabilitat-plasticitat. Discutim dues pràctiques principals d'entrenament continu: Per una banda, entrenar la xarxa en una primera tasca amb un conjunt de dades gran i congelar-la, de forma que només s'entrena la capa de classificació de forma continua en les tasques successives. Per l'altra, entrenar tota la xarxa de nou per a cada nova tasca. En el primer cas, proposem aprofitar les covariàncies de les característiques per tenir en compte les distribucions anisotròpiques de les noves classes en què el model no ha estat entrenat. En el segon cas, analitzem la qüestió del desplaçament semàntic a l'espai d'*embedding* i proposem generar mostres adversarials a partir de les dades de la nova tasca que puguin comportar-se de manera similar a les de la tasca antiga. Aquestes mostres generades són pseudo-exemples que s'utilitzen per rastrejar el moviment dels prototips de classe en l'espai d'*embedding* en evolució.

Més enllà de la classificació d'imatges, també estudiem l'aprenentatge continu per a la recuperació d'informació, on l'objectiu és recuperar el document més similar a un document de referència dins d'un corpus gran de documents. Discutim la qüestió de la incompatibilitat que sorgeix de desar prèviament les representacions del corpus de les tasques antigues i d'utilitzar el model més recent per codificar les consultes durant la recuperació. Proposem desplaçar

les representacions de les consultes a l'espai de representacions antic durant la recuperació per compensar el desplaçament de l'espai de representacions, evitant així la necessitat de re-indexar o calcular les representacions de tot el corpus cada vegada que s'actualitza el model.

Per a l'aprenentatge federat, explorem l'ús d'extractors de característiques preentrenats que assoleixen un rendiment superior explotant només les distribucions de característiques dels clients en un entorn sense entrenament. Analitzem com el fet de compartir estadístiques de segon ordre dels clients augmenta el pressupost de comunicació i proposem estimar aquestes estadístiques de segon ordre al servidor utilitzant només estadístiques de primer ordre dels clients amb un estimador de covariància demostrablement no esbiaixat. L'enfocament proposat sense entrenament redueix dràsticament el cost de comunicació mentre aconsegueix una inicialització de classificador més estable i eficaç.

Paraules clau: *aprenentatge profund, visió per computador, aprenentatge continu, aprenentatge federat, aprenentatge per transferència, aprenentatge incremental, oblit catastròfic, recuperació d'informació*

Contents

1 Introduction

Deep learning has become one of the most influential domains in artificial intelligence, because of its ability to learn meaningful patterns from large amounts of data. Instead of relying on handcrafted rules, deep learning aims to automatically discover useful representations from the data. For instance, if we want a system to recognize cats in pictures, we do not need to provide details of possible shapes of a cat's ear or tail. We simply give it many pictures labeled "cat" or "not cat" and over time the model finds out by itself what features matter to recognize cats. This approach has enabled significant progress in tasks such as image recognition, natural language processing, information retrieval, translation, and image generation, among others, making deep learning foundational to technology across various domains.

Deep learning relies on neural networks, which are models made of multiple layers of interconnected units inspired by the way neurons work in the brain. Training a neural network means adjusting the connections between these units so that the network produces increasingly accurate predictions over time. This training process occurs through repetition - the network makes a guess, compares it to the correct answer, and then adjusts slightly to improve the results. After several of these small adjustments, the network becomes skilled at the task. However, deep learning networks usually need all training data at once. They learn everything at the same time to be able to identify patterns better and are not designed to learn new knowledge at a later time without forgetting already acquired knowledge.

To address this limitation, the continual learning paradigm has been introduced. In realistic scenarios, we rarely learn everything at once; we learn gradually from a stream of data. The challenge of continual learning is that when a model trained on some existing data learns something new, it often overwrites or forgets previously acquired knowledge, a problem referred to as *catastrophic forgetting*. Continual learning aims to understand this phenomenon and provide new methods and theory to address it. This ability to learn continually is critical for deep learning models which operate in the real

world, where new data constantly appear over time.

Finally, as neural networks become more integrated into modern technologies, a new concern arises: *data privacy*. Many applications like phones, healthcare systems, or smart environments contain sensitive personal data that cannot be shared onto a single central server for training on all data together. *Federated learning* addresses this situation by letting neural networks learn from data distributed across many different devices without the data ever leaving those devices. Each device trains their own neural network on its own data, and only these network updates, not the raw data, are shared and aggregated in a central location to achieve a superior network.

1.1 Background

Before discussing the main challenges addressed in this thesis, we briefly elaborate on the two main research fields on which this work focuses, namely continual learning and federated learning. For more extensive discussion, we refer to the following surveys: [25, 113, 185, 188, 201, 207].

1.1.1 Continual Learning

Deep learning has gained widespread use in various computer vision tasks, demonstrating exceptional performance when trained on a dataset in a single session. However, a significant challenge arises when new data is introduced incrementally in multiple phases or tasks, not at the same time. As a result, neural networks need to continually adapt without forgetting previously learned information. In *Continual Learning* (CL), the neural network is expected to accumulate knowledge from the ever-changing stream of new tasks data [25, 113, 185, 207].

We show in fig. 1.1, a continual learning setting in which the network learns on new data over time with the goal of good recognition of all the data categories seen so far. We consider multiple categories or classes of data in each task. When the network is naively trained on new tasks, the model performs poorly on old tasks - the model struggles to classify dogs after learning cats and birds due to the *catastrophic forgetting* phenomenon [114, 142]. Existing studies on continual learning explored different methods [35, 57, 96, 138, 150] that can help the model preserve old information while learning new information.

Continual learning [172] have been broadly classified into Class-Incremental Learning (CIL) and Task-Incremental Learning (TIL). In CIL, the objective is to incrementally learn new classes and achieve the highest accuracy for all

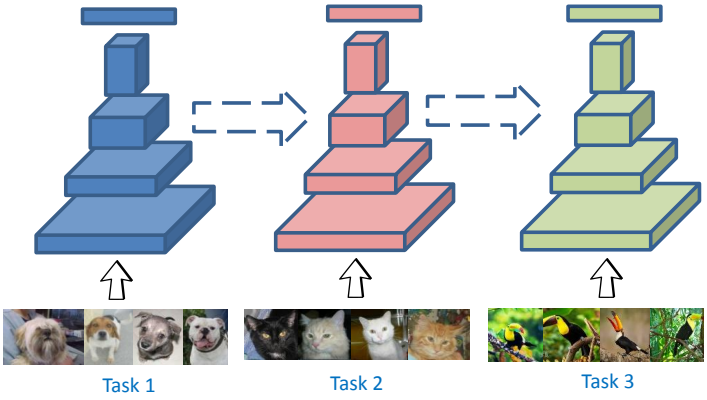


Figure 1.1: Illustration of continual learning scenario where the neural network is updated on new tasks over time (dogs -> cats -> birds). The goal is to continually learn in such a manner that the network does not forget the information from old tasks after learning on the new task. After learning the network on birds, we want the network to be able to recognize all the categories (dogs, cats and birds) well.

classes seen so far in a task-agnostic way without knowing which tasks the evaluated samples are from. In TIL, access to task-id is available at inference time and thus the objective is to classify the given image to only classes present in that particular task. In this thesis, we consider the CIL setting for image classification tasks and the TIL setting for image retrieval tasks.

One of the simplest approaches to mitigate forgetting is storing exemplars (a subset of images for each class from all old tasks) [7, 31, 35, 138, 180, 190, 193, 208, 210]. These exemplars are later replayed with the current task data during training in new tasks. While most initial works used raw image samples from old tasks as exemplars, some works [66, 100, 131, 171] also explored replaying features and intermediate representations from old tasks.

Although effective, these methods necessitate storing input data from previous tasks, leading to multiple challenges in practical settings such as legal concerns with new regulations (e.g. European GDPR where users can request to delete personal data), and privacy issues when dealing with sensitive data like in medical imaging. With an increasing number of tasks, maintaining a representative exemplar set requires significant memory, which subsequently increases the computational load due to training on a larger rehearsal dataset. Furthermore, in certain cases where large-scale pre-trained models are used, we generally do not have access to the pre-training data and thus no exemplars to

prevent forgetting of existing knowledge. Hence, the focus has shifted towards more realistic and challenging *exemplar-free* CIL (EFCIL) methods [45, 111, 134, 199, 212–214] where the model only has access to the data from the current task, making it even more susceptible to *catastrophic forgetting* of previously learned knowledge.

Another crucial aspect of continual learning is semantic drift in feature representations [199] after the model is trained on new data. This implies that the embeddings for a given image will be in a different location in the updated embedding space compared to the old embedding space. Semantic drift affects commonly used embedding networks [24, 117] that map images to different positions in the embedding space (where distances correspond to semantic dissimilarities between data points) since the prototypes of old classes stored from previous tasks are outdated in the updated embedding space. In the context of EFCIL settings, we will discuss continual learning across different settings where either the entire network is updated at every task leading to semantic drift or only the classifier is updated while keeping the backbone of the network frozen.

1.1.2 Federated Learning

Federated learning (FL) is a widely used paradigm for distributed learning across multiple clients. In FL, each client trains their local model on their private data and then sends model updates to a common global server that aggregates this information into a global model. The objective is to learn a global model that performs similarly to a model jointly trained on all the client data. We illustrate in fig. 1.2, an FL scenario considering two classes in each client where the local models are aggregated to obtain the server model which is then shared to the clients. This aggregation of models is performed iteratively for several rounds to obtain a superior model at the server which learns from the data of all clients.

A major concern in existing FL algorithms [115] is the poor performance when the client data is not identically and independently distributed (iid) or when classes are imbalanced between clients [1, 69, 92, 204]. In [106], the authors showed that client drift in FL is mainly due to drift in client classifiers which optimize to local data distributions, resulting in forgetting knowledge from clients of previous rounds [18, 89]. Another challenge in FL is the partial participation of clients in successive rounds based on client availability [92], which becomes particularly acute with large numbers of clients [68, 144]. To address these challenges, recent works have focused on algorithms to better tackle data heterogeneity across clients using pre-trained models [38, 88, 106, 164].

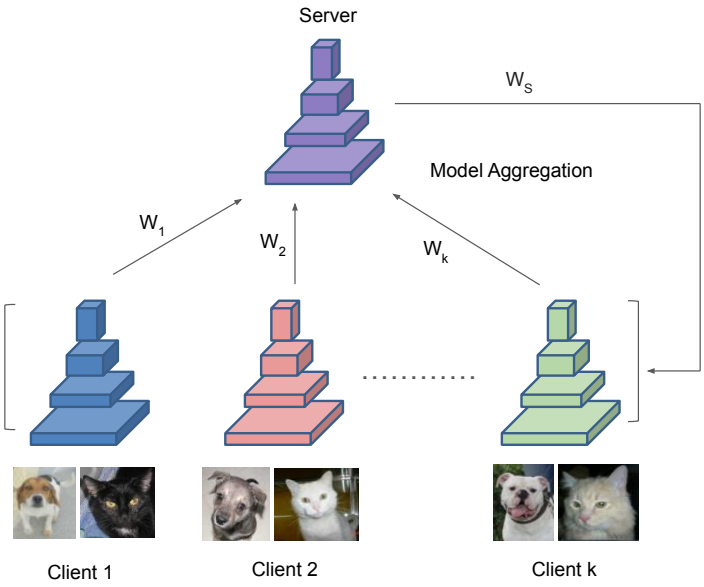


Figure 1.2: Illustration of a federated learning setting where multiple models $(W_1, W_2, .., W_k)$ trained on the private data of each client are aggregated in the central server. The updated model W_S from the server is sent back to the clients which are then used as the base model. The models are trained locally for few epochs in each client and then shared to the server to obtain an aggregated model which are then sent back to clients and used for local updates in the next round. This iterative local training and global aggregation is a prominent approach in FL known as FedAvg [115].

Most existing work on federated learning considers training the model from scratch [1,69,92,115,204] and only recently federated learning work investigated training from pre-trained models [38,88,125,189]. Initializing client models with the same pre-trained weights has been found to be effective in reducing data heterogeneity and achieved faster convergence [22,88,125] due to a more stable initialization. It has also been found to reduce the performance gap between a federated and a centralized setting. In this thesis, we discuss federated learning in the context of pre-trained models and explore methods to make it more communication-efficient.

1.2 Challenges in Continual and Federated Learning

Here we elaborate on the main challenges that we have identified in this thesis.

1.2.1 Classifier-Incremental Learning

One of the critical issues in continual learning is the stability-plasticity dilemma which refers to the challenge of learning new information (plasticity) without overwriting what was learned before (stability). If a model is too plastic, it easily forgets knowledge from previous tasks and learns new tasks very well; if it is too stable, it struggles to learn new tasks. In EFCIL settings, some approaches [158, 211–213] update the entire network on every task favoring more plasticity while several methods [6, 30, 129, 134] propose to freeze the feature extractor after learning good representations on the first task and only update the classifier at every task to learn the new classes. We refer to the latter setting as *classifier-incremental learning* since the continual learning dynamics apply only to the classifier.

Several studies [26, 65, 129, 132, 199, 205] explore using class-wise prototypes for classification in CIL, which generate new class prototypes using the frozen feature extractor and classify the test image features based on the Euclidean distance to the prototypes. This classifier is often referred to as the Nearest-Class-Mean (NCM) classifier and its success is contributed to the fact that class distributions can be well approximated by an isotropic distribution after training [117]. *In chapter 2 of this thesis, we investigate whether it is still optimal to assume isotropic representations and use the euclidean distance for classification in CIL settings since the frozen model never learns isotropic representations of new classes.*

1.2.2 Continual Representation Learning

While *classifier-incremental learning* approaches work well when the first task is big (having diverse representative data) or when we start from a pre-trained network with very good representations, these methods struggle when the first task is not very big and the model does not learn very strong feature representations. As a result, the model needs to be updated in new tasks to learn better feature representations. We refer to this setting as *continual representation learning* since the entire network is updated at every task to learn new feature representations continually.

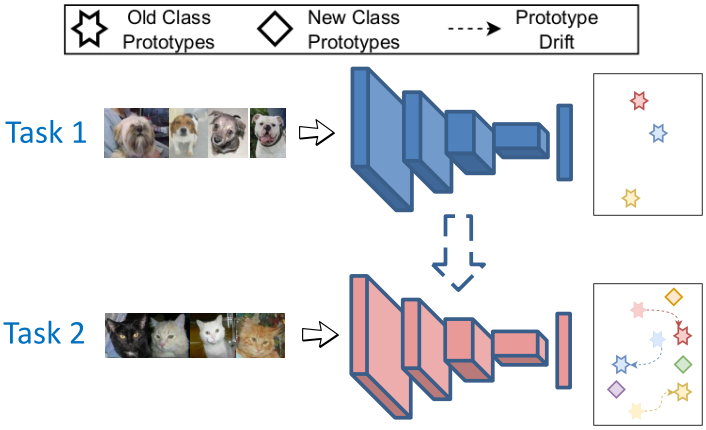


Figure 1.3: Illustration of semantic drift in continual learning settings where the old class prototypes drift to different positions in the embedding space after learning on new tasks. Due to the drift in the embedding space, old class prototypes stored from previous tasks are obsolete and we need to estimate their updated positions in the embedding space to improve their classification performance.

The continually evolving feature embedding space results in semantic drift [199]. As a consequence, the stored class prototypes or class means move to different positions after training on a new task. We illustrate this phenomenon of *semantic drift* in the embedding space in fig. 1.3 where the old class prototype positions need to be updated in order to achieve good performance on old classes. One way of estimating the updated prototype positions in the new embedding space is by using exemplars. Existing exemplar-free methods to estimate semantic drift are based on the unrealistic assumption that the drift of current task data is similar to past data drift [199]. *In chapter 3 of this thesis, we explore how to estimate the drift in old class prototypes by adapting current task data to mimic old task data. We investigate how to correct the prototype positions in the updated embedding space during continual representation learning.*

1.2.3 Continual Learning for Information Retrieval

Text embedding models enable semantic search, powering several NLP applications [64, 90, 136] like Retrieval Augmented Generation (RAG) by efficient information retrieval (IR). IR methods generally encode a huge corpus of

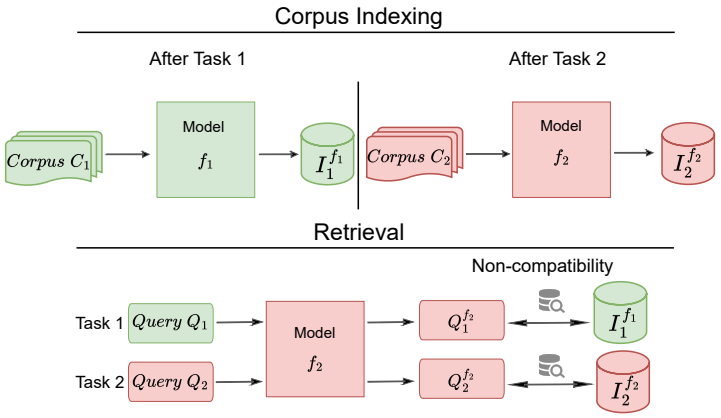


Figure 1.4: Illustration of continual learning for information retrieval. There exists non-compatibility between query embeddings $Q_1^{f_2}$ from task 1 and the previously indexed corpus document embeddings $I_1^{f_1}$ since they are obtained using different models.

documents (partitioning them in chunks of sentences) to low-dimensional embeddings and store them in a database index. During retrieval, a semantic search over the corpus is performed and the document whose embedding is most similar to the query embedding is returned. However, text embedding models are commonly studied in scenarios where the training data is static, thus limiting its applications to dynamic scenarios where new training data emerges over time. For such scenarios, continually updating the embedding model could lead to considerable performance gains.

When updating an embedding model with new training data, using the already indexed corpus is suboptimal due to the *non-compatibility* issue [12, 137, 179], since the model which was used to obtain the embeddings of the corpus has changed due to updates on new tasks (see fig. 1.4). While re-indexing of old corpus documents using the updated model enables compatibility, it requires much higher computation and time. *In chapter 4 of this thesis, we analyze continual learning for information retrieval and explore how to continually perform retrieval without any expensive re-indexing of millions of documents over time.*

1.2.4 Training-free Federated Learning

Several recent works on FL [21, 106, 125, 135, 156, 163, 164, 189] started using strong pre-trained feature extractors. Using the pre-trained representations

reduces the effect of data heterogeneity across clients and allows for faster convergence. This significantly reduces the computation and communication costs compared to federated optimization from scratch. Recent work [88] shows that sharing class means from clients to the server and using the aggregated global class means for classifier initialization achieves very high performance without any training. This could be further improved by sharing second-order feature statistics from clients and using them to initialize the global classifier [5, 38]. These training-free methods often outperform training-based FL methods like FedAvg [115] and FedAdam [139].

While these methods perform well, the sharing of high-dimensional second-order statistics incur very high communication costs. The communication costs scale linearly with the number of clients and quadratically with the feature dimensionality. *In chapter 5 of this thesis, we explore how to make the training-free FL methods more communication-efficient by estimating second-order feature statistics from only the client class means.*

1.3 Objectives and approach

In this thesis, we explore several aspects of continual and federated learning and propose solutions addressing existing limitations. We discuss continual learning for image classification and information retrieval applications in the *exemplar-free* setting and federated learning for image classification. Here, we define the objectives of the chapters and the proposed solutions for the problems discussed above.

1.3.1 Effective Classifiers for Classifier-Incremental Learning

Exemplar-free class-incremental learning (CIL) poses several challenges since it prohibits the rehearsal of data from previous tasks and thus suffers from catastrophic forgetting. Recent approaches to incrementally learning the classifier (by freezing the feature extractor after the first task) have gained much attention. Here, we analyze the use of class prototypes for EFCIL underlying the assumption that the feature representations of all classes are isotropic. We summarize the contributions of chapter 2 as follows:

> **Looking Beyond Euclidean Classifiers in Continual Learning:** In an analysis of the feature distributions of classes, we show that classification based on Euclidean metrics is successful for jointly trained features. However, when learning from non-stationary data,

we observe that the Euclidean metric is suboptimal and that feature distributions are heterogeneous. To address this challenge, we revisit the anisotropic Mahalanobis distance for CIL. In addition, we empirically show that modeling the feature covariance relations is better than previous attempts at sampling features from normal distributions and training a linear classifier. Unlike existing methods, our approach generalizes to both many- and few-shot CIL settings, as well as to domain-incremental settings.

1.3.2 Drift Compensation for Continual Representation Learning

Continual learning methods are known to suffer from catastrophic forgetting, a phenomenon that is particularly hard to counter for methods that do not store exemplars of previous tasks. Therefore, to reduce potential drift in the feature extractor, existing exemplar-free methods are typically evaluated in settings where the first task is significantly larger than subsequent tasks to start with high-quality feature representations. Their performance drops drastically in more challenging EFCIL settings starting with a smaller first task where the model needs to learn new feature representations continually. We address this problem of feature drift estimation in chapter 3 as follows:

Adversarial Drift Compensation for Continual Prototype Updates: We propose to adversarially perturb the current samples such that their embeddings are close to the old class prototypes in the old model embedding space. We then estimate the drift in the embedding space from the old to the new model using the perturbed images and compensate the prototypes accordingly. The generation of these images is simple and computationally cheap. We demonstrate in our experiments that the proposed approach better tracks the movement of prototypes in embedding space and outperforms existing methods on several standard continual learning benchmarks as well as on fine-grained datasets.

1.3.3 Re-Indexing Free Continual Document Retrieval

In continual learning for information retrieval, it is critical to address the *non-compatibility* issue and study how the already indexed corpus can still be effectively used without expensive re-indexing. The goal is to enable compatibility between the already indexed embeddings from corpus of previous tasks

and the query embeddings of previous tasks which are obtained using the latest updated model during retrieval. We summarize the contributions of chapter 4 as follows:

Enabling Compatibility in Continual Document Retrieval: We employ embedding distillation on both query and document embeddings to maintain stability and propose a query drift compensation method during retrieval to project new model query embeddings to the old embedding space. This enables compatibility with previously indexed corpus embeddings extracted using the old model and thus reduces the forgetting. We show that the proposed method significantly improves performance without any re-indexing on a continual learning benchmark with large-scale datasets.

1.3.4 Communication-efficient Federated Learning

Using pre-trained models has been found to reduce the effect of data heterogeneity and speed up federated learning algorithms. Recent works have explored training-free methods using first- and second-order statistics to aggregate local client data distributions at the server and achieve high performance without any training. While using second-order statistics improves performance, it also increases the communication costs significantly due to the very high dimensionality of the matrices. In this chapter, we explore how to estimate the second-order statistics based on only first-order statistics from the clients. We summarize the contributions of chapter 5 as follows:

Towards Communication-efficient Federated Learning. We propose a training-free method based on an unbiased estimator of class covariance matrices which only uses first-order statistics in the form of class means communicated by clients to the server. We show how these estimated class covariances can be used to initialize the global classifier, thus exploiting the covariances without actually sharing them. We also show that using only within-class covariances results in a better classifier initialization. Our approach improves performance in the range of 4-26% with exactly the same communication cost when compared to methods sharing only class means and achieves performance competitive or superior to methods sharing second-order statistics with dramatically less communication overhead. The proposed method is much more communication-efficient than federated prompt-tuning methods and still outperforms them.

Finally, using our method to initialize classifiers and then performing federated fine-tuning or linear probing yields better performance.

2 FeCAM: Exploiting the Heterogeneity of Class Distributions in Exemplar-Free Continual Learning*

2.1 Introduction

In exemplar-free CIL setting, the challenge is to discriminate between old and new classes without access to old data. While some methods [158, 211–213] trained the model on new classes favoring plasticity and used knowledge distillation [55] to preserve old class representations, other methods [6, 30, 107, 129, 134] froze the feature extractor after the first task, thus favoring stability and incrementally learned the classifier. One of the drawbacks is the inability to learn new representations with a frozen feature extractor. Inspired by transfer learning [152], the objective of these classifier-incremental methods is to make best use of the learned representations from the pretrained model and continually adapt the classifier to learn new tasks. Recently, pretrained feature extractors have been used in exemplar-free CIL by prompt-based methods [187], using linear discriminant analysis [129] and with a simple nearest class mean (NCM) classifier [65]. These methods use a transformer model pretrained on large-scale datasets like ImageNet-21k [140] and solely focus on classifier-incremental learning.

This chapter investigates methods to enhance the representation of class prototypes in CIL, aiming to improve plasticity within the stability-favoring classifier-incremental setting. A standard practice in few-shot CIL [99, 132, 202, 205, 209] is to obtain the feature embeddings of new class samples and average them to generate class-wise prototypes. The test image features are then classified by computing the Euclidean distance to the mean prototypes. The Euclidean distance is used in the NCM classifier, following [50], which claims that the highly non-linear nature of learned representations eliminates the need to learn the Mahalanobis [27, 191] metric previously used [117]. Our analysis shows that this holds true for classes that are considered during training, however, for new classes, the Euclidean distance is suboptimal. To address this

*This chapter is based on a publication in Advances in Neural Information Processing Systems, 2023 [45].

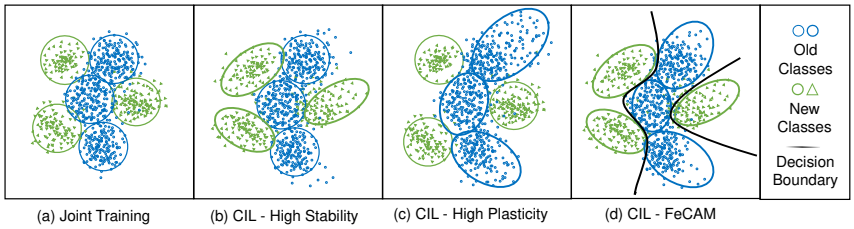


Figure 2.1: Illustration of feature representations in CIL settings. In Joint Training (a), deep neural networks learn good isotropic spherical representations [50] and thus the Euclidean metric can be used effectively. However, it is challenging to learn isotropic representations of both old and new classes in CIL settings. When the model is too stable in (b), it is unable to learn good spherical representations of new classes and when it is too plastic in (c), it learns spherical representations of new classes but loses the spherical representations of old classes. Thus, it is suboptimal to use the isotropic euclidean distance. We propose FeCAM in (d) which models the feature covariance relations using Mahalanobis metric and learns better non-linear decision boundaries for new classes.

problem, we propose to use the anisotropic Mahalanobis distance. In fig. 2.1, we explain how the feature representations vary in CIL settings. Here, the high-stability case in CIL is explored, where the model does not achieve spherical representations for new classes in the feature space, unlike joint training. Thus, it is intuitive to take into account the feature covariances while computing the distance. The covariance relations between the feature dimensions better captures the more complex class structure in the high-dimensional feature space. Additionally, in fig. 2.3, we analyze singular values for old and new class features to observe the changes in variances in their feature distributions, suggesting a shift towards more anisotropic representations.

While previous methods [117] proposed learning Mahalanobis metrics, we propose using an optimal Bayes classifier by modeling the covariance relations of the features and employing class prototypes. We term this approach **Fe**ature **C**ovariance-**A**ware **M**etric (FeCAM). We compute the covariance matrix for each class from the feature embeddings corresponding to training samples and perform correlation normalization to ensure similar variances across all class representations, which is crucial for distance comparisons. We investigate various ways of using covariance relations in continual settings. We posit that utilizing a Bayes classifier enables better learning of optimal decision boundaries compared to previous attempts [194] involving feature sampling from Gaussian

distributions and training linear classifiers. The proposed approach is simple to implement and requires no training since we employ a Bayes classifier. The Bayes classifier FeCAM can be used for both many-shot CIL and few-shot CIL, unlike existing methods. Additionally, we achieve superior performance with pretrained models on both class-incremental and domain-incremental benchmarks.

2.2 Related Work

Many-shot class-incremental learning (MSCIL) is the conventional setting, where sufficient training data is available for all classes. A critical aspect in many-shot CIL methods is the semantic drift in the feature representations [199] while training on new tasks. While recent methods use knowledge distillation [55] or regularization strategies [76, 199] to maintain the representations of old classes, these methods are dependent on storing images [10, 11, 19, 31, 35, 57, 73, 96], representations [66] or instances [102] from old tasks and becomes ineffective in practical cases where data privacy is required. Another set of methods [6, 30, 107, 129, 134] proposed freezing the feature extractor after the first task and learning only the classifier on new classes. We follow the same setting and do not violate privacy concerns by storing exemplars.

On the other hand, Few-Shot Class-Incremental Learning (FSCIL) considers that very few (1 or 5) samples per class are available for training [99, 165, 177]. Generally, these settings assume a big first task and freezes the network after training on the initial classes. To address the challenges of FSCIL, various techniques have been proposed. One approach is to use meta-learning to learn how to learn new tasks from few examples [9,123,146,169]. Another approach is to incorporate variational inference to learn a distribution over models that can adapt to new tasks [124]. Most FSCIL methods [2, 23, 99, 132, 165, 202, 205, 209] obtains feature embeddings of a small number of examples and averages them to get the class-wise prototypes. These methods uses the Euclidean distance to classify the features assuming equally spread classes in the feature space. We explore using the class prototypes in both MSCIL and FSCIL settings.

Since the emergence of deep neural networks, Euclidean distance is used in NCM classifier following [50], instead of Mahalanobis distance [117]. Mahalanobis distance has been recently explored for out-of-distribution detection with generative classifiers [87] and an ensemble using Mahalanobis [73] and also within the context of cross-domain continual learning [157].

2.3 Proposed Approach

2.3.1 Motivation

For the classification of hand-crafted features, Mensink *et al.* [117] proposed the nearest class mean (NCM) classifier using (squared) Mahalanobis distance $\mathcal{D}_M$ instead of Euclidean distance to assign an image to the class with the closest mean (see also illustration in fig. 2.2) :

$$y^* = \underset{y=1,\dots,Y}{\arg\min}\, \mathcal{D}_M(x, \mu_y), \quad \mathcal{D}_M(x, \mu_y) = (x - \mu_y)^T M (x - \mu_y) \tag{2.1}$$

where Y is the number of classes, $x, \mu_y \in \mathbb{R}^D$, class mean $\mu_y = \frac{1}{|X_y|} \sum_{x \in X_y} x$, and M is a positive definite matrix. They learned a low-rank matrix $M = W^T W$ where $W \in \mathbb{R}^{m \times D}$, with $m \leq D$.

However, with the shift towards deep feature representations, Guerriero *et al.* [50] assert that the learned representations with a deep convolutional network $\phi : \mathcal{X} \to \mathbb{R}^D$, eliminate the need of learning the Mahalanobis metric M and the isotropic Euclidean distance $\mathcal{D}_e$ can be used as follows:

$$y^* = \underset{y=1,\dots,Y}{\arg\min}\, \mathcal{D}_e(\phi(x), \mu_y), \quad \mathcal{D}_e(\phi(x), \mu_y) = (\phi(x) - \mu_y)^T (\phi(x) - \mu_y) \tag{2.2}$$

where $\phi(x), \mu_y \in \mathbb{R}^D$, $\mu_y = \frac{1}{|X_y|} \sum_{x \in X_y} \phi(x)$ is the class prototype or the average feature vector of all samples of class y. Here, $\phi(x)$ is the feature vector corresponding to image x and could be the output of penultimate layer of the network. In Euclidean space, $M = I$, where I is an identity matrix.

The success of the NCM classifier for deep learned representation (as observed by [50]) has also been adopted by the incremental learning community. The NCM classifier with Euclidean distance is now commonly used in incremental learning [26, 132, 138, 199, 202, 205, 209, 212]. However, in incremental learning, we do not jointly learn the features of all data, but are learning on a non-static data stream. As a result, the underlying learning dynamics which result in representations on which Euclidean distances perform excellently might no longer be valid. Therefore, we perform a simple comparison with the Euclidean and Mahalanobis distance (see fig. 2.4) where we use a network trained on 50% of classes of CIFAR100 (identified as *old classes*). Interestingly, indeed the *old classes*, on which the feature representation is learned, are very well classified with Euclidean distance, however, for the *new classes* this no

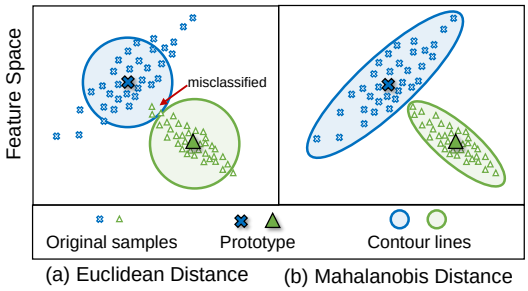
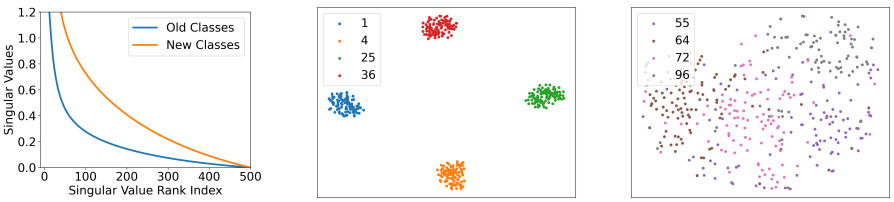


Figure 2.2: Illustration of distances (contour lines indicate points at equal distance from prototype).



(a) Singular values (b) Old classes (c) New classes

Figure 2.3: (a) Singular values comparison for old and new classes, (b-c) Visualization of features for old classes and new classes by t-SNE, where the colors of points indicate the corresponding classes.

longer holds, and the Mahalanobis distance obtains far superior results.

As a second experiment, we compare singular values of *old* and *new classes* (see fig. 2.3a). We observe that the singular values of new class features vary more and are in general larger than those of old classes. This points out that new class distributions are more heterogeneous (and are more widely spread) compared to the old classes and hence the importance of a heterogeneous distance measure (like Mahalanobis). This is also confirmed by a t-SNE plot of both old and new classes (see fig. 2.3b,c) showing that the new classes, which were not considered while training the backbone are badly modeled by a spherical distribution assumption, as is underlying the Euclidean distance.

Based on these results, two extreme cases of CIL are illustrated in fig. 2.1, considering maximum stability where the backbone is frozen, and maximum plasticity where the training is done with fine-tuning without preventing for-

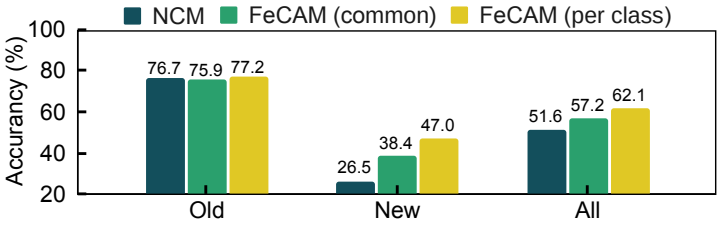


Figure 2.4: Accuracy Comparison of NCM (Euclidean) and FeCAM (Mahalanobis) using common covariance matrix and a matrix per class on CIFAR100 50-50 (2 task) sequence, for Old, New, and All classes at the end of the learning sequence.

getting. In this chapter, we revisit the nearest class mean classifier based on a heterogeneous distance measure. We perform this for classifier incremental learning, where the backbone is frozen after the first task. However, we think that conclusions based on the heterogeneous nature of class distributions also have consequences for continual deep learning where the representations are continually updated.

2.3.2 FeCAM Classifier

When modeling the feature distribution of classes with a multivariate normal feature distribution $\mathcal{N}(\mu_y, \mathbf{\Sigma}_y)$, the probability of a sample feature x belonging to class y can be expressed as,

$$P(x|C=y) \approx \exp \frac{-1}{2}(x-\mu_y)^T \mathbf{\Sigma}_y^{-1}(x-\mu_y), \tag{2.3}$$

It is straightforward to see that this is the optimal Bayesian classifier, since:

$$\arg\max_y P(Y|X) = \arg\max_y \frac{P(X|Y)P(Y)}{P(X)} = \arg\max_y P(X|Y)P(Y)$$
$$\arg\max_y P(X|Y)P(Y) = \arg\max_y P(X|Y) \tag{2.4}$$

Since logarithm is a concave function $\arg\max_y P(X|Y) = \arg\max_y logP(X|Y)$

$$\arg\max_y logP(X|Y) = \arg\max_y \{-(x-\mu_y)^T \mathbf{\Sigma}_y^{-1}(x-\mu_y)\}$$
$$= \arg\min_y \mathcal{D}_M(x, \mu_y) \tag{2.5}$$

where the squared mahalanobis distance $\mathcal{D}_M(x, \mu_y) = (x-\mu_y)^T \mathbf{\Sigma}_y^{-1}(x-\mu_y)$.

In the following, we elaborate on several techniques that can be applied to improve and stabilize Mahalanobis-based distance classification. We apply these, resulting in our FeCAM classifier (an ablation study in section section 2.4.2 confirms the importance of these on overall performance).

Covariance Matrix Approximation. We obtain the covariance matrix from the feature vectors $\phi(x)$ corresponding to the samples x at any task. The covariance matrix can be obtained in different ways. A common covariance matrix $\mathbf{\Sigma}^{1:t}$ can be incrementally updated as the mean covariance matrix for all seen classes till task t as follows:

$$\mathbf{\Sigma}^{1:t} = \mathbf{\Sigma}^{1:t-1} \cdot \frac{|Y^{1:t-1}|}{|Y^{1:t}|} + \mathbf{\Sigma}^t \cdot \frac{|Y^{1:t}| - |Y^{1:t-1}|}{|Y^{1:t}|} \tag{2.6}$$

where $\mathbf{\Sigma}^t$ refers to the common covariance matrix obtained from the features of samples $x \in X^t$ from all classes seen in task t and $|.|$ denotes the number of classes seen till that task. This common covariance matrix will represent the feature distribution for all seen classes and requires storing only the common covariance matrix from the last task.

The other alternative is to use a covariance matrix $\mathbf{\Sigma}_y$ for $y \in 1, ..Y$ to represent the feature distribution of each class separately. Here, $\mathbf{\Sigma}_y$ is the covariance matrix obtained from the feature vectors of all the samples $x \in X_y$. This will involve storing a separate matrix for all seen classes.

Normalization of Covariance Matrices. The covariance matrix $\mathbf{\Sigma}_y$ obtained for each class will have different levels of scaling and variances along different dimensions. Particularly, due to the notable shift in feature distributions between the old and new classes, the variances are much higher for the new classes. As a result, the Mahalanobis distance of features from different classes will have different scaling factors, and the distances will not be comparable. So, in order to be able to use a covariance matrix per class, we perform a correlation matrix normalization on all the covariance matrices. In order to make the multiple covariance matrices comparable, we make their diagonal elements equal to 1. A normalized covariance matrix can be obtained as:

$$\hat{\mathbf{\Sigma}}_y(i,j) = \frac{\mathbf{\Sigma}_y(i,j)}{\sigma_y(i)\sigma_y(j)}, \quad \sigma_y(i) = \sqrt{\mathbf{\Sigma}_y(i,i)}, \quad \sigma_y(j) = \sqrt{\mathbf{\Sigma}_y(j,j)} \tag{2.7}$$

where $\sigma_y(i)$ and $\sigma_y(j)$ refers to the standard deviations along the dimensions i and j respectively.

We identify the difficulties of obtaining an invertible covariance matrix in cases when the number of samples are less than the number of dimensions. So, we use a covariance shrinkage method to get a full-rank matrix. We also show that a simple gaussianization of the features using tukey's normalization [170] is also helpful.

Covariance Shrinkage. When the number of samples available for a class is less than the number of feature dimensions, the covariance matrix $\mathbf{\Sigma}$ is not invertible, and thus it is not possible to use $\mathbf{\Sigma}$ to get the Mahalanobis distance. This is a very serious problem since most of the deep learning networks have large number of feature dimensions [200]. Similar to [82, 174], we perform a covariance shrinkage to obtain a full-rank matrix as follows:

$$\mathbf{\Sigma}_s = \mathbf{\Sigma} + \gamma_1 V_1 I + \gamma_2 V_2 (1 - I), \tag{2.8}$$

where V_1 is the average diagonal variance, V_2 is the average off-diagonal covariance of $\mathbf{\Sigma}$ and I is an identity matrix.

Tukey's Ladder of Powers Transformation. Tukey's Ladder of Powers transformation aims to reduce the skewness of distributions and make them more Gaussian-like. We transform the feature vectors $\phi(x)$ using Tukey's Ladder of Powers transformation [170]. It can be formulated as:

$$\tilde{\phi(x)} = \begin{cases} \phi(x)^\lambda & \text{if } \lambda \neq 0 \\ \log(\phi(x)) & \text{if } \lambda = 0 \end{cases} \tag{2.9}$$

where λ is a hyperparameter to decide the degree of transformation of the distribution. In our experiments, we use $\lambda = 0.5$ following [194].

We obtain the normalized features using Tukey's transformation and then use the transformed features to obtain the covariance matrices. When using multiple matrices, we perform correlation normalization to make them comparable. The final prediction and the squared Mahalanobis distance to the different class prototypes using one covariance matrix per class can be obtained as:

$$y^* = \underset{y=1,\ldots,Y}{\arg\min}\, \mathcal{D}_M(\phi(x), \mu_y), \quad \mathcal{D}_M(\phi(x), \mu_y) = (\tilde{\phi(x)} - \tilde{\mu}_y)^T (\hat{\mathbf{\Sigma}}_y)_s^{-1} (\tilde{\phi(x)} - \tilde{\mu}_y) \tag{2.10}$$

where $\tilde{\phi(x)}$ and $\tilde{\mu}_y$ refers to the tukey transformed features and prototypes

respectively and $(\hat{\mathbf{\Sigma}}_{\mathbf{y}})_s^{-1}$ denotes the inverse of the covariance matrices which first undergoes shrinkage followed by normalization. Note that the covariance matrices are computed using the tukey transformed features. Similarly, the common covariance matrix $\mathbf{\Sigma}^{1:t}$ can be used in eq. (2.10).

In algorithm 1, we present the pseudo code for using FeCAM classifier.

Algorithm 1 FeCAM

Training data $(D_1, D_2, .., D_T)$, Test data for evaluation $(X_1^e, X_2^e, .., X_T^e)$, Model ϕ
for task $t \in [1, 2, .., T]$ **do**
  if $t == 1$ **then**
    Train ϕ on $D_1 = (X_1, Y_1)$ # Train the feature extractor
  end if
  for $y \in Y_t$ **do**
    $\mu_y = \frac{1}{|X_y|} \sum_{x \in X_y} \phi(x)$ # Compute the prototypes
    $\phi(\tilde{X}_y) = Tukeys(\phi(X_y))$ # Tukeys transformation Eq. (9)
    $\mathbf{\Sigma}_y = Cov(\phi(\tilde{X}_y))$ # Compute the covariance matrices
    $(\mathbf{\Sigma}_y)_s = Shrinkage(\mathbf{\Sigma}_y)$ # Apply covariance shrinkage Eq. (8)
    $(\hat{\mathbf{\Sigma}}_y)_s = Normalization((\mathbf{\Sigma}_y)_s)$ # Apply normalization Eq. (7)
  end for
  for $x \in X_t^e$ **do**
    $y^* = \underset{y=1,..,Y_t}{\arg\min}\ \mathcal{D}_M(\phi(x), \mu_y)$ where
    $\mathcal{D}_M(\phi(x), \mu_y) = (\phi(\tilde{x}) - \tilde{\mu}_y)^T (\hat{\mathbf{\Sigma}}_y)_s^{-1} (\phi(\tilde{x}) - \tilde{\mu}_y)$
      # Compute the squared mahalanobis distance to prototypes
  end for
end for

2.3.3 On the Suboptimality of Learning Linear Classifier

Previous methods like [82, 194] in few-shot learning assumed Gaussian distributions of classes in feature space and proposed to transfer statistics of old classes to obtain calibrated distributions of new classes and then sample examples from the calibrated distribution to train a linear logistic regression classifier. In our setting, we consider a similar baseline for comparison which assumes Gaussian distributions for features of old classes (by storing the mean and the covariance matrix from the features of the old classes) and samples features from these distributions to learn a linear classifier. We state that this is not an

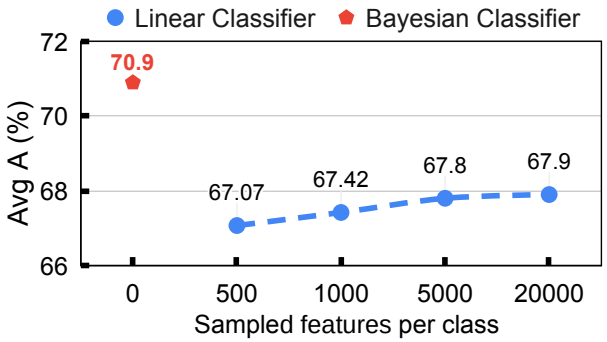


Figure 2.5: Avg Acc comparison of bayesian and linear classifier on CIFAR100 (T=5) setting.

ideal solution since the optimal decision boundaries need not be linear. The optimal decision boundaries are linear when the covariances of all classes are equal like in the Euclidean space. When the covariances of classes are not equal, the optimal decision boundaries are non-linear and forms a quadratic surface in high-dimensional feature space. We show in fig. 2.5 that using the optimal Bayesian classifier obtains much better performance compared to sampling features and training a linear classifier, even when sampling many examples per class.

2.4 Experiments

2.4.1 Experimental Setup

We evaluated FeCAM with strong baselines on multiple datasets and different scenarios.

MSCIL datasets and setup. We conduct experiments on three publicly available datasets: 1) CIFAR100 [81] - consisting of 100 classes, 32×32 pixel images with 500 and 100 images per class for training and testing, respectively; 2) TinyImageNet [86] - a subset of ImageNet with 200 classes, 64×64 pixel images, and 500 and 50 images per class for training and testing, respectively; 3) ImageNet-Subset [28] - a subset of the ImageNet LSVRC dataset [145] consisting of 100 classes with 1300 and 50 images per class for training and testing, respectively. We divide these datasets into incremental settings, where

the number of initial classes in the first task is larger and the remaining classes are evenly distributed among the incremental tasks. We experiment with three different incremental settings for CIFAR100 and ImageNet-Subset: 1) 50 initial classes and 5 incremental learning (IL) tasks of 10 classes; 2) 50 initial classes and 10 IL tasks of 5 classes; 3) 40 initial classes and 20 IL tasks of 3 classes. For TinyImageNet, we use 100 initial classes and distribute the remaining classes into three incremental settings: 1) 5 IL tasks of 20 classes; 2) 10 IL tasks of 10 classes; 3) 20 IL tasks of 5 classes. At test time, task IDs are not available.

FSCIL datasets and setup. We conduct experiments on three publicly available datasets: 1) CIFAR100 (described above); 2) miniImageNet [177] - consisting of 100 classes, 84×84 pixel images with 500 and 100 images per class for training and testing, respectively; 3) Caltech-UCSD Birds-200-2011 (CUB200) [178] - consisting of 200 classes, 224×224 pixel images with 5994 and 5794 images for training and testing, respectively. For CIFAR100 and miniImageNet, we divide the 100 classes into 60 base classes and 40 new classes. The new classes are formulated into 8-step 5-way 5-shot incremental tasks. For CUB200, we divide the 200 classes into 100 base classes and 100 new classes. The new classes are formulated into 10-step 10-way 5-shot incremental tasks.

Compared methods. We compare with several exemplar-free CIL methods in the many-shot setting [6,57,76,101,134,138,199,211–213] and in the few-shot setting [23,99,132,165,177,202,205,209]. Results of compared methods marked with $*$ are reproduced. For the upper bound of CIL, a joint training on all data is presented as a reference.

Evaluation. Similar to [134, 212, 213], we evaluate the methods in terms of average incremental accuracy. Average incremental accuracy A_{inc} is the average of the accuracy a_t of all incremental tasks (including the first task) and is a fair metric to compare the performances of different methods across multiple tasks.

$$A_{inc} = \frac{1}{T}\sum_{t=1}^{t=T} a_t \tag{2.11}$$

Implementation details. We use PyCIL [206] framework for our experiments. For both MSCIL and FSCIL settings, the main network architecture is ResNet-18 [53] trained on the first task using SGD with an initial learning rate of 0.1 and a weight decay of 0.0001 for 200 epochs. For the shrinkage, we use $\gamma_1 = 1$ and $\gamma_2 = 1$ for many-shot CIL and higher values $\gamma_1 = 100$ and $\gamma_2 = 100$ for few-shot CIL in our experiments. Following most methods, we store all the class prototypes. Similar to [211], we also store the covariance matrices for all classes

Table 2.1: Average top-1 incremental accuracy in exemplar-free many-shot CIL with different numbers of incremental tasks. Best results - **in bold**, second best - underlined.

CIL Method	CIFAR-100			TinyImageNet			ImageNet-Subset		
	T=5	T=10	T=20	T=5	T=10	T=20	T=5	T=10	T=20
EWC [76]	24.5	21.2	15.9	18.8	15.8	12.4	-	20.4	-
LwF-MC [138]	45.9	27.4	20.1	29.1	23.1	17.4	-	31.2	-
DeeSIL [6]	60.0	50.6	38.1	49.8	43.9	34.1	67.9	60.1	50.5
MUC [101]	49.4	30.2	21.3	32.6	26.6	21.9	-	35.1	-
SDC [199]	56.8	57.0	58.9	-	-	-	-	61.2	-
PASS [212]	63.5	61.8	58.1	49.6	47.3	42.1	64.4	61.8	51.3
IL2A [211]	66.0	60.3	57.9	47.3	44.7	40.0	-	-	-
SSRE [213]	65.9	65.0	61.7	50.4	48.9	48.2	-	67.7	-
FeTrIL* [134]	67.6	66.6	63.5	55.4	54.3	53.0	73.1	71.9	69.1
Eucl-NCM	64.8	64.6	61.5	54.1	53.8	53.6	72.2	72.0	68.4
FeCAM - $\mathbf{\Sigma}^{1:t}$	68.8	68.6	67.4	56.0	55.7	55.5	75.8	75.6	73.5
FeCAM - $\mathbf{\Sigma}_y$	**70.9**	**70.8**	**69.4**	**59.6**	**59.4**	**59.3**	**78.3**	**78.2**	**75.1**
Upper Bound	79.2	79.2	79.2	66.1	66.1	66.1	84.7	84.7	84.7

seen until the current task. In the experiments with visual transformers, we use ViT-B/16 [34] architecture pretrained on ImageNet-21k [161]. The extracted features are 512 dimensional when using Resnet-18 and 768 dimensional when using pretrained ViT.

2.4.2 Experimental Results

Many-shot CIL Results

The results for an exemplar-free MSCIL setup are presented in table 2.1. We present the results for a set of different MSCIL methods, joint training of a classifier, and a simple NCM classifier (Eucl-NCM) on a frozen backbone. FeCAM outperformed all others by a large margin in all settings. FeCAM version with $\Sigma^{1:t}$, storing a single covariance matrix representing all classes, already gave significantly better results than the current state-of-the-art method - FeTrIL. However, FeCAM with covariance matrix approximation Σ_y pushes the average incremental accuracy even higher and present excellent results. It is worth noticing that Eucl-NCM outperforms many existing CIL methods. Only FeTrIL performs better than Eucl-NCM on all datasets. In fig. 2.6, we present accuracy curves after each task for ten task scenarios. SSRE has a lower starting point due to a different network architecture. The rest of the methods, despite having the same starting point, end up with very different accuracies at the last task. Eucl-NCM still presents more competitive results than SSRE. FeTrIL presents better performance but is still far from FeCAM

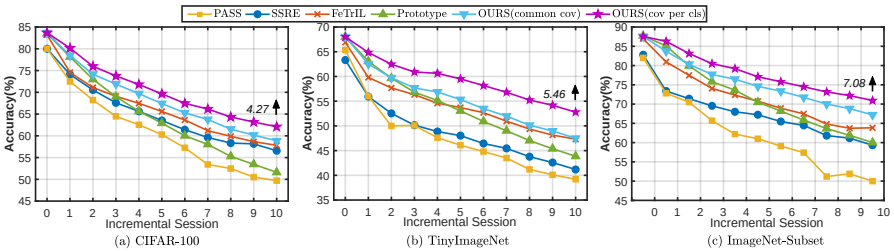


Figure 2.6: Accuracy of each incremental task for: (a) CIFAR100, (b) TinyImageNet, and (c) ImageNet-Subset and multiple MSCIL methods. We annotate the Avg $\mathcal{A}$cc. of all sessions between FeCAM and the runner-up method at the end of each curve. Here, prototype refers to NCM with euclidean distance.

Table 2.2: Comparison of our method with recent exemplar-based methods which store 2000 exemplars. For fair comparison, we show the number of parameters (in millions) by #P. Results on Imagenet-Subset excerpted from [207].

CIL Method			CIFAR-100 (T = 5)		ImageNet-Subset (T = 5)	
	#P	**Ex.**	Avg. $\mathcal{A}$cc	Last $\mathcal{A}$cc	Avg. $\mathcal{A}$cc	Last $\mathcal{A}$cc
iCaRL [138]	11.17	✓	65.4	56.3	62.6	53.7
PODNet [35]	11.17	✓	67.8	57.6	73.8	62.9
Coil [210]	11.17	✓	-	-	59.8	43.4
WA [203]	11.17	✓	69.9	61.5	65.8	56.6
BiC [190]	11.17	✓	66.1	55.3	66.4	49.9
FOSTER [180]	11.17	✓	67.9	60.2	69.9	63.1
DER [193]	67.02	✓	**73.2**	**66.2**	<u>77.6</u>	**71.1**
MEMO [208]	53.14	✓	-	-	76.7	70.2
FeCAM	11.17	✗	<u>70.9</u>	<u>62.1</u>	**78.3**	<u>70.9</u>

with a common covariance matrix. The FeCAM with a covariance matrix per class outperforms all other methods starting from the first incremental task. Here, in comparison to common covariance matrix, we pay the price in memory and need to store covariance matrix per class. Despite storing a matrix per class, we have less memory overhead compared to exemplar-based methods and do not violate privacy concerns by storing images.

Additionally, we compare our method against popular exemplar-based CIL methods in table 2.2, where the memory buffer is set to 2K exemplars. Our method outperforms all others that do not expand the model significantly (see #P column for the number of parameters after the last task). Only Dynamically Expandable Representation (DER), which grows the model almost six times, can outperform our method.

Table 2.3: Avg $\mathcal{A}$cc or Test $\mathcal{A}$cc at the end of the last task on class-incremental Split-Cifar100 [81], Split-ImageNet-R [54] and domain-incremental CoRe50 [103] benchmarks. All methods are initialized with pretrained weights from ViT-B/16 [34] for fair comparison.

	Split-Cifar100	Split-ImageNet-R	CoRe50
CIL Method	Avg $\mathcal{A}$cc	Avg $\mathcal{A}$cc	Test $\mathcal{A}$cc
FT-frozen	17.7	39.5	-
FT	33.6	28.9	-
EWC [76]	47.0	35.0	74.8
LwF [96]	60.7	38.5	75.5
L2P [187]	83.8	61.6	78.3
NCM [65]	83.7	55.7	85.4
FeCAM	**85.7**	**63.7**	**89.9**
Joint	90.9	79.1	-

Experiments with pre-trained models

In table 2.3 different settings for exemplar-free MSCIL are presented where we follow experimental settings of Learning-to-Prompt (L2P) method [187]. Here, all methods use a ViT encoder pre-trained on ImageNet-21K. L2P [187] is a strong baseline that does not train the encoder and learns an additional 46K prompt parameters. However, as Janson *et al.* [65] presented, a simple NCM classifier can perform better for some datasets in CIL, e.g., Split-ImageNet-R [54], Split-CIFAR100 [81] and for domain-incremental learning on CoRe50 [103].

We use the widely-used benchmark in continual learning, Split-CIFAR-100 which splits the original CIFAR-100 [81] into 10 tasks with 10 classes in each task unlike the other settings in Table 1, which have different task splits. Based on ImageNet-R [54], Split-ImageNet-R was recently proposed by [186] for continual learning which contains 200 classes randomly divided into 10 tasks of 20 classes each. It contains data with different styles like cartoon, graffiti and origami, as well as hard examples from ImageNet with a high intra-class diversity making it more challenging for CIL experiments. We use CoRe50 [103] for domain-incremental settings where the domain of the same class of objects is changing in new tasks. It consists of 50 different types of objects from 11 domains. The first 8 domains are used for learning and the other 3 domains are used for testing. Since it has a single test task, we report the test accuracy after learning on all 8 domains similar to [65, 187]. Results of compared methods excerpted from [65].

We use the proposed FeCAM method with the pre-trained ViT using a

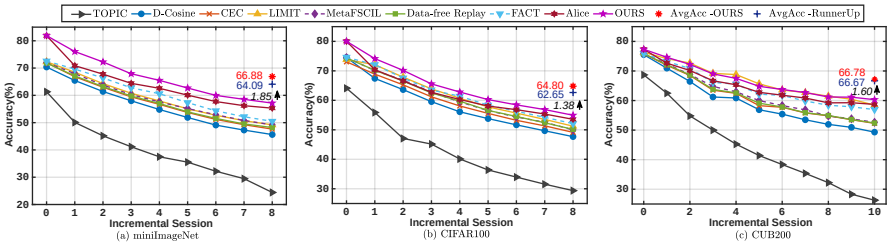


Figure 2.7: FSCIL methods accuracy of each incremental task for (a) miniImageNet, (b) CIFAR100 and (c) CUB200. We annotate the performance gap after the last session and Avg $\mathcal{A}$cc. of all sessions between FeCAM and the runner-up method at the end of each curve.

covariance matrix per class on CIL settings. In the domain-incremental setting, we maintain a single covariance matrix per class across domains and update the matrix in every new domain by averaging the matrix from the previous domain and from the current one. FeCAM outperformed both L2P and NCM, in all the settings. Notably, FeCAM outperforms L2P by 11.55% and NCM by 4.45% on the CoRe50 dataset.

Few-shot CIL Results

In many-shot settings, it is possible to obtain a very informative covariance matrix from the large number of available samples, while it can be difficult to get a good matrix in FSCIL from just 5 samples per class. To stabilize the covariance estimation, we use higher values of γ_1 and γ_2. In fig. 2.7 we present accuracy curves after each task for FSCIL on miniImageNet, CIFAR100, and CUB200 datasets. While TOPIC [165] does better than the finetuning methods, it does not perform well in comparison to the recent methods which has significantly improved the performance. ALICE [132] recently proposed to obtain compact and well-clustered features in the base task which helps in generalizing to new classes. We follow the experimental settings from ALICE [132]. We take the strong base model after the first task from ALICE and use the proposed FeCAM classifier (with a covariance matrix per class) instead of NCM classifier in the incremental tasks. We outperform ALICE significantly on all the FSCIL benchmarks.

FeCAM can easily be adapted to available few-shot methods in CIL since most methods obtain class prototypes from few-shot data of new classes and then use the euclidean distance for classification. We show in this chapter that

starting from the base task model from ALICE and simply using the FeCAM metric for classification significantly improves the performance across all tasks for the standard few-shot CIL benchmarks.

We report the average accuracy after each task for all methods on Cifar100 in table 2.4, on CUB200 in table 2.5 and on miniImageNet in table 2.6.

Table 2.4: Detailed accuracy of each incremental session on CIFAR100 dataset. Best among columns in **bold**.

Method	Accuracy in each session (%)									Avg $\mathcal{A}$
	0	1	2	3	4	5	6	7	8	
Finetune	64.10	39.61	15.37	9.80	6.67	3.80	3.70	3.14	2.65	16.54
D-Cosine [177]	74.55	67.43	63.63	59.55	56.11	53.80	51.68	49.67	47.68	58.23
CEC [202]	73.07	68.88	65.26	61.19	58.09	55.57	53.22	51.34	49.14	59.53
LIMIT [209]	73.81	72.09	67.87	63.89	60.70	57.77	55.67	53.52	51.23	61.84
MetaFSCIL [23]	74.50	70.10	66.84	62.77	59.48	56.52	54.36	52.56	49.97	60.79
Data-free Replay [99]	74.40	70.20	66.54	62.51	59.71	56.58	54.52	52.39	50.14	60.78
FACT [205]	74.60	72.09	67.56	63.52	61.38	58.36	56.28	54.24	52.10	62.24
ALICE [132]	**80.03**	70.38	66.6	62.72	60.28	58.06	56.83	55.35	53.56	62.65
ALICE+FeCAM	**80.03**	**74.15**	**70.16**	**65.57**	**62.82**	**60.25**	**58.46**	**56.86**	**54.94**	**64.80**

Table 2.5: Detailed accuracy of each incremental session on CUB200 dataset. Best among columns in **bold**.

Method	Accuracy in each session (%)											Avg $\mathcal{A}$
	0	1	2	3	4	5	6	7	8	9	10	
Finetune	68.68	43.70	25.05	17.72	18.08	16.95	15.10	10.06	8.93	8.93	8.47	21.97
D-Cosine [177]	75.52	70.95	66.46	61.20	60.86	56.88	55.40	53.49	51.94	50.93	49.31	59.36
CEC [202]	75.85	71.94	68.50	63.50	62.43	58.27	57.73	55.81	54.83	53.52	52.28	61.33
LIMIT [209]	76.32	74.18	**72.68**	**69.19**	**68.79**	**65.64**	63.57	62.69	**61.47**	60.44	58.45	66.67
MetaFSCIL [23]	75.90	72.41	68.78	64.78	62.96	59.99	58.30	56.85	54.78	53.82	52.64	61.93
Data-free Replay [99]	75.90	72.14	68.64	63.76	62.58	59.11	57.82	55.89	54.92	53.58	52.39	61.52
FACT [205]	**77.92**	74.94	71.57	66.32	65.96	62.49	61.23	59.76	57.94	57.56	56.41	64.70
FACT+FeCAM	**77.92**	**75.34**	72.23	67.56	67.02	63.50	62.39	61.25	59.84	59.10	57.89	65.80
ALICE [132]	77.34	72.64	70.17	66.68	65.34	62.78	61.81	60.84	59.22	59.26	58.70	64.98
ALICE+FeCAM	77.34	74.64	72.22	69.02	67.50	64.82	**63.74**	**62.70**	61.20	**61.14**	**60.30**	**66.78**

Table 2.6: Detailed accuracy of each incremental session on miniImageNet dataset. Best among columns in **bold**.

Method	Accuracy in each session (%)									Avg $\mathcal{A}$
	0	1	2	3	4	5	6	7	8	
Finetune	61.31	27.22	16.37	6.08	2.54	1.56	1.93	2.6	1.4	13.45
D-Cosine [177]	70.37	65.45	61.41	58.00	54.81	51.89	49.10	47.27	45.63	55.99
CEC [202]	72.00	66.83	62.97	59.43	56.70	53.73	51.19	49.24	47.63	57.75
LIMIT [209]	72.32	68.47	64.30	60.78	57.95	55.07	52.70	50.72	49.19	59.06
MetaFSCIL [23]	72.04	67.94	63.77	60.29	57.58	55.16	52.90	50.79	49.19	58.85
Data-free Replay [99]	71.84	67.12	63.21	59.77	57.01	53.95	51.55	49.52	48.21	58.02
FACT [205]	72.56	69.63	66.38	62.77	60.6	57.33	54.34	52.16	50.49	60.70
ALICE [132]	**81.87**	70.88	67.77	64.41	62.58	60.07	57.73	56.21	55.31	64.09
ALICE+FeCAM	**81.87**	**76.06**	**72.24**	**67.92**	**65.49**	**62.69**	**59.98**	**58.54**	**57.16**	**66.88**

Table 2.7: Ablation of the performance indicating the contribution from the different components of our proposed method FeCAM for MSCIL with five tasks on CIFAR-100 and ImageNet-Subset datasets. Note that here we use variance normalization (different from eq. (2.7)) when using diagonal matrix.

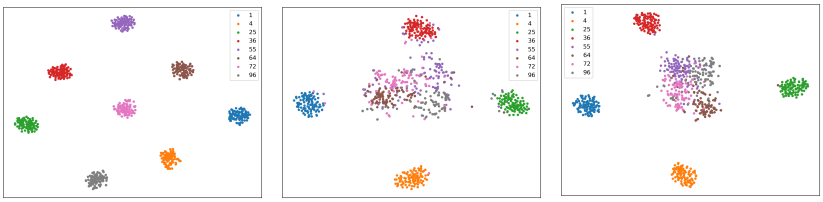
Distance	Cov. Matrix	Tukey eq. (2.9)	Shrinkage eq. (2.8)	Norm. eq. (2.7)	CIFAR-100 (T=5)		ImageNet-Subset (T=5)	
					Last Acc	Avg Acc	Last Acc	Avg Acc
Euclidean	-	✗	-	-	51.6	64.8	60.0	72.2
Euclidean	-	✓	-	-	54.4	66.6	66.2	73.6
Mahalanobis	Full	✗	✗	✗	14.6	29.7	33.5	45.1
Mahalanobis	Full	✓	✗	✗	20.6	36.2	54.0	65.6
Mahalanobis	Full	✗	✓	✗	44.6	59.3	39.9	56.9
Mahalanobis	Full	✓	✓	✗	52.1	62.8	56.5	67.3
Mahalanobis	Diagonal	✓	✓	✗	55.2	66.9	64.0	74.1
Mahalanobis	Full	✗	✓	✓	55.4	65.9	58.1	68.5
Mahalanobis	Full	✓	✓	✓	**62.1**	**70.9**	**70.9**	**78.3**

For further analysis to demonstrate the applicability of FeCAM, we take the base task model from FACT [205] and use FeCAM in the incremental tasks for the CUB200 dataset. FeCAM improves the performance on all tasks when applied to FACT as shown in table 2.5.

One of the main drawbacks of the many-shot continual learning methods is overfitting on few-shot data from new classes and hence these methods are not suited for few-shot settings. FeCAM is a single solution for both many-shot and few-shot settings and thus can be applied in both continual learning settings.

Ablation Studies

FeCAM propose to use multiple different components to counteract the CIL effect on the classifier performance. In table 2.7 the ablation study for MSCIL is presented, where contribution of each component is exposed in a meaning of average incremental and last task accuracy. Here, we consider the settings using a covariance matrix per class. We show that Tukeys transformation significantly reduces the skewness of the distributions and improves the accuracy using both Euclidean and Mahalanobis distances. The effect of the covariance shrinkage is more significant for CIFAR100 which has 500 images per class (less than 512 dimensions of feature space) while it also improves the performance on ImageNet-Subset. Here, we also show the usage of the diagonal matrix where we use only the diagonal (variance) values from the covariance matrices. We divide the diagonal matrices by the norm of the diagonal to normalize the variances. When using the diagonal matrix, the storage space is reduced from D^2 to D. Finally, we show that using the correlation normalization from eq. (2.7) gives the best accuracy by tackling the variance shift and better using the feature covariances.



(a) Joint-Training (b) CL-Frozen backbone (c) CL-Frozen backbone (Tukeys transformation)

Figure 2.8: The t-SNE plot for the features of new and old classes after Joint-Training (a) and after learning only the first 50 classes (b,c). In Joint-Training, the features are well clustered for all classes, however when the feature extractor is trained only on the first 50 classes, the new class representations are spread out. On applying Tukeys transformation, the new class embeddings are better clustered.

Time complexity. While previous methods train the classifier [134] or the model [212, 213] for several epochs in the new tasks, we do not perform any such training in new tasks. Among the existing methods, FeTrIL [134] claims to be the fastest. We compare the time taken for the incremental tasks on ImageNet-Subset (T=5) for FeTrIL and the proposed FeCAM method. Using one Nvidia RTX 6000 GPU, FeTrIL takes 44 minutes to complete all the new tasks while FeCAM takes only 6 minutes.

Feature transformations. We analyze the t-SNE plot for the feature distributions of old and new classes from CIFAR100 50-50 (2 tasks) setting in different scenarios in fig. 2.8. When the model is trained jointly on all classes, the features of all classes are well clustered and separated from each other. In CIL settings with frozen backbone when the model is trained on only the first 50 classes, the features are well-clustered for the old classes while the features for new classes are scattered and not well-separated. When we make the feature distributions more gaussian using Tukeys transformation, we observe that the new class features are comparatively better clustered.

CIL settings with a small first task. Usually one method sticks to one setting. Exemplar-free methods use 50% of data in the first task as equally splitting is a much more challenging setting which is usually tackled by storing exemplars or by expanding the network in new tasks.

When half of the total classes is present in the first task, the feature extractor

Table 2.8: Analysis of performance when the first task has 20 classes only and 20 new classes are added in incremental tasks, on CIFAR-100 and ImageNet-Subset datasets.

Method	CIFAR-100		ImageNet-Subset	
	Last Acc	Avg Acc	Last Acc	Avg Acc
Euclidean-NCM	30.6	50.0	35.0	54.5
FeTrIL	46.2	61.3	48.4	63.1
FeCAM	**48.1**	**62.3**	**52.3**	**66.4**

is better. When we start with fewer classes (20 classes in the first step) and add 20 new classes at every task, we can observe the same behavior in table 2.8. FeCAM still works and outperforms other methods. However, the average incremental accuracy is not very high in this challenging setting because the representation learned in the first task is not as good as in the big first task setting.

Storage requirements. We analyze the storage requirements of FeCAM and compare it with the exemplar-based CIL methods in table 2.9 for ImageNet-Subset (T=5) setting. Due to the symmetric nature of covariance matrices, we can store half (lower or upper triangular) of the covariance matrices and reduce the storage to half. While most of the exemplar-based methods preferred a constant storage requirement of 2000 exemplars, storage requirement for FeCAM gradually increases across steps and is still less by about 206 MBs after the last task.

Table 2.9: Analysis of storage requirements across tasks for FeCAM and the exemplar-based methods (storing 2000 exemplars) for the ImageNet-Subset (T=5) setting.

Method	Task 0	Task 1	Task 2	Task 3	Task 4	Task 5
Exemplar-based	312 MB	312 MB	312 MB	312 MB	312 MB	312 MB
FeCAM	53 MB	63 MB	75 MB	85 MB	96 MB	106 MB

Pre-training with dissimilar classes. Similar to [73], we perform experiments using the DeiT-S/16 vision transformer pretrained on the ImageNet data with different pre-training data splits and then evaluate the performance of NCM (with euclidean distance) and the proposed FeCAM method on Split-CIFAR100 (10 tasks with 10 classes in each task). In order to make sure that the pretrained classes are not similar to the classes of CIFAR100, [73] manually removed 389 classes from the 1000 classes in ImageNet. We take the publicly

Table 2.10: Performance of FeCAM and NCM-euclidean using Deit-S/16 pretrained transformer on Split-CIFAR100 dataset.

Method	DeiT pre-trained on 1k classes		DeiT pre-trained on 611 classes [73]	
	Last Acc	Avg Acc	Last Acc	Avg Acc
Euclidean-NCM	60.5	71.4	58.5	69.2
FeCAM	**70.2**	**78.5**	**68.6**	**76.9**

available DeiT-S/16 weights pre-trained on remaining 611 classes of ImageNet by [73] and evaluate NCM and FeCAM as shown in table 2.10. As expected, the performance of both methods drops a bit when the pre-training is not done on the similar classes. Still FeCAM outperforms NCM by about 10% on the final accuracy. Thus, this experiment further validates the effectiveness of modeling the covariance relations using our FeCAM method in settings where images from the initial task are dissimilar to new task images.

Hyperparamaters. We analyze the effect of the covariance shrinkage hyperparamaters γ_1 and γ_2 in fig. 2.9 for the many-shot setting (T=5) on Cifar100. Based on the observations, we see that the chosen parameters $\gamma_1 = 1$ and $\gamma_2 = 1$ obtain good results. Similarly, we use $\gamma_1 = 1$ and $\gamma_2 = 1$ for all many-shot experiments on CIFAR100, TinyImageNet and ImageNet-Subset. We use $\gamma_1 = 1$ and $\gamma_2 = 0$ for the experiments on Split-CIFAR100 and Core50 datasets. For Split-ImageNet-R, We use $\gamma_1 = 10$ and $\gamma_2 = 10$. For all the few-shot CIL settings, we obtain better results with $\gamma_1 = 100$ and $\gamma_2 = 100$.

Since the Resnet-18 feature extractor uses a ReLU activation function, the feature representation values are all non-negative, so the inputs to tukey's ladder of powers transformation are all valid. However, when using the ViT encoder pre-trained on ImageNet-21K, we also have negative values in the feature representations, hence we do not apply the tukey's transformation on the features for those experiments.

2.5 Conclusions

In this chapter, we revisit the anisotropic Mahalanobis distance for exemplar-free CIL and propose using it to effectively model covariance relations for classification based on prototypes. We analyze the heterogeneity in the feature distributions of the new classes that are not learned by the feature extractor. To address the feature distribution shift, we propose our Bayes classifier method FeCAM, which uses Mahalanobis distance formulation and additionally uses some

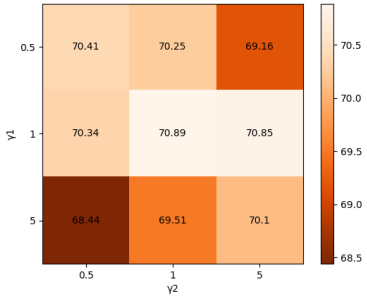


Figure 2.9: Impact of covariance shrinkage hyperparameters on many-shot CIFAR100 (T=5) setting using the proposed FeCAM method

techniques like correlation normalization, covariance shrinkage, and Tukey's transformation to estimate better covariance matrices for continual classifier learning. We validate FeCAM on both many- and few-shot incremental settings and outperforms the state-of-the-art methods by significant margins without the need for any training steps. Additionally, FeCAM evaluated in the class- and domain-incremental benchmarks with pretrained vision transformers yields state-of-the-art results. FeCAM does not store any exemplars and performs better than most exemplar-based methods on many-shot CIL settings. As a future work, FeCAM can be adapted to CIL settings where the feature representations are continually learned.

Limitations. The proposed approach needs a strong feature extractor or a large amount of data in the first task to learn good representations, as we do not learn new features but reuse the ones learned on the first task (or from pretrained network). Therefore, the method is not apt when training from scratch, starting with small tasks. We would then need to extend the theory to feature distributions which undergo feature drift during training; next to prototype drift [199] also covariance changes should be modeled.

3 Resurrecting Old Classes with New Data for Exemplar-Free Continual Learning*

3.1 Introduction

The *exemplar-free CIL* (EFCIL) setting has been extensively studied recently [45, 111, 134, 199, 212–214]. However, unlike exemplar-based methods, the EFCIL methods are only effective when starting with high-quality feature representations and are thus dependent on having a large initial task which is typically half of the whole dataset. However, a more practical CIL approach should be able to perform well on training from a smaller initial task and at the same time should not store exemplars. We define this as a *small-start setting* and analyze how existing EFCIL methods perform in this setting.

A critical aspect in CIL is the *semantic drift* of feature representations [199] after training on new tasks. This results in the movement of class distributions in feature space. Thus, it is crucial to track the old class representations after learning new tasks. While the class-mean in the new feature space can be effectively estimated using Nearest-Mean of Exemplars (NME) [35, 138], it is challenging to estimate it without exemplars. Usually, this drift is minimized with heavy functional regularization, which consequently restricts the plasticity of the network. Another way is to estimate it from the drift of current data, as done in SDC [199] or by augmenting old prototypes using new class features [111, 154]. In this chapter, we propose a novel drift estimation method using adversarial examples to resurrect old class prototypes in the new feature space as shown in fig. 3.1.

Adversarial examples [3, 61, 108, 119, 162] are maliciously crafted inputs that are designed to fool a neural network into predicting a different output than the one initially predicted for the original input. Exploiting the concept of targeted adversarial attacks [84, 108], we propose to perturb the new data such that the adversarial images result in embeddings close to the old prototypes. Now, the drift from old to new feature space is estimated using these adversarial samples,

*This chapter is based on a publication in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024 [47].

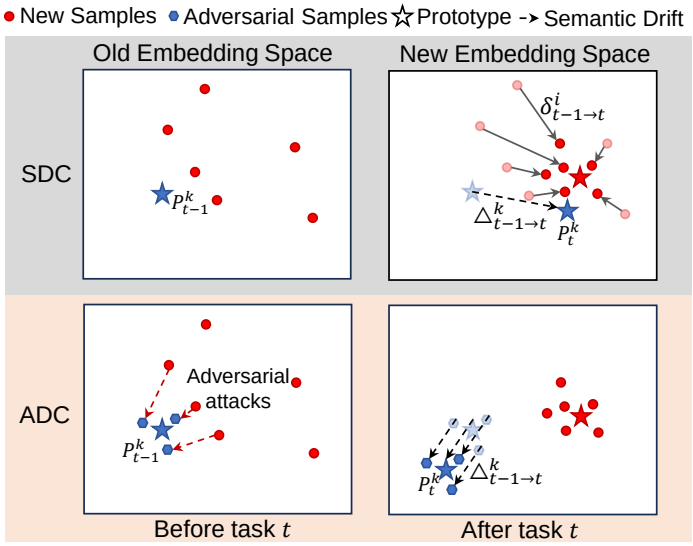


Figure 3.1: Illustration of Adversarial Drift Compensation (ADC) and SDC [199]. In SDC, the drift $\Delta^k_{t-1\to t}$ is estimated as the average of drift of all new task samples after training on a new task. Instead, we propose to move the new task features close to the old prototype P^k_{t-1} of class k by perturbing the new images using targeted adversarial attacks. The drift of the adversarial samples from old to new feature space is used to resurrect all old prototypes.

which serve as pseudo-exemplars for the old classes. We hypothesize that the pseudo-exemplars behave like the original exemplars in the feature space, and thus we exploit them to measure the drift. This generation of adversarial samples is computationally cheaper and much faster (only a few iterations) compared to data-inversion methods [197] which inverts embeddings to realistic images.

Following recent studies [45, 65, 107], we explore using class prototypes with an NCM [138] classifier and show that a simple baseline of logits distillation [96] with an NCM classifier often outperforms existing EFCIL methods in the small-start setting. Applying our proposed drift compensation method with this baseline, we obtain state-of-the-art performance with significant gains over existing methods on standard CL benchmarks using CIFAR-100 [81], TinyImageNet [86] and ImageNet-Subset [28] as well as fine-grained datasets like

CUB-200 [178] and Stanford Cars [80]. Our contributions can be summarized as:

- We study the challenging EFCIL settings and highlight the importance of continually learning from small-start settings instead of assuming the availability of half of the dataset in the first task.
- We present a novel and intuitive method - Adversarial Drift Compensation (ADC) to estimate semantic drift and resurrect old class prototypes in the new feature space. We also investigate how adversarially generated samples transfer in CIL settings from old to the new model.
- We perform experiments on several CIL benchmarks and outperform state-of-the-art methods by a large margin on several benchmark datasets. Especially notable are our results on fine-grained datasets, where we report performance gains of around 9% for last task accuracy.

3.2 Related Work

Class-Incremental Learning. CIL [25, 113, 207] methods aim to learn new data which arrives incrementally and suffers from the catastrophic forgetting problem [114, 142]. During evaluation in CIL without the task id, it is difficult to distinguish classes that belong to different tasks [160]. While in general this setting is tackled using *rehearsal approaches* [7, 31, 35, 57, 143] by storing raw inputs, some attempts have been made without storing raw inputs. LwF [96] prevents important changes in the network by preventing the output of the current model to drift too much from the output of the previous model. PASS [212] learns the backbone using self-supervised learning and later uses functional regularization and feature rehearsal, SSRE [213] proposed an architecture organization strategy that aims to transfer invariant knowledge across tasks. In FeTRIL [134], the authors freeze the feature extractor and estimate the position of the old class features by using the current task data variance. Recently, FeCAM (chapter 2) leveraged the mean and covariance of the previous task features and proposed a mahalanobis distance-based classifier.

Drift estimation. When updating the feature extractor on new classes, the representation learned for the old class prototypes changes and thus the need to rectify those drifts [199]. SDC [199] showed that the new data can be used to estimate the drift of the old prototype representations. Recent methods [111, 154, 168] also explored how to update the prototypes learned in old tasks to counter the drift. Toldo et al. proposed to learn the relations

between old and new class features to estimate the drift. NAPA-VQ [111] proposed to augment the prototypes using the topological information of classes in the feature space. Prototype Reminiscence [154] proposed to dynamically reshape old class feature distributions by interpolating the old prototypes with the new sample features. In this work, we generate adversarial samples which behaves as pseudo-exemplars and is then used to measure the drift.

CL using Adversarial Attacks. Adversarial Attacks has been studied in-depth in recent years [3, 61, 162], and has been later harnessed to create realistic looking images from a trained vision model [120], including inputs that can be later used for training [197]. Some recent methods [37, 67, 83, 155] in exemplar-based CIL borrowed the idea of adversarial attacks. ASER [155] used the kNN-specific Shapley value to obtain more representative buffer samples. GMED [67] edits the exemplars by monitoring the change in loss when training on incoming data. RAR [83] used the pairwise relations between the exemplars and the new samples and perturb the exemplars to obtain samples close to the decision boundaries. While all these approaches use adversarial attacks on the memory samples, we use it to perturb the new data to simulate the old data.

3.3 Method

We consider the EFCIL setup where new classes emerge over time and we are not allowed to store samples from old classes. These classes come in different tasks, one task at a time, and the tasks contain a mutually exclusive set of classes. When training on task t, we have access to current dataset $D_t = \{X_t, Y_t\}$ with images X_t and labels Y_t. The main goal of EFCIL is to learn a model h that correctly classifies the data into classes encountered so far. We use $h_t(x) = \sigma(W_t f_t(x))$, where f_t is the feature extractor parameterized by θ_t learned in task t and W_t is weight matrix of the linear classifier with softmax function σ.

3.3.1 Motivation

In general, for a new feature extractor f_t trained on new data and an old feature extractor f_{t-1} from previous task, we have access to old class prototypes up to task $t-1$ denoted by $P_{t-1}^{Y_{1:t-1}}$. We compute the prototypes for all new classes after training in the current task. For a class k in task t, we compute $P_t^k = \frac{1}{|X_t^k|} \sum_{x \in X_t^k} f_t(x)$, where X_t^k is the set of samples from class k, $f_t(x)$ is the feature embedding for an image x from class k. However, the old class prototypes were computed on the old feature space $f_{t'}$ $(t' < t)$ in old tasks and

have drifted to a different position in the new feature space f_t after training on new data.

Previously, SDC [199] proposed to compensate the drift of these prototypes $P_{t-1}^{Y_{1:t-1}}$ by computing the drift from old model embeddings $f_{t-1}(x)$ to new model embeddings $f_t(x)$ corresponding to all images x in the current task. This drift of the current data is then used to approximate the drift of previous task prototypes by considering a weighted Gaussian window around the prototypes (giving more weights to drift vectors close to the prototype). However, the quality of this drift approximation for previous prototypes is expected to be low when few current task data points are close to a given prototype in the embedding space. We show in our experiments that SDC indeed struggles to estimate the drift in the small-start settings where the feature representations change considerably.

In a similar fashion, it is possible to estimate this drift without using such weights, by simply choosing the closest samples to each old class prototype and compute the average feature of these samples when fed to the new backbone. We select samples from the current task that are close to the old prototype of a given class and verify that such samples also lie close to the oracle prototype (in the new feature space). We analyze in Fig. fig. 3.2 that there exist a correlation between the distance to the old prototype in the old feature space and the distance to the oracle prototype in the new feature space (see blue dots). This motivates us to leverage current task samples so that their distance to the old prototype in the old feature space is even smaller, which could in turn improve the drift estimation. We hypothesize that this can be done by computing adversarial samples from the current task samples, aiming for their representation to match one of the old class prototypes in the old feature space.

3.3.2 Adversarial Drift Estimation

To estimate the drift of old class prototypes after updating the model on new classes, it is desirable to have the exemplars. These exemplars can be passed through the new model to compute the *oracle* prototype position in the new feature space. However, in the exemplar-free setting, we can only access the new data. In order to use the new data to represent the old data, we exploit the concept of targeted adversarial attacks [84, 108] to target one old class at a time and perturb the new data in a way that it serves as a substitute of old data to the model. We perform adversarial attacks on new data to move its embeddings very close to old prototypes in the old feature space. Using the adversarial samples, we can estimate the drift from old to new feature space and compensate it as illustrated in fig. 3.3.

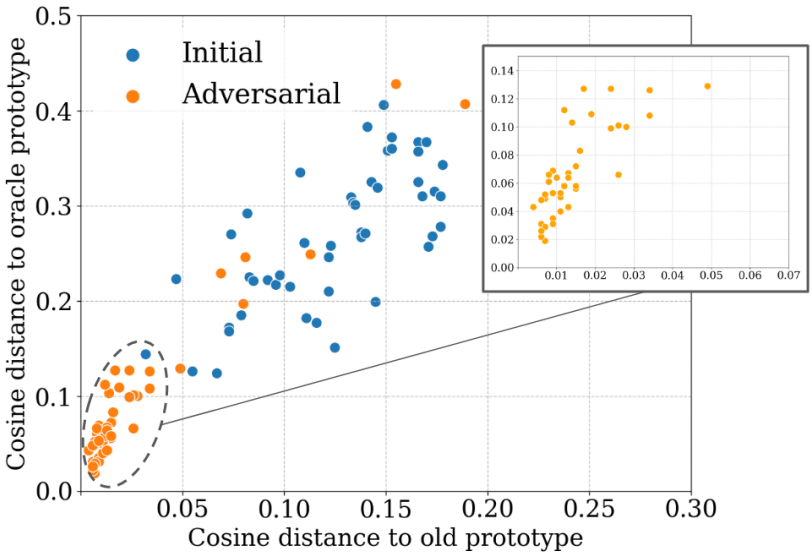


Figure 3.2: Illustration to show that the cosine distance between embeddings and old prototype in the old feature space is correlated with the cosine distance between embeddings and oracle prototype in the new feature space. This holds true for embeddings of both initial and adversarial samples. For demonstration, we select few current-task samples that are closest to the old prototype and choose the same target old class for all samples. The blue and orange points represents the non-modified current class samples and the modified samples using our proposed approach respectively. In this analysis, we compute the oracle prototype using all old task data in the new feature space.

To estimate the drift of prototype P^k_{t-1} for a target old class k, we obtain $\mathcal{X}^k$ by sampling a set of m data points from the current task data X_t which are closest to P^k_{t-1} based on L2 distance between the embeddings of samples in X_t and the prototype P^k_{t-1}. We aim to perturb the samples $x \in \mathcal{X}^k$ and obtain $\mathcal{X}^k_{adv}$ such that the adversarial samples $x_{adv} \in \mathcal{X}^k_{adv}$ are closer to P^k_{t-1} and are now classified to class k using the NCM classifier in the old feature space:

$$k = \underset{y \in Y_{1:t-1}}{\arg\min} \|f_{t-1}(x_{adv}) - P^y_{t-1}\|_2. \tag{3.1}$$

We propose the following optimization objective by computing the mean

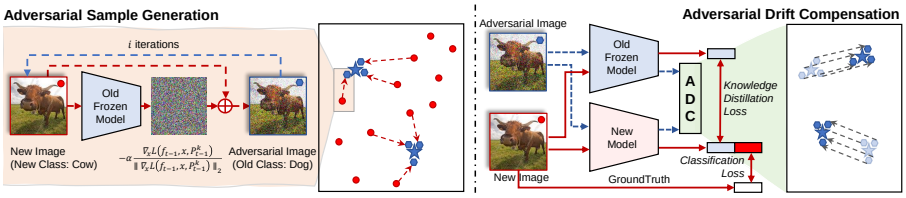


Figure 3.3: (a) Adversarial Sample Generation: On the old model feature space, the new samples closest to the old prototype are selected and iteratively perturbed in the direction of the target old prototype to generate adversarial samples which are now misclassified as the target old class resulting in embeddings closer to the old prototype. We perform this for every old class (we show 2 classes here for demonstration). (b) Model Training with Drift Compensation: The new model is trained using the classification loss for learning new classes and knowledge distillation loss to prevent forgetting of old classes. After the new model is trained, the adversarial samples generated using the old model are passed through both the models and the drift from old to new feature space is estimated. This is then used to update the old prototypes.

squared error between the features $f_{t-1}(x)$ and the prototype P^k_{t-1} as:

$$L(f_{t-1}, \mathcal{X}^k, P^k_{t-1}) = \frac{1}{|\mathcal{X}^k|} \sum_{x \in \mathcal{X}^k} \|f_{t-1}(x) - P^k_{t-1}\|^2_2. \tag{3.2}$$

In order to move the feature embedding in the direction of the target prototype P^k_{t-1}, we obtain the gradient of the loss with respect to the data $x \in \mathcal{X}^k$, normalize it to get the unit attack vector and scale it by α as follows:

$$x_{adv} \leftarrow x - \alpha \frac{\nabla_x L(f_{t-1}, x, P^k_{t-1})}{\|\nabla_x L(f_{t-1}, x, P^k_{t-1})\|_2} \; \forall x \in \mathcal{X}^k \tag{3.3}$$

where $\nabla_x L(f_{t-1}, x, P^k_{t-1})$ is the gradient of the objective function with respect to the data x and α refers to the step size. We perform the attack for i iterations.

Here, the goal is different from conventional adversarial attacks like FGSM and its variants [33,44,84,108] which aim to minimize the perturbation in order to keep the perturbed image visually similar to the real image by having a fixed ϵ-budget generally based on ℓ_2 or ℓ_∞-norm of perturbation. In our case, we do not need to apply such restrictions on the distance between initial and final image, instead, we only clip the perturbed image in the existing range

of pixel values. We do observe that our formulation is closer to the ℓ_2-norm based attack as we use ℓ_2 normalization of the gradient vector to obtain a unit perturbation vector which is scaled using the step size.

Continual Adversarial Transferability. An interesting aspect of our method is that the adversarial samples x_{adv} are crafted on the old feature extractor f_{t-1} and then passed to the new feature extractor f_t expecting that the adversarial samples will still be misclassified as the target old class k. We define this as *continual adversarial transferability* where the adversarial samples generated on the old feature space still behave in the same way on the new feature space. This is feasible since the old model and the new model are not entirely different because of the knowledge distillation used in order to reduce catastrophic forgetting. This is related with the concept of adversarial transferability [62,108,118,130], where an attack obtained on one neural network also behaves as an attack on other independently trained neural network based architectures.

We analyze the oracle setting using the old class data to validate the continual adversarial transferability. We show in Fig. fig. 3.2 that distance of the adversarial samples from their target prototypes in the old feature space is still correlated with their distance to the oracle prototypes of the target class in the new feature space. This suggests that the adversarial samples crafted using the old feature space are still effective in the new feature space and therefore allows us to reliably compute the drift from these adversarial samples.

3.3.3 Drift Compensation

The adversarial samples when passed through the new feature extractor f_t are expected to lie close to the drifted prototype and hence are used to compute the drift. After generating the adversarial samples for each target class k, we measure the prototype drift as:

$$\Delta^k_{t-1\rightarrow t} = \frac{1}{|\mathcal{X}^k_{adv}|} \sum_{x_{adv}\in\mathcal{X}^k_{adv}} (f_t(x_{adv}) - f_{t-1}(x_{adv})) \tag{3.4}$$

where $x_{adv} \in \mathcal{X}^k_{adv}$ is the set of only those adversarial samples which are classified as the target class k using the NCM classifier. We resurrect the old prototypes by compensating the drift as follows:

$$P^k_t = P^k_{t-1} + \Delta^k_{t-1\rightarrow t} \tag{3.5}$$

After compensating all old prototypes, we use the NCM classifier in the new feature space for classifying the test samples. Unlike SDC [199], we do not perform weighted averaging based on the distances to the prototype since embeddings from adversarial images are very close to the prototypes and we found no gain by applying this additional weighting scheme.

In algorithm 2, we present the pseudo-code for adversarial drift compensation.

Algorithm 2 Adversarial Drift Compensation

Input: Images X_t, feature extractors f_t and f_{t-1}, old prototypes $P_{t-1}^{Y_{1:t-1}}$, step size α, number of generated samples m, number of iterations for perturbation i.
Output: Resurrected prototypes $P_t^{Y_{1:t-1}}$.
for $k \in Y_{1:t-1}$ **do**
    # Sample a set $\mathcal{X}^k$ of m new samples from X_t which are closest to P_{t-1}^k based on L2 distance.
    for i iterations **do**
        $L(f_{t-1}, \mathcal{X}^k, P_{t-1}^k) =$
            $\frac{1}{|\mathcal{X}^k|} \sum_{x \in \mathcal{X}^k} \|f_{t-1}(x) - P_{t-1}^k\|_2^2.$
        $x_{adv} = x - \alpha \frac{\nabla_x L(f_{t-1}, x, P_{t-1}^k)}{\|\nabla_x L(f_{t-1}, x, P_{t-1}^k)\|_2} \forall x \in \mathcal{X}^k$
        $x \leftarrow x_{adv}$
    end for
    add x in $\mathcal{X}_{adv}^k$ if $\arg\min_{y \in Y_{1:t-1}} \|f_{t-1}(x) - P_{t-1}^y\|_2 = k$. # Store only those samples in $\mathcal{X}_{adv}^k$ which are successfully misclassified as k
    $\Delta_{t-1 \to t}^k = \frac{1}{|\mathcal{X}_{adv}^k|} \sum_{x_{adv} \in \mathcal{X}_{adv}^k} f_t(x_{adv}) - f_{t-1}(x_{adv})$
    $P_t^k = P_{t-1}^k + \Delta_{t-1 \to t}^k$ # Prototype resurrection
end for

3.3.4 Training Strategy

In addition to learning new classes, we perform knowledge distillation [96] on the logits to transfer knowledge from the frozen teacher model at previous task $t-1$ to the student model at current task t as follows:

$$\mathcal{L} = \mathcal{L}_{ce}(h_t(x), y) + \lambda \mathcal{L}_{ce}(h_{t-1}(x), h_t(x)) \tag{3.6}$$

Method	CIFAR-100				TinyImageNet				ImageNet-Subset			
	T = 5		T=10		T = 5		T =10		T = 5		T = 10	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [96]	45.35	61.94	26.14	46.14	38.81	49.70	<u>27.42</u>	38.77	50.88	69.11	37.90	61.60
NCM	53.53	<u>66.35</u>	41.31	57.85	38.69	50.45	26.56	<u>41.04</u>	57.74	71.99	<u>45.86</u>	65.04
SDC [199]	<u>54.94</u>	64.82	<u>41.36</u>	<u>58.02</u>	<u>40.05</u>	<u>50.82</u>	27.15	40.46	<u>59.82</u>	<u>74.10</u>	43.72	<u>65.83</u>
PASS [212]	49.75	63.39	37.78	52.18	36.44	48.64	26.58	38.65	50.96	66.15	38.90	54.74
SSRE [213]	42.39	56.57	29.44	44.38	30.13	43.20	22.48	34.93	40.30	57.57	28.12	45.87
FeTrIL [134]	45.11	60.42	36.69	52.11	29.91	43.99	23.88	36.35	49.18	63.83	40.26	55.12
FeCAM [45]	47.28	61.37	33.82	48.58	25.62	39.85	23.21	35.32	54.18	67.21	42.68	57.45
ADC	**59.14**	**69.62**	**46.48**	**61.35**	**41.0**	**50.94**	**32.32**	**43.04**	**62.40**	**74.84**	**47.58**	**67.07**

Table 3.1: Evaluation of EFCIL methods on small-start settings. Best results in **bold** and second best results are <u>underlined</u>.

where λ refers to the regularization strength, the first term $\mathcal{L}_{ce}$ refers to the cross-entropy loss for learning new classes and the second term performs the regularization by forcing the probabilities of old classes on the old model h_{t-1} and new model h_t to be similar and thus prevents forgetting.

3.4 Experiments

Datasets. We perform experiments on several CIL benchmarks. CIFAR-100 [81] contains 50k training images of size 32x32 and 10k test images, divided in 100 classes. TinyImageNet [86] contains 100k training images and 10k test images from 200 classes and image size of 64x64, taken as a subset of ImageNet [28]. ImageNet-Subset is a subset of the ImageNet (ILSVRC 2012) dataset [145] containing 100 classes with a total of 130k training images and 5k test images and image size of 224x224. We equally split all these datasets in 5 and 10 tasks. This is different from the big-start settings with half of the dataset in first task, commonly used in EFCIL benchmarks [45, 134, 212]. We also use two fine-grained datasets for our experiments. CUB-200 [178] contains 200 classes of birds with 224x224 image size, 5994 images for training and 5794 images for testing. We use the 5-split and 10-split settings for CUB-200. Stanford Cars [80] consists of 196 car models with 224x224 images, 8144 for training and 8041 for testing and we split it into 7 and 14 tasks. As implemented in PyCIL [206], for CIFAR-100, we use the same augmentation policy which consists of small random transformations like contrast or brightness changes. Similarly, for the other datasets, we use the default set of augmentations which include random crop and random horizontal flip. For a fair evaluation, we use the same set of augmentations for all the methods.

Method	CUB-200				Stanford Cars			
	T = 5		T=10		T = 7		T =14	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [96]	<u>58.68</u>	<u>71.31</u>	41.96	60.15	<u>45.18</u>	61.14	30.33	49.93
NCM	52.74	67.13	38.47	57.83	42.22	59.06	31.60	51.34
SDC [199]	55.20	68.64	41.63	60.43	45.03	<u>61.75</u>	32.15	<u>53.18</u>
PASS [212]	34.04	49.00	26.37	41.08	20.71	37.13	12.30	25.46
FeTrIL [134]	54.66	67.45	49.09	62.42	36.92	54.09	34.29	50.41
FeCAM [45]	53.47	66.39	<u>51.78</u>	<u>64.97</u>	40.64	56.24	<u>37.50</u>	52.78
ADC	**64.46**	**73.49**	**57.97**	**68.91**	**54.86**	**67.07**	**45.07**	**61.39**

Table 3.2: Evaluation of EFCIL methods on fine-grained datasets. Best results in **bold** and second best results are <u>underlined</u>.

Training Details. We use the PyCIL framework [206] as a basis for all our experiments. The training is performed using the ResNet18 model [53] and the SGD optimizer. For CIFAR-100, in the first task, we use a starting learning rate of 0.1, momentum of 0.9, batch size of 128 and weight decay of 5e-4 for 200 epochs, the learning rate is reduced by a factor of 10 after 60, 120, and 160 epochs. In the subsequent tasks, we use an initial learning rate of 0.05 reduced by a factor of 10 after 45 and 90 epochs and train for 100 epochs. Following [113], we set the regularization strength to 10 and the temperature to 2. The network is trained from scratch on CIFAR-100, TinyImageNet and ImageNet-Subset. For the experiments on fine-grained datasets, we use the ImageNet pretrained weights following standard practice [147, 199]. For ADC, we use a α value of 25, iterations $i = 3$ and number of closest samples $m = 100$ for all the datasets. Similar to most existing methods, we store all the class prototypes.

Compared Methods. Since none of the EFCIL methods are designed to start from a small first task, we implement those methods in our small-start settings. This includes LwF [96], PASS [212], SSRE [213], FeTRIL [134] and FeCAM (chapter 2). Naturally, we also include a comparison to the existing drift-estimation method SDC [199] and the baseline model with NCM classifier. For SDC and NCM results reported in table 3.1 and table 3.2, we train the models using LwF and perform NCM classification in the feature space. For FeTrIL and FeCAM, the feature extractor is frozen after the first task, while for the other methods, it is continually learned. Note that here we adapt SDC with distillation on the logits, which is different from [199] where they performed distillation on the features.

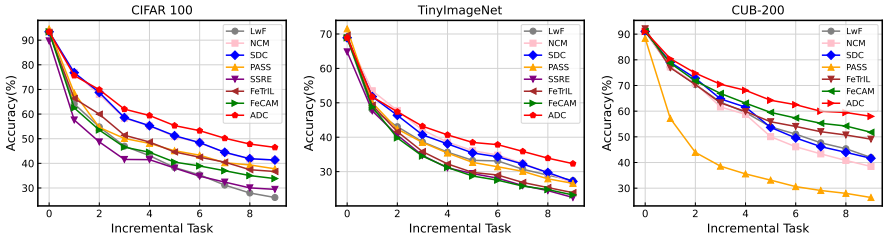


Figure 3.4: Accuracy after each incremental task for CIFAR-100, TinyImageNet and CUB-200 datasets on 10 task settings. ADC improves over the compared methods starting from the initial to the last task.

Evaluation. We report the average accuracy after the last task denoted by A_{last} and the average incremental accuracy which is the average of the accuracy after all tasks (including the first one) denoted by A_{inc}. A_{inc} better reflects the performance of the methods across all the tasks.

3.4.1 Quantitative Evaluation

We observe that methods proposed for the big-start settings of EFCIL are not effective in small-start settings and perform poorly. A simple baseline trained with LwF and using NCM classifier is performing better than most of the existing approaches - SSRE, PASS, FeTrIL and FeCAM in several settings. While SDC improves over NCM, the proposed method ADC outperforms all existing methods in both last task accuracy and average incremental accuracy across all settings in table 3.1 and table 3.2. ADC outperforms the second-best method SDC by 4.2% on 5-task and by 5.12% on 10-task settings of CIFAR-100 on last-task accuracy. For TinyImageNet, ADC improves over the second-best method by 0.95% on 5-task and by 5.17% on 10-task settings. On ImageNet-Subset, ADC is better by 2.58% on 5-task and by 1.72% on 10-task settings after the last task.

We also evaluate the EFCIL methods on the challenging fine-grained datasets of CUB-200 and Stanford Cars. We observe in table 3.2 that LwF is a strong baseline here, particularly in the 5-task and 7-task settings and methods like NCM and SDC are not much better than LwF. While PASS performs poorly on both datasets, FeTrIL and FeCAM performs better with FeCAM outperforming the other methods on the 10-task setting of CUB-200 and 14-task setting of Stanford Cars. ADC outperforms the runner-up methods by 5.78% on 5-task setting and by 6.19% on 10-task settings of CUB-200. On Stanford Cars dataset,

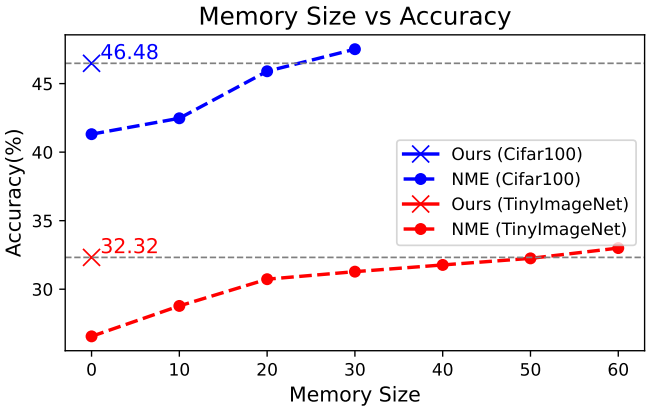


Figure 3.5: Memory Size vs accuracy comparison of NME and ADC on CIFAR-100 and TinyImageNet (T=10) settings.

ADC is better by 9.68% on 7-task setting and 7.57 % on 14-task setting. We analyze how the accuracy after each task varies for all the methods in fig. 3.4 and observe that ADC consistently outperforms the other methods across all tasks.

Comparison to NME. We compare the last-task accuracy of ADC with exemplar-based NME where the exemplars are used to estimate the old class prototype positions in the new feature space. We show in fig. 3.5 that ADC outperforms NME using 20 exemplars per class for CIFAR-100 (total memory size of 2000 samples) and using 50 exemplars per class (total memory size of 10k samples) for TinyImageNet.

3.4.2 Computational overhead of ADC

Using ADC requires some additional computation to be made in-between each training session. In this section, we provide an estimation of the additional computation required by our method and compare it to the training time of a single task. At the end of each task, our method requires estimating the drift of each stored prototype (1 per old class) and for each of these, compute several adversarial samples starting from available current task samples. As a consequence, the training time of our method scales linearly with the number of classes. For each class, we compute 100 adversarial samples in a single batch and perform 3 training iterations. In order to perform one iteration, we need

(a) Impact of iterations ($\alpha = 25$)

Iterations	1	2	3	5	10
A_{inc}	60.94	61.23	**61.35**	61.25	60.93
A_{last}	45.96	46.45	**46.48**	45.95	45.28

(b) Impact of α (iterations=3)

α	1	10	25	50	100
A_{inc}	60.46	61.14	**61.35**	60.83	60.93
A_{last}	44.89	46.43	**46.48**	45.19	45.28

(c) Impact of the number of samples

Samples	25	50	100	300	500	1000
A_{inc}	60.19	60.89	61.35	61.54	**61.64**	61.47
A_{last}	43.98	45.63	46.48	46.66	**46.83**	46.32

Table 3.3: Impact of hyperparameters on CIFAR100 (T=10) setting using the proposed ADC method.

to compute the gradient of the adversarial loss with respect to the input image, whose cost is equivalent to the one of a normal training backward pass [151]. So, if we denote the number of classes by N_c, and the number of iterations by N_i, we need to perform $N_c \times N_i$ backward passes. In the case of CIFAR-100 and ImageNet-Subset divided in 10 tasks each containing 10 classes, this means an overhead of, $\sum_{t=1}^{9} 10 \times t \times 3 = 1350$ backward passes. In contrast, one new task is trained for 100 epochs with a batch size of 128 (39 batches per epochs with 10 tasks on CIFAR-100), which amounts to 3900 backward passes per task, and two times more for the first task (trained for 200 epochs). In total, our method increases the computational cost by 3.1% on this setting. For the 5-task setting of CIFAR-100 and ImageNet-Subset, it increases by 2.5%.

3.4.3 Ablation Studies

In table 3.3, we conduct an analysis on the impact of various hyperparameters, including the number of iterations, α, and the number of closest samples used for ADC, on CIFAR-100 (T=10) setting. Based on the observations in table 3.3a, we find that choosing even a very low number of iterations, specifically 3, yields favorable results when generating the perturbed images. Additionally, using $\alpha = 25$ achieves good accuracy for both incremental and

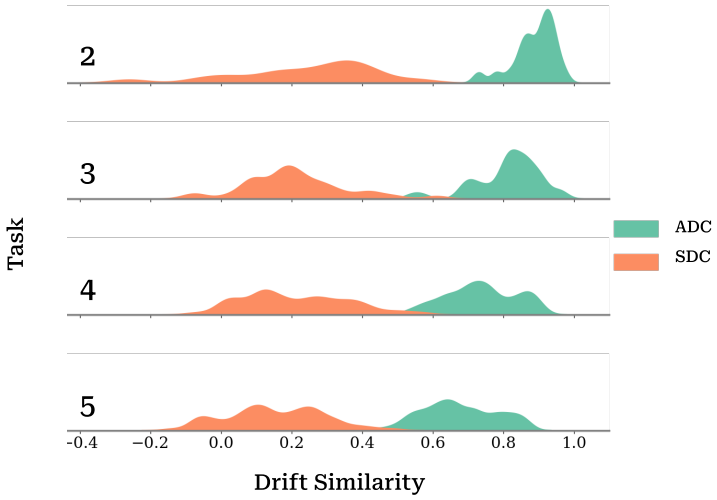


Figure 3.6: Comparison between the two drift estimation methods SDC [199], and the proposed ADC, on CIFAR-100 (5 tasks). We compute the drift for each class with the two methods and report the distribution of drift estimation quality, measured by computing the cosine similarity between the estimated drift vector and the true drift (obtained using old data), for all previous class prototypes.

final task evaluations in table 3.3b. Regarding the number of closest samples to the target old prototype, table 3.3c shows that the accuracy improves marginally on considering more than 100 samples. So, we take the 100 closest samples for our experiments which is computationally cheaper and yet achieves very good accuracy. Interestingly, even taking only the closest 25 samples achieves 2.62% better accuracy than the runner-up method SDC.

Drift estimation quality. We validate through table 3.1 and table 3.2 that the designed ADC method is giving better accuracy results than the previous SDC method for all datasets. As an additional verification, we check that this method was indeed better than SDC at estimating the old prototypes drift. To do so, we use both SDC and ADC on the same trained checkpoints on CIFAR-100 5-task settings and compare the estimated drift to the true drift computed using old data. We report the results in fig. 3.6, where we show the distribution of the estimated drift qualities. One drift per class is estimated and we compute the cosine similarity of estimated drift to the true drift. We see that for all training tasks, the drifts estimated with ADC are

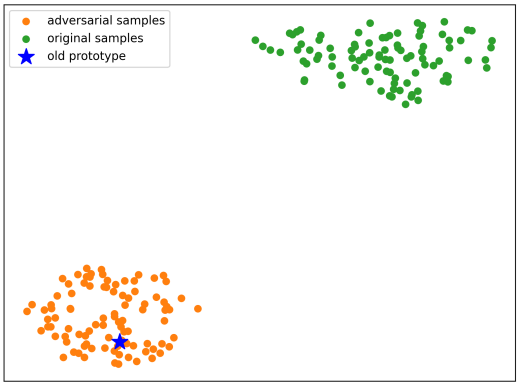


Figure 3.7: The t-SNE plot demonstrates that the adversarial samples generated using our proposed method lie close to the target old class prototype compared to the closest current task samples (in green) and can thus be reliably used for drift estimation.

of better quality than the ones estimated with SDC. We observe that some class drift estimations with SDC have negative cosine similarity with the true drift. However, we also see that the estimation quality decreases slightly for later training tasks. Indeed, as the backbone drifts more and more, it gets harder to estimate the actual drift. The fact that we see this decrease more prominently for ADC might be because the similarities obtained by SDC are already centered around a low-value (0.15) after the second task, whereas the better ADC drift estimation is centered first around 0.9, to then decrease and reach a minimum average of 0.7. This validates that ADC can track the movement of prototypes in feature space.

Robustness to different class orders. In CIL, the order of classes can influence the performance and thus we shuffle the class orders and observe how ADC and the existing methods like LwF, NCM, SDC, FeTrIL and FeCAM perform. While we used the seed 1993 following previous works [113, 134, 138, 199] for the results before, here we use four different seeds 0, 1, 2, 3 and report the mean and standard deviation using these 5 seeds for both the last task accuracy A_{last} and the average incremental accuracy A_{inc} in tables 3.4 to 3.6. The proposed method ADC outperforms SDC and NCM consistently across all settings on CIFAR-100, TinyImageNet and ImageNet-Subset. This demonstrates the robustness of ADC which improves over the existing methods irrespective of the class order.

Method	T = 5		T=10	
	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [96]	45.67 ± 1.37	62.12 ± 1.08	26.59 ± 2.44	45.60 ± 2.52
NCM	$\underline{52.68 \pm 0.57}$	$\underline{65.91 \pm 0.4}$	40.53 ± 2.74	$\underline{56.21 \pm 3.55}$
SDC [199]	52.26 ± 3.0	63.88 ± 1.23	$\underline{40.65 \pm 1.82}$	56.18 ± 3.0
FeTrIL [134]	45.52 ± 0.33	60.84 ± 0.46	37.0 ± 0.57	52.08 ± 0.51
FeCAM [45]	46.93 ± 0.34	61.49 ± 0.55	33.13 ± 0.93	48.10 ± 1.27
ADC	$\mathbf{58.12 \pm 1.42}$	$\mathbf{69.29 \pm 1.17}$	$\mathbf{45.43 \pm 3.03}$	$\mathbf{59.59 \pm 4.11}$

Table 3.4: Evaluation of EFCIL methods with mean and standard deviation using 5 random seeds for CIFAR-100. Best results in **bold** and second best results are underlined.

Method	T = 5		T=10	
	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [96]	39.03 ± 0.43	50.96 ± 0.93	27.75 ± 0.51	40.04 ± 0.96
NCM	38.76 ± 0.18	51.74 ± 0.8	28.07 ± 0.97	$\underline{42.86 \pm 0.98}$
SDC [199]	$\underline{40.28 \pm 0.37}$	$\underline{52.21 \pm 0.89}$	$\underline{28.15 \pm 0.67}$	42.09 ± 1.03
FeTrIL [134]	29.94 ± 0.83	45.08 ± 0.98	23.6 ± 0.42	37.41 ± 0.63
FeCAM [45]	26.03 ± 0.49	41.11 ± 0.93	23.78 ± 0.48	37.30 ± 1.23
ADC	$\mathbf{41.29 \pm 0.47}$	$\mathbf{52.36 \pm 0.95}$	$\mathbf{32.68 \pm 0.43}$	$\mathbf{44.89 \pm 1.03}$

Table 3.5: Evaluation of EFCIL methods with mean and standard deviation using 5 random seeds for TinyImageNet. Best results in **bold** and second best results are underlined.

Method	T = 5		T=10	
	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [96]	49.16 ± 1.34	68.88 ± 0.74	34.18 ± 3.69	57.96 ± 3.64
NCM	56.80 ± 1.90	72.45 ± 0.87	$\underline{44.04 \pm 1.93}$	64.43 ± 0.43
SDC [199]	$\underline{59.62 \pm 1.56}$	$\underline{74.44 \pm 0.76}$	42.68 ± 1.88	$\underline{65.26 \pm 0.59}$
FeTrIL [134]	50.52 ± 1.06	65.64 ± 0.95	40.74 ± 0.50	56.34 ± 0.76
FeCAM [45]	53.83 ± 0.46	67.88 ± 0.67	42.46 ± 0.89	57.93 ± 1.45
ADC	$\mathbf{61.62 \pm 0.93}$	$\mathbf{75.29 \pm 0.61}$	$\mathbf{47.62 \pm 1.55}$	$\mathbf{67.03 \pm 0.46}$

Table 3.6: Evaluation of EFCIL methods with mean and standard deviation using 5 random seeds for ImageNet-Subset. Best results in **bold** and second best results are underlined.

Perturbation guarantee. We specifically select the closest samples to each old prototype, one at a time (see algorithm 2) to ensure we generate adversarial samples for all the old classes. On CIFAR100 (T=10), we get an average

of 59 samples out of 100, which are successfully perturbed for all old classes after the last task. While performing 5 iterations (instead of 3) generates an average of 69 successful perturbations for old classes, this does not lead to a significant accuracy change (Tab. 3a). Therefore, we have used 3 iterations in our implementation.

We analyze the position of the closest current task samples and the generated adversarial samples with respect to a target old class prototype in the old feature space using a t-SNE plot in fig. 3.7. We observe that the adversarial samples lie close to the prototype, while the original samples are distant from the prototype. This validates the effectiveness of the adversarial attack in the old feature space and shows how new samples obtained using targeted adversarial attacks can be used to represent old classes. These adversarial samples behave as pseudo-exemplars and can now be used to estimate the drift of prototypes from the old to the new feature space.

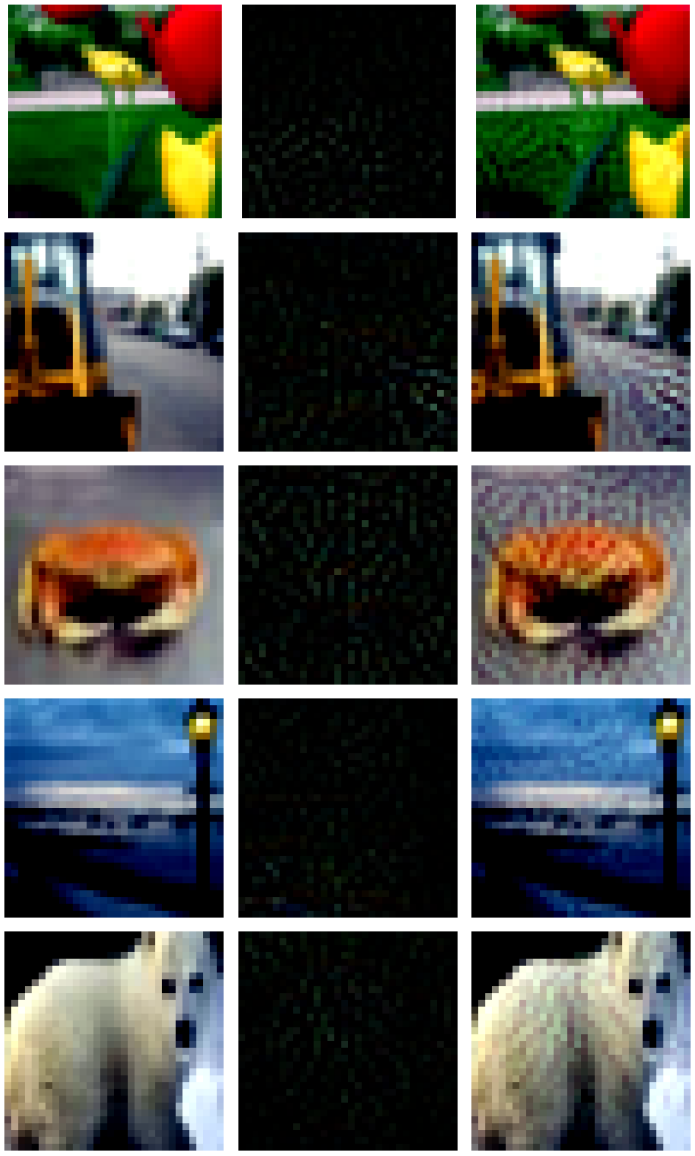
Visualization of adversarial images We visualize the original and adversarially perturbed images and the corresponding perturbations for CIFAR-100 and TinyImageNet in fig. 3.8 and fig. 3.9. We observe that the perturbations are perceptible in most of the adversarial images generated from low-resolution images of CIFAR-100 and TinyImageNet while for ImageNet-Subset, CUB-200 and Stanford Cars having high-resolution images of 224x224, the perturbations are not perceptible.

3.5 Conclusions

In this study, we explored a drift compensation method for exemplar-free continual learning. Drawing inspiration from adversarial attack techniques, we introduced a novel approach called Adversarial Drift Compensation. This method involves generating samples from the new task data in a manner that adversarial images result in embeddings close to the old prototypes. This approach allows us to more accurately estimate the drift of old prototypes in class-incremental learning without the need for any exemplars. Furthermore, we conducted an analysis of continual adversarial transferability, revealing an intriguing observation: generated samples for the old feature space (previous task) continue to behave similarly in the new feature space (current task). This sheds light on why the Adversarial Drift Compensation method performs exceptionally well. Through a series of experiments, we demonstrated that ADC effectively tracks the drift of class distributions in the embedding space, surpassing existing exemplar-free class-incremental learning methods on several standard benchmarks. Importantly, these improvements are achieved with-

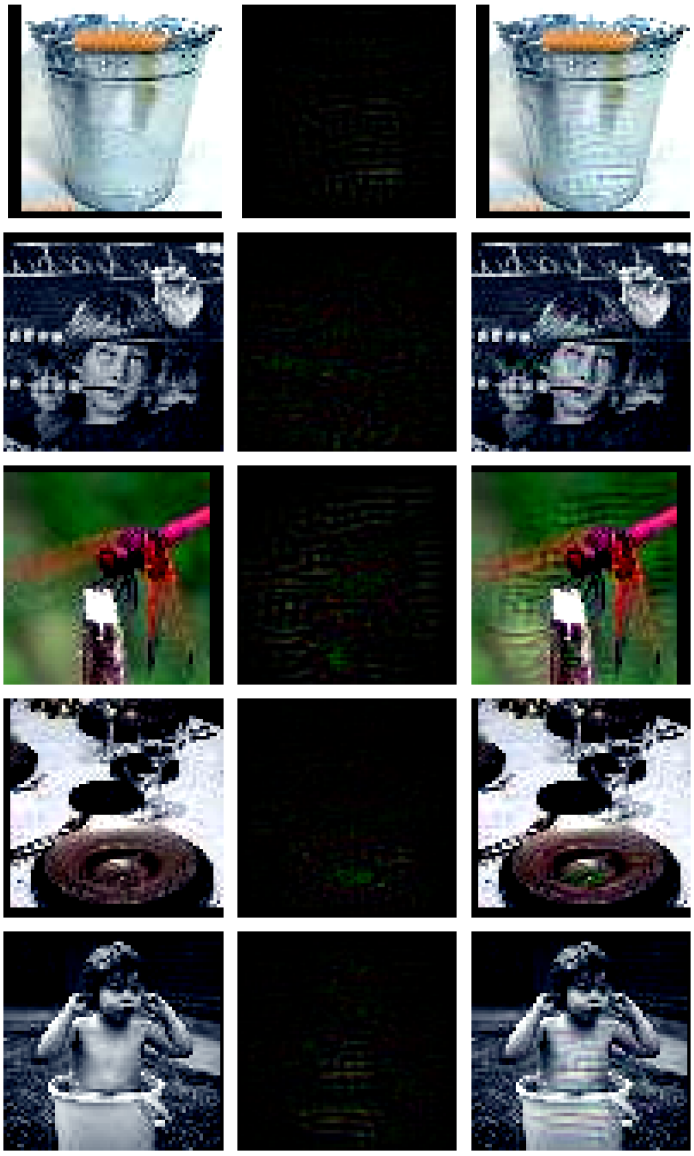
out imposing extensive computational overhead or requiring a large memory footprint.

Limitations. The ADC method, as currently designed, requires the access to the task boundaries during training in order to trigger the computation of the old prototypes drift and to access a big enough quantity of current data. The method would for instance be more challenging to use and would require changes in order to be applied in the online continual learning setting, or the continual few-shot learning setting where only a small amount of current data is available. Future work can explore these directions.



(a) Original image (b) Perturbation (c) Adversarial image

Figure 3.8: Visualization of image, perturbation and the corresponding adversarial image for some samples from CIFAR-100.



(a) Original image (b) Perturbation (c) Adversarial image

Figure 3.9: Visualization of image, perturbation and the corresponding adversarial image for some samples from TinyImageNet.

4 Query Drift Compensation: Enabling Compatibility in Continual Learning of Retrieval Embedding Models*

4.1 Introduction

Information Retrieval (IR) is widely used in several NLP applications like semantic search and Retrieval-Augmented Generation (RAG) for LLMs. Text embeddings which encode a sentence or a chunk of text to low-dimensional embedding vectors are commonly used for these applications [64, 90, 136]. Semantic search enables us to retrieve most relevant responses from a document corpus for a given query. While non-semantic lexical approaches like TF-IDF and BM25 [141] were traditionally used for retrieval, dense retrievers like transformer architectures [175] are now widely used for semantic search [20, 121, 126, 166, 196]. While lexical methods consider queries and documents as bag-of-words, dense retriever models encode the queries and corpus documents in a shared semantic embedding space [42] which enables us to precompute and index the document embeddings from the corpus before performing retrieval. The standard practice [121, 126, 166] is to consider a static setting in which the retriever model is trained on several datasets and the corpus documents are indexed using the trained retriever model. These indexed corpus document embeddings are then used for evaluation of retrieval for each task or dataset separately. However, in many practical applications not all datasets are jointly available, and new data arrives over time. In order to improve the retrieval models on newly incoming datasets, one needs to continually train them. In this work, we study how the retriever models can be fine-tuned on new datasets over time and how updating the model can impact the retrieval process and performance.

Continual Learning (CL) enables neural networks to learn a sequence of tasks one after another and perform well on all seen tasks. Several research works [15, 25, 113, 176, 185, 207] explore various aspects of CL, primarily for image classification tasks. A major challenge in CL is catastrophic forgetting [71, 114]

*This chapter is based on a publication in the Conference on Lifelong Learning Agents, 2025 [49].

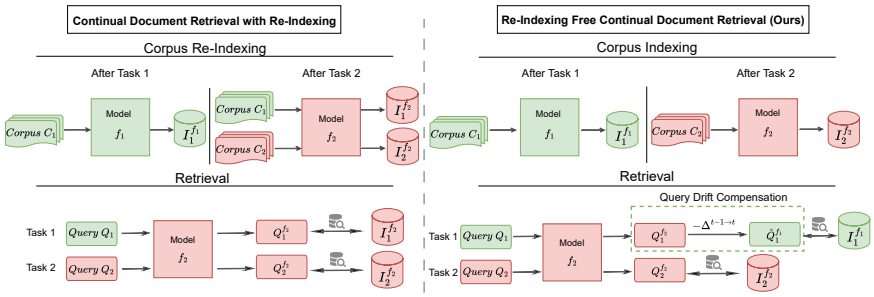


Figure 4.1: Continual Document Retrieval (CDR). Here, we consider a two-task continual setting and illustrate two different approaches to tackle the *embedding drift* issue for old task retrieval in CDR which arises due to non-compatibility between query embeddings from the updated model $Q_1^{f_2}$ (in red) and corpus embeddings indexed using the old model $I_1^{f_1}$ (in green). (left) A naive approach to make the query and corpus embeddings compatible is by re-indexing the corpus documents using the updated model. However, re-indexing large amounts of documents from all old tasks after updating the model on every new task is time-consuming and computationally expensive. (right) To avoid re-indexing, we propose to estimate embedding drift from old to new model $\Delta^{t-1 \to t}$ and compensate the drift from $Q_1^{f_2}$ during retrieval. The proposed query drift compensation approach enables compatibility by projecting the query embedding back to the embedding space of f_1 without any need for expensive re-indexing.

which refers to forgetting old tasks after learning new tasks. We focus on the task-incremental learning setting [172], where the task-id information is available during inference or retrieval. In this work, we discuss how continually updating the retriever models for document retrieval could lead to issues of non-compatibility between query and corpus document embeddings for old tasks. This is due to the fact that during retrieval of queries from old tasks, the query embeddings are obtained using the updated model while the corpus embeddings were previously indexed using the old model from the respective tasks. The issue of non-compatibility has previously been studied in CL for image retrieval works [12, 137, 179]. The drop in performance of old tasks when naively fine-tuning on new tasks can be attributed to the *embedding drift* (also referred to as semantic drift [199]) between old and new embedding spaces, as we demonstrate in our experiments.

One naive approach could be re-indexing of corpus documents from all old tasks (also known as back-filling in image retrieval [153]) to ensure that the

query and the corpus document embeddings are in the same embedding space as shown in fig. 4.1 (left). However, re-indexing of all old corpus documents after each task would involve very high computation costs. In order to avoid the re-indexing of old documents, we propose a novel query drift compensation approach which projects the query embeddings from the new model back to the embedding space of their respective tasks as illustrated in fig. 4.1 (right). We estimate query drift vectors for each task transition using the query samples from training data of the current task which we use to approximate the query drift of previous tasks. We compensate or subtract these drift vectors from new model query embeddings to move them back to the old space. Thus, instead of re-indexing old documents, we propose a simple query projection which makes the query and document embeddings compatible and thus reduces the forgetting of old tasks significantly.

We propose a CL benchmark for document retrieval tasks where we train on new datasets in each task using query-document pairs (see table 4.1) and aim to improve the document retrieval performance on all old and new tasks. We fine-tune the retriever model on several datasets from the BEIR benchmark [166] to improve retrieval on specific data or domains. We also propose to employ embedding distillation between new and the previous task model to reduce the *embedding drift* and the forgetting of old tasks. Our contributions can be summarized as:

- We establish a benchmark for Continual Document Retrieval (CDR) to evaluate the effect of continual training on large-scale datasets and observe forgetting of old tasks after fine-tuning on new tasks. We show that knowledge distillation using both query and corpus embeddings during fine-tuning can reduce the forgetting.

- We discuss how non-compatibility due to the *embedding drift* hurts retrieval performance. We propose a novel approach - Query Drift Compensation (QDC) which estimates the embedding drift between tasks after training on a new task. During retrieval, we propose to compensate the drift from queries (extracted from the updated model) to enable compatibility with the document embeddings (indexed using the old model).

- We demonstrate in our experiments how the proposed QDC enables continual learning of retrieval models on new datasets without any re-indexing of old documents and performs similar to joint training. We also show that our approach outperforms re-indexing based CDR.

4.2 Related Works

Continual Learning. CL methods [25, 113, 185, 207] focus primarily on reducing catastrophic forgetting of old classes after learning new classes. CL methods can be classified into class-incremental, task-incremental and domain-incremental settings [172]. In this work, we focus on the task-incremental setting where we have access to task-id during inference, since it is a more realistic setting for retrieval. Existing CL approaches can be broadly divided into replay-based, regularization-based, parameter isolation-based and prototype-based methods. Replay-based methods [7, 138] store exemplars from old tasks and use them during training on new tasks. However, storing exemplars has several limitations due to data regulations and privacy issues. Regularization-based methods prevent updates of important weights for old classes [76] or employ knowledge distillation approaches [35, 96] between previous and current model to preserve knowledge from previous tasks. Some methods [112, 150] divide the model to learn task-specific parameters. Prototype-based methods [45, 199] store a prototype representation for all seen classes and classify samples based on distances to the class prototypes. Semantic drift compensation [43, 47, 199] has been used to improve prototype-based methods by updating old prototypes. We use the concept of drift compensation for backward projection of queries from latest model to old model embedding space, thus enabling query-corpus compatibility.

Document Retrieval. Following success of large language models, the use of neural retrieval models [70, 72, 97, 105, 121] has become more common than traditional lexical approaches [141], which have the lexical gap [8]. Dense retrieval models [42] map queries and documents in a shared dense embedding space, and scores their relevance based on cosine similarity between query and document embeddings. Recent works [51, 126] use modified BERT [29] and perform MLM pretraining followed by unsupersived contrastive finetuning on large corpus of data and finally finetune on labelled datasets. In this work, we use the nomic embedding model from [126] which outperforms several competitive models [51, 63, 94, 184] in retrieval tasks.

Continual Document Retrieval. In CDR, the retriever model is expected to continually learn from new tasks over time without forgetting the previous tasks. While several works studied continual image retrieval [13, 179] and continual multimodal retrieval [183], fewer studies have explored continual learning in information retrieval. [104] investigated the catastrophic forgetting problem in neural ranking models for information retrieval. [40] built a continual information retrieval setting using a single dataset MS MARCO [4] and

Table 4.1: Examples of query-document pairs of different datasets from the BEIR [166] benchmark.

Dataset	Query	Document
MS MARCO [4]	how much magnesium in kidney beans	Kidney Beans. A cup of kidney beans contains 70 mg of magnesium and is a great source of protein and fiber. More: How to Bake With Beans.
NQ [85]	what is the meaning of a crown	Crowns are the main symbols of royal authority.[21]
HotpotQA [195]	What compound, known as aqua fortis or spirit of niter is used in rocket propellant?	Nitric acid (HNO), also known as aqua fortis and spirit of niter, is a highly corrosive mineral acid.
FEVER [167]	what team won the az state peach bowl	The 1970 Peach Bowl was a college football bowl game between the Arizona State Sun Devils and the North Carolina Tar Heels .
FiQA-2018 [110]	How can a 'saver' maintain or increase wealth in low interest rate economy?	Personally, I invest in mutual funds. Quite a bit in index funds, some in capital growth & international.

observed that catastrophic forgetting exists in IR to a lesser extent than image classification tasks. [17] proposed a re-indexing free memory-based method for first-stage retrieval in a different setting with unlabeled new task documents. Recently, [56] investigated how existing CL methods work in CDR by dividing MS MARCO into multiple tasks based on topics. Another set of works [77, 116] explored differentiable search index for CDR on how to encode the corpus of documents in the model parameters where the model output for a given query is the predicted document. In this work, we propose a more comprehensive benchmark for CDR using five large-scale datasets from [166] and analyze how continual training affect the retrieval performance.

4.3 Continual Learning for Document Retrieval

Here, we introduce and formalize the setting of Continual Document Retrieval (CDR). Following [166], we refer to a text of any length from the corpus as a 'document'. Document retrieval aims to return the most relevant document d from the given corpus C as a response to the query q provided by the user. In the continual setting, we refer to the set of queries and corpus of documents for task $t \in [1, T]$ as Q_t and C_t. Here, we denote a single query and document from task t as q_t and d_t respectively, where $d_t \in C_t$. In the first task, we train the embedding model f_1 on query-document pairs $\{Q_1, D_1\}$ from the training set of task 1. Note that $D_1 \in C_1$ refers to the set of documents from the corpus of task 1 that are used for training (typically the entire corpus is larger). We use the trained model f_1 for indexing all the documents from C_1 and refer to the indexed document embeddings as $I_1^{f_1}$. Similarly, we also use $q_j^{f_i} = f_i(q_j)$. Here, the superscript term denotes the model used for indexing and the subscript term denotes the task to which the document belongs. Similarly, for task t, we train embedding model f_t on $\{Q_t, D_t\}$ pairs from the training

set of task t and the corpus documents indexed by f_t are referred to as $I_t^{f_t}$. In our setting, we consider that during the training of task t we have no access to any data from previous tasks (also known as exemplar-free continual learning [45, 109, 134, 159]).

Non-compatibility due to Embedding Drift. After task t, we have access to the updated model f_t only, and thus we need to use f_t to embed the queries from all tasks during retrieval phase. For an old task $t' < t$, the queries $Q_{t'}$ from task t' are embedded using f_t denoted as $Q_{t'}^{f_t}$ but all the corpus documents $C_{t'}$ from t' were already indexed after task t' using $f_{t'}$ and stored as $I_{t'}^{f_{t'}}$. We refer to this phenomenon as *embedding drift*. This leads to non-compatibility between query and document embeddings, since they are in two different embedding spaces (the embedding space has been updated during the continual training). We illustrate the setting of continual document retrieval and the non-compatibility issue in fig. 4.1.

Re-indexing of old task corpus. A naive approach to enable compatibility of query and document embeddings is to re-index all the corpus documents from all old tasks to obtain $I_{t'}^{f_t} \; \forall \; t' < t$ after training on a new task t. Re-indexing removes the embedding drift between the queries $Q_{t'}^{f_t}$ and corpus documents $I_{t'}^{f_t}$, which are then in the same embedding space thus resolving the non-compatibility issue (see fig. 4.1).

However, re-indexing involves high computation costs and it potentially takes a considerable amount of time to re-index large amounts of documents from all old tasks after finetuning on every new task. Therefore, in this chapter, we focus on re-indexing free CDR. Re-indexing (or back-filling) based approach has been found to be effective and considered an upper bound in fine-tuning of image retrieval models [153]. Interestingly, from our analysis in CDR, we show that re-indexing based CDR performs poorly. We attribute this to misalignment due to the unequal drift of query and corpus embeddings (similar to observations in multi-modal continual retrieval [183]). We revisit and analyze this in the experiments section.

4.4 Re-indexing Free Continual Document Retrieval

4.4.1 Training Strategy

We follow the training strategy from [126] which performs masked language modeling (MLM) pre-training to train a long-context BERT model followed by multi-stage contrastive learning [94]. The first stage is unsupervised contrastive pre-training using the InfoNCE contrastive loss [128] on huge amounts of publicly available text-pairs which are mined from the web. Following the pre-training stages, the final step is performing supervised contrastive finetuning on each dataset at each task. Here, we consider the model pre-trained using MLM and unsupervised contrastive learning and then continually finetune the pre-trained model with supervised contrastive learning on query-document (q, d) training pairs of each task.

For the continual training at task t, we initialize a new model f_t with the weights of f_{t-1} and then perform contrastive training on (q, d) pairs from the training set of task t. For each (q, d) pair, we consider the most similar documents as hard negatives and use $\mathcal{H}$ hard negatives for each query which are mined from the same dataset. We minimize the contrastive loss function for a given batch of (q, d) pairs as follows:

$$\mathcal{L}_C = -\frac{1}{n}\sum_i \log \frac{e^{S(f_t(q_i),f_t(d_i))/\tau}}{\sum_{j=1}^{n} e^{S(f_t(q_i),f_t(d_j))/\tau} + \sum_{h=1}^{\mathcal{H}} e^{S(f_t(q_i),f_t(d_h))/\tau}}, \quad (4.1)$$

where $S(q, d)$ is the cosine similarity, τ is the temperature parameter and n is the batch size. Here, the first term in the denominator $\sum_{j=1}^{n} e^{S(f_t(q_i),f_t(d_j))/\tau}$ includes the in-batch negatives when $j \neq i$ and second term $\sum_{h=1}^{\mathcal{H}} e^{S(f_t(q_i),f_t(d_h))/\tau}$ includes the hard-negatives for the corresponding query.

4.4.2 Embedding Knowledge Distillation

A naive application of eq. (4.1) for all the tasks would lead to catastrophic forgetting and an embedding which is especially tailored to the last task. Therefore, here we adapt a commonly used regularization technique in CL for CDR to prevent forgetting [96]. More particularly, to reduce the feature drift and improve stability, we propose to perform feature distillation [199] by minimizing the cosine distance between the embeddings from old and new models. We perform the distillation on embeddings of both queries and

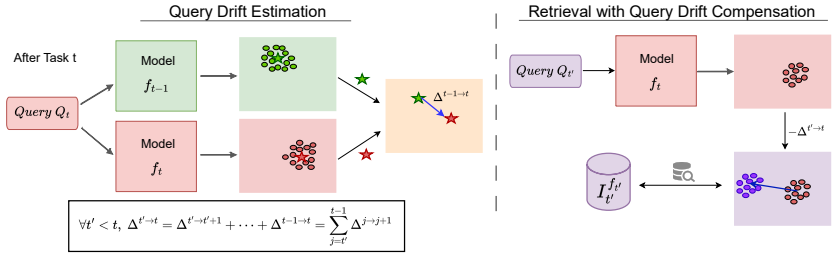


Figure 4.2: Illustration of Query Drift Compensation. (left) We show how to estimate the query drift vectors $\Delta^{t-1\to t}$ for each task transition (from $t-1$ to t) after training on task t. We store the drift vectors of all old tasks. For an old task t' ($t' < t$), we obtain the drift vector $\Delta^{t'\to t}$ by addition of all drift vectors from task t' to t. (right) During retrieval of old task t', we compensate the query embeddings with drift vector $\Delta^{t'\to t}$ to project them from embedding space of task t to that of t'. As a result, we compare the query and previously indexed document embeddings in the same embedding space of task t' (in purple), thus avoiding the non-compatibility issue.

documents from training data of the new task. We minimize the following distillation loss for a given batch of (q, d) pairs:

$$\mathcal{L}_D = \frac{1}{n}\sum_{i=1}^{n}(D_c(f_t(q_i), f_{t-1}(q_i)) + D_c(f_t(d_i), f_{t-1}(d_i))), \tag{4.2}$$

where D_c is the cosine distance between embeddings and n is the batch size. Even though the distillation improves the stability of the network, it does not completely solve the non-compatability issue discussed before, since the learning of new tasks still requires the embedding space to adapt and incorporate new knowledge.

4.4.3 Query Drift Compensation

Here we propose Query Drift Compensation (QDC) which addresses the non-compatibility issue in re-indexing free CDR. In this case, for $t' < t$ we would like to query the corpus $I_{t'}^{f_{t'}}$ with the query embedding from the same embedding model $Q_{t'}^{f_{t'}}$, however we have access to the query embedding using the latest embedding model $Q_{t'}^{f_t}$, which leads to the non-compatibilty issue. To enable backward compatibility of old task query embeddings $Q_{t'}^{f_t}$, $\forall\, t' < t$ (indexed

using continually updated new model f_t) with the corpus embeddings indexed using the old model $f_{t'}$, we propose to use drift compensation inspired by SDC [199] which was proposed for image classification in continual learning.

We define the difference between the query embeddings $q_{t'}^{f_{t'}}$ and $q_{t'}^{f_t}$ as the *query drift*:

$$\delta^{t' \to t} = q_{t'}^{f_t} - q_{t'}^{f_{t'}}, \tag{4.3}$$

If we have this query drift, the desired embedding $q_{t'}^{f_{t'}}$ can be easily computed with $q_{t'}^{f_{t'}} = q_{t'}^{f_t} - \delta^{t' \to t}$. However, we do not have access to $\delta^{t' \to t}$ as we cannot access the old task training data (the old task queries $q_{t'}$ cannot be used during task t). Therefore, in the following, we propose a method to approximate this drift based on the current task query drift.

After learning a new task, we estimate the drift in the embedding space from f_{t-1} to f_t using the queries from the training data of the current task as shown in fig. 4.2 (left). We project the queries $q \in Q_t$ through f_{t-1} and then through f_t. Now that we have the queries and their corresponding embeddings from f_{t-1} and f_t, we simply estimate the drift of queries and then average those drift vectors to obtain a single drift vector $\Delta^{t-1 \to t}$ for each task t as follows:

$$\Delta^{t-1 \to t} = \frac{1}{\mathcal{N}_t} \sum_{q \in Q_t} (f_t(q) - f_{t-1}(q)), \tag{4.4}$$

where $\mathcal{N}_t$ is the number of queries in training data of task t. Thus, after learning a new task model f_t, we compute the query drift for task t and store the drift vector $\Delta^{t-1 \to t}$.

The drift vector from the last task can be simply added to drift vectors from previous tasks to obtain $\Delta^{t' \to t}$ as follows:

$$\Delta^{t' \to t} = \Delta^{t' \to t'+1} + \cdots + \Delta^{t-1 \to t} = \sum_{j=t'}^{t-1} \Delta^{j \to j+1}, \tag{4.5}$$

During retrieval time, for task t', we pass the queries $Q_{t'}$ through the updated model f_t to obtain embeddings $Q_{t'}^{f_t}$ and then compensate the embeddings by subtracting the drift vector of the corresponding task $\Delta^{t' \to t}$ to project them back to the embedding space of $f_{t'}$ as illustrated in fig. 4.2 (right). For query

Algorithm 3 Proposed Method for Continual Document Retrieval

Continual Training:
Input:
$t \in [1, T]$: task number; (Q_t, D_t): training data; C_t: corpus
f_0: pre-trained model; $\mathcal{H}, \tau$: hyper-parameters
Output:
f_t: Model trained in task t
$\{I_z^{f_z}\}\ \forall\ z \in [1, t]$: Indexed corpus embeddings
$\{\Delta^{z \to z+1}\}\ \forall\ z \in [1, t-1]$: drift vectors

for task $t \in [1, T]$ **do**
  $f_t = f_{t-1}$ # f_{t-1} is frozen and f_t is trainable
  for each (q, d) pair in (Q_t, D_t) **do**
    $\mathcal{L}_C = -\frac{1}{n}\sum_i \log \frac{e^{S(f_t(q_i), f_t(d_i))/\tau}}{\sum_{j=1}^{n} e^{S(f_t(q_i), f_t(d_j))/\tau} + \sum_{h=1}^{\mathcal{H}} e^{S(f_t(q_i), f_t(d_h))/\tau}}$
    if $t > 1$ **then**
      $\mathcal{L}_D = \frac{1}{n}\sum_{i=1}^{n} D_c(f_t(q_i), f_{t-1}(q_i)) + D_c(f_t(d_i), f_{t-1}(d_i))$
      $\mathcal{L} = \mathcal{L}_C + \mathcal{L}_D$
    else
      $\mathcal{L} = \mathcal{L}_C$
    end if
  end for
  # After f_t is trained, we estimate the drift vectors
  if $t > 1$ **then**
    $\Delta^{t-1 \to t} = \frac{1}{\mathcal{N}_t}\sum_{q \in Q_t}(f_t(q) - f_{t-1}(q))$ # Store $\Delta^{t-1 \to t}$
  end if
  # Indexing of corpus documents using trained model f_t
  $I_t^{f_t} \leftarrow f_t(d)\ \forall\ d \in C_t$ # Store document embeddings in $I_t^{f_t}$
end for

$q_{t'} \in Q_{t'}$, we perform the query drift compensation as follows:

$$\hat{Q}_{t'}^{f_{t'}} = \hat{f}_{t'}(q_{t'}) = f_t(q_{t'}) - \Delta^{t' \to t}, \tag{4.6}$$

where $\hat{f}_{t'}(q_{t'})$ is the set of queries which are estimated using the proposed method and is expected to be similar to $f_{t'}(q_{t'})$. Having estimated $\hat{Q}_{t'}^{f_{t'}}$ for queries from task t', we can now compare the query $\hat{q} \in \hat{Q}_{t'}^{f_{t'}}$ and indexed document embeddings $I_{t'}^{f_{t'}}$ in the same embedding space of $f_{t'}$ as follows:

$$\mathcal{I}_i = I_{t'}^{f_{t'}}[i]; \qquad index = \arg\max_i(S(\hat{q}, \mathcal{I}_i)); \qquad \hat{d} = C_{t'}[index], \tag{4.7}$$

Algorithm 4 Retrieval Evaluation after task t:

Input:
f_t: Model from task t
$t' \in [1, t]$: task for evaluation
$Q_{t'}$: test query data from task t'
$C_{t'}$: corpus
$\{I_z^{f_z}\} \forall\ z \in [1, t]$: Indexed corpus embeddings
$\{\Delta^{z \to z+1}\} \forall\ z \in [1, t-1]$: drift vectors
Output:
$\hat{D}_{t'}$: retrieved documents corresponding to $Q_{t'}$

for task $t' \in [1, t]$ **do**
  if $t' < t$ **then**
    $\Delta^{t' \to t} = \sum_{j=t'}^{t-1} \Delta^{j \to j+1}$
    $\hat{Q}_{t'}^{f_{t'}} = f_t(q_{t'}) - \Delta^{t' \to t}\ \forall\ q_{t'} \in Q_{t'}$
  else
    $\hat{Q}_{t'}^{f_{t'}} = f_t(q_{t'})\ \forall\ q_{t'} \in Q_{t'}$ # $t' = t$
  end if
  for $\hat{q}$ in $\hat{Q}_{t'}^{f_{t'}}$ **do**
    $\mathcal{I}_i = I_{t'}^{f_{t'}}[i]$
    $index = \arg\max_i(S(\hat{q}, \mathcal{I}_i))$
    $\hat{d} = C_{t'}[index]$
    $\hat{D}_{t'} \leftarrow \hat{d}$ # Store retrieved documents in $\hat{D}_{t'}$
  end for
end for

where $\mathcal{I}_i$ refers to embeddings from the indexed corpus $I_{t'}^{f_{t'}}$ at index i and S refers to the cosine similarity. Thus, we resolve the non-compatibility issue by simple vector subtraction without any re-indexing. We summarize the proposed approach for training in algorithm 3 and retrieval evaluation in algorithm 4. In the experiments, we also explore estimating multiple query drift vectors per task, but do not find this to significantly improve results.

4.5 Experiments

4.5.1 Experimental Setup

Datasets. We use 5 retrieval datasets from the BEIR benchmark [166] and continually train and evaluate them in task-incremental learning setup. We use the datasets in the following sequence - MS MARCO [4], NQ [85], Hot-

Table 4.2: Details of datasets used in the proposed benchmark for CDR. Details excerpted from [166].

Split (→)			Train	Test			Avg. Word Lengths	
Task (↓)	Domain (↓)	Dataset (↓)	#Pairs	#Query	#Corpus	Avg. D / Q	Query	Document
Passage-Retrieval	Misc.	MS MARCO	532,761	6,980	8,841,823	1.1	5.96	55.98
Question Answering (QA)	Wikipedia	NQ	132,803	3,452	2,681,468	1.2	9.16	78.88
	Wikipedia	HotpotQA	170,000	7,405	5,233,329	2.0	17.61	46.30
	Finance	FiQA-2018	14,166	648	57,638	2.6	10.77	132.32
Fact Checking	Wikipedia	FEVER	140,085	6,666	5,416,568	1.2	8.13	84.76

potQA [195], FEVER [167] and FiQA-2018 [110]. We select these datasets since they have sufficient training set of query-document pairs and hence we can continually train them. The other datasets from BEIR benchmark has very little to no training data and are primarily used for zero-shot retrieval evaluation. We discuss the details of the datasets in table 4.2.

Implementation. We use the nomic embedding model [126] with 768 dimensional feature embeddings for our experiments. We use the MTEB benchmark [122] for evaluating the retrieval models on different tasks. We use the pre-trained model[†] from [126] which is pre-trained with MLM followed by unsupervised contrastive pre-training. The nomic model is a modified version of BERT base [29] resulting in a 137M parameter model with 8192 sequence length. Starting with the pre-trained model, we fine-tune them continually for our experiments. Following [126], we use $\mathcal{H} = 7$ hard negatives for each query which are mined from the corresponding dataset corpus using the gte-base model[‡] [94]. For each task, we train for one epoch with a batch size of 128, learning rate of 2×10^{-5} and weight decay of 0.01. Similar to [126], we find that training for more epochs does not improve performance. We use 4 NVIDIA L40S GPUs to train the models for our experiments. While we compare the nomic model performance with other dense retrievers in table 4.9, the main evaluation for the continual setting is based on the pre-trained nomic model which we continually finetune on new tasks.

Metrics. We use the Normalised Cumulative Discount Gain (nDCG@10) metric [198] for top-10 retrieved documents in our evaluations following [51, 126, 166]. We also report other metrics like Recall@10 and MAP@10 (Mean Average Precision) in section 4.5.3.

[†]nomic pre-trained model (https://huggingface.co/nomic-ai/nomic-embed-text-v1-unsupervised)

[‡]gte-base model (https://huggingface.co/thenlper/)

Table 4.3: Performance comparison of different approaches for Continual Document Retrieval tasks. Here, we report the nDCG@10 scores for retrieval of each task. For continually trained methods, we use the latest model after training on T5 for retrieval of all tasks. We highlight the proposed QDC-based approaches in purple. †: results excerpted from [166].

Method	Continual	T1 - MS MARCO	T2 - NQ	T3 - Hotpot QA	T4 - FEVER	T5 - FiQA2018	Avg. Score
BM25†	✗	22.8	32.9	60.3	75.3	23.6	43.0
Joint Training	✗	39.2	50.7	71.7	75.1	33.8	54.1
FT	✓	34.8	50.4	59.1	68.2	34.4	49.4
FT + QDC	✓	38.4	52.5	67.5	75.0	34.4	53.5
FT + KD	✓	37.6	52.6	65.8	63.7	34.3	50.8
FT + KD + QDC	✓	39.3	54.1	69.3	73.6	34.3	**54.1**
FT (with re-indexing)	✓	33.8	45.7	62.9	73.5	34.4	50.1
FT + KD (with re-indexing)	✓	34.1	46.7	64.1	72.6	34.3	50.4

4.5.2 Experimental Results

Comparison to BM25. We compare with commonly used lexical retriever BM25 [141] in table 4.3. We observe that BM25 performs very poorly on most datasets compared to FT or joint training except on FEVER where it outperforms all other approaches.

Impact of KD. We show in table 4.3 that embedding knowledge distillation (KD) improves the stability of the model which is evident from better performance of old tasks where FT+KD improves MS MARCO by 2.8%, NQ by 2.2% and HotpotQA by 6.7% over FT. While KD improves over FT by 1.4% on an average, we also observe that KD can affect the plasticity of the model since the performance on newer tasks are affected (FEVER drops by 4.5%).

Impact of QDC. We show in table 4.3 that QDC outperforms FT and FT+KD and improves the performance of all tasks after continually training on all tasks. Using QDC improves over FT across all tasks by 4.1% on average. When used with FT+KD models, QDC improves by 3.3% on average and outperforms all other approaches. We also evaluate the impact of QDC on the performance by evaluating on all tasks after training on each task in table 4.4. We observe poor performance of old tasks (in yellow) with naive fine-tuning. When using QDC for retrieval of old tasks, the performance improves for all tasks, thereby reducing the forgetting significantly (denoted by PD). Finally, using QDC with FT+KD models achieves the best average accuracy and least PD after each task.

Comparison to joint training. We compare the performance of continually trained model with the jointly trained model, which is trained on all five datasets at the same time in a static setting. We observe that the proposed

Table 4.4: Performance of the proposed method for Continual Document Retrieval tasks after training on each task. Here, we show the retrieval performance (nDCG@10 scores) for all datasets as the model is continually trained on a new task. We denote the performance of old tasks in yellow and the zero-shot retrieval performance of future tasks in green. The performance drop (PD) from the task in which the model is learned on a given dataset (denoted by TX) to the last task suggests the forgetting of that dataset.

		Training Sequence →					
	Eval on	T1 - MS MARCO	T2 - NQ	T3 - HotpotQA	T4 - FEVER	T5 - FiQA2018	PD (TX - T5)
FT	MS MARCO	40.2	38.7	38.2	36.3	34.8	5.4
	NQ	50.4	54.7	51.6	50.7	50.4	4.3
	HotpotQA	61.3	56.4	71.5	68.8	59.1	12.4
	FEVER	67.8	58.0	66.1	72.8	68.2	4.6
	FiQA2018	31.4	32.4	29.4	29.3	34.4	-
	Avg Acc.	50.2	48.1	51.4	51.6	49.4	
	Eval on	T1 - MS MARCO	T2 - NQ	T3 - HotpotQA	T4 - FEVER	T5 - FiQA2018	PD (TX - T5)
FT+QDC	MS MARCO	40.2	39.9	38.5	37.8	38.4	1.8
	NQ	50.4	54.7	53.0	50.8	52.5	2.2
	HotpotQA	61.3	56.4	71.5	70.5	67.5	4.0
	FEVER	67.8	58.0	66.1	72.8	75.0	-2.2
	FiQA2018	31.4	32.4	29.4	29.3	34.4	-
	Avg Acc.	50.2	48.3	51.7	52.2	53.5	
	Eval on	T1 - MS MARCO	T2 - NQ	T3 - HotpotQA	T4 - FEVER	T5 - FiQA2018	PD (TX - T5)
FT+KD+QDC	MS MARCO	40.2	40.2	39.4	39.0	39.3	0.9
	NQ	50.4	54.7	54.1	52.8	54.1	0.6
	HotpotQA	61.3	58.7	72.7	72.5	69.3	3.4
	FEVER	67.8	61.1	67.4	70.5	73.6	-3.1
	FiQA2018	31.4	32.7	30.3	29.3	34.3	-
	Avg Acc.	50.2	49.5	52.8	52.8	54.1	

method (FT+KD+QDC) performs similarly to the jointly trained model on average. Note that the joint training here considers only those hard-negatives which are mined from each dataset and are not jointly mined. In other words, we use the same hard-negatives for each dataset in both continual finetuning and joint training for fair comparison. While the joint training performance could improve with jointly mined hard-negatives, we follow the standard practice of joint training [126].

Generalizability. We also evaluate the zero-shot performance in table 4.4 (in green) for unseen tasks and demonstrate how training on each new task affects the generalizability of the model. The zero-shot performance depends on the latest task the model is trained on. For instance, we see a drop in performance of FEVER after training on NQ while it improves after training on Hotpot QA. This could be due to high domain overlap between Hotpot QA and FEVER as shown in [166]. We also observe an improvement in zero-shot performance

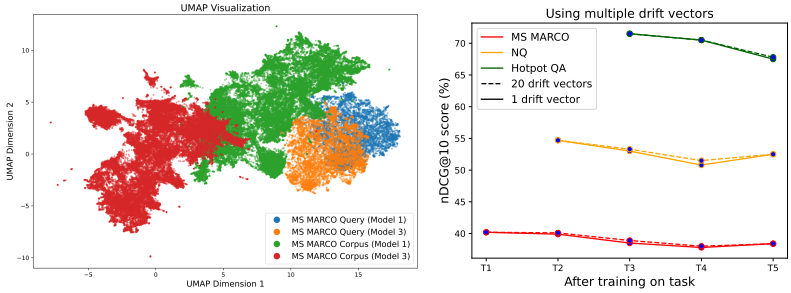


Figure 4.3: (left) UMAP visualization of the drift in query and corpus embeddings of MS MARCO after fine-tuning on NQ and Hotpot QA (using Model after task 3). (right) Analysis of the performance when using multiple drift vectors to represent drift between embedding spaces of old and new task.

when using KD.

Comparison to re-indexing. While re-indexing was found to be effective in image retrieval [153], we show that in CDR, re-indexing of old task documents does not improve performance significantly and performs worse compared to FT on initial tasks like MS MARCO and NQ. The proposed method QDC significantly outperforms re-indexing approach, even with re-indexing on the model trained with embedding distillation as shown in table 4.3. Previously, Wang et al., [183] observed unequal drifts in image and text modalities for continual multimodal retrieval. We observe a similar phenomenon here and show in fig. 4.3 (left) that the drift in corpus document embeddings is higher compared to the drift in query embeddings for MS MARCO after training on NQ followed by Hotpot QA. Thus, the corpus document embeddings (in red) are poorly aligned with the query embeddings (in orange) in the updated embedding space after task 3.

We attribute the poor performance of re-indexing in CDR to the unequal drift in query and document embeddings. We hypothesize that the unequal drift could be due to the difference in length of queries and documents. Queries are much shorter in length while documents could be a paragraph. From table 4.2, we observe that documents have a much higher average word lengths than queries (MS MARCO documents are 10 times longer than the queries on an average). This suggests that the documents could face higher forgetting or higher embedding drift compared to queries.

In the case of multi-modal retrieval [183], the unequal drift arises from the modality gap at initialization which is preserved during the contrastive training approach (as discussed in [98] for vision-language models). A recent

Table 4.5: Average drift values for Query and Corpus across text lengths.

	Short	Medium	Long
Query	0.0340	0.0375	0.0393
Corpus	0.0559	0.0573	0.0635

work [149] extensively analyzed and presented that the gap between modalities actually arises from the information imbalance in the two modalities where one modality (visual) has more information than the other one (text). We think that this explanation could also be applicable to our setting despite that we only have a single modality, since there is a significant information imbalance between queries and corpus. The imbalance in information is caused by the difference in the lengths of queries and corpus where queries are usually much shorter in length and the corpus are much longer and more detailed.

We analyze in table 4.5 how the queries and corpus of different lengths of the first task MS Marco drift after training on NQ. We divide the queries and corpus into three groups (short, medium and long) based on lengths and report the average of the cosine drift in each group. We observe that corpus documents having more information drift more than the queries as also seen in fig. 4.3 (left). We observe that within the queries, the medium-sized and longer queries drift more on average than the shorter ones. Similarly, among the corpus documents, we see that longer documents drift more than the short ones. This confirms the correlation between the length of the query or document and the drift.

4.5.3 Analysis and Ablation Studies

Enabling compatibility with QDC. There are two ways of maintaining query and corpus compatibility by keeping them in the same embedding space, one by re-indexing (bringing documents forward to the new space) and the other by projecting queries back to the old embedding space. While re-indexing solves the embedding space alignment problem and brings query and corpus in the same space, it does not maintain good discriminative power for the old task corpus documents since it uses the updated model which is not the best model for the old task. On the other hand, we preserve the discriminative capabilities of the model to encode the corpus in QDC since we use the documents indexed with the old model as shown in table 4.6.

So, despite adding more computation costs, FT with re-indexing uses the updated model with reduced discriminative capabilities for both queries and

Table 4.6: Comparison of methods and their properties across tasks.

Task	Method	Query Embedding	Corpus Embedding	Compatibility	Discriminative Power for T1
T1	FT	T1	T1	✓	✓
T2	FT	T2	T1	✗	✓
T2	FT with re-indexing	T2	T2	✓	✗
T2	FT+QDC	T1	T1	✓	✓

the corpus of the old task. While QDC still uses the best model for the old task for indexing the documents and the drift compensated queries solves the alignment problem. Note that the corpus embeddings play a more important role here since a single query embedding is searched across millions of corpus documents. So, it is more important to preserve the corpus embedding space for old tasks and the best way is to use the old model indexed embeddings.

QDC with multiple drift vectors. In the proposed method, we consider a single drift vector for each task transition. One could also estimate multiple drift vectors in the embedding space. This could be done by dividing the embedding space into k clusters with corresponding centroids P_k and estimating a drift vector for each of these centroids $\Delta_k^{t-1 \to t}$ by averaging the drift of queries belonging to each cluster. During retrieval, the queries $q_{t'}$ could be assigned to the closest centroid k' and then compensated with the drift vector of the centroid $\Delta_{k'}^{t' \to t}$ as follows:

$$k' = \arg\max_k S(P_k, f_t(q_{t'})); \qquad \hat{f}_{t'}(q_{t'}) = f_t(q_{t'}) - \Delta_{k'}^{t' \to t} \tag{4.8}$$

where k' is the closest cluster to the query embedding. Using multiple drift vectors involves storing the cluster centroids and the drift vectors for each centroid.

In our experiments in fig. 4.3 (right), we empirically demonstrate that using a single drift vector is effective to estimate the query drift and estimating multiple drift vectors for each task transition does not improve the performance significantly. Using 20 drift vectors improves the performance of MS MARCO by 0.4% after T3, NQ by 0.7% after T4, Hotpot QA by 0.3% after T5, while achieving the same performance for FEVER. Based on these observations, we advocate using a single drift vector for each task which achieves similar performance with simpler and faster retrieval since it does not need to find the closest cluster centroid for drift compensation.

Using QDC in Class-Incremental Learning Setting It would be interesting to extend QDC to CIL setting with no access to task-id and future

works could explore that setting. Similar to several existing works in CIL which predicts task-id (see discussion in [74]), the proposed method QDC could be adapted to CIL setting by predicting the task-id of a given query as discussed below.

- After training on each task, the feature centroid of that task can be stored. Centroids of all old tasks could be updated by adding the drift vector $\Delta^{t-1 \to t}$ of the current task. So, at the end of task t, we have the task centroids for all tasks in the updated embedding space.
- During inference with queries extracted using the latest task model, we can predict the task-id based on cosine distance between query embedding and the task centroids.
- After predicting the task-id for queries, we can perform QDC to move the query back to the embedding space of the task.

Performance evaluation with other metrics We show the performance of different methods using other metrics like recall@10 and MAP@10 in tables 4.7 and 4.8. We observe the same trend that using QDC outperforms FT and FT+KD and achieves similar performance as joint training.

Table 4.7: Performance comparison of different approaches for Continual Document Retrieval tasks. Here, we report the **recall@10** scores for retrieval of each task. For continually trained methods, we use the latest model after training on T5 for retrieval of all tasks. We highlight the proposed QDC-based approaches in purple.

Method	T1 - MS MARCO	T2 - NQ	T3 - Hotpot QA	T4 - FEVER	T5 - FiQA2018	Average
Joint Training	60.8	72.8	75.5	87.9	40.3	67.5
FT	54.2	71.3	64.5	81.9	41.1	62.6
FT + QDC	59.5	73.3	72.2	87.1	41.1	66.6
FT + KD	58.2	73.3	70.6	78.2	41.4	64.3
FT + KD + QDC	60.6	75.3	73.7	86.1	41.4	**67.4**
FT (with re-indexing)	53.6	67.0	67.6	87.4	41.1	63.3
FT + KD (with re-indexing)	53.8	68.1	69.2	86.1	41.4	63.7

Comparison of Nomic pre-trained model with other retrieval models We compare the performance of the nomic retriever model trained either jointly or continually with classical dense retrievers like DPR [70], ColBERT [72] and ANCE [192]. We report performance of DPR, ColBERT and ANCE from BEIR [166]. While the other dense retriever models like ColBERT perform competitively, the nomic retriever model outperforms them. For the continual

Table 4.8: Performance comparison of different approaches for Continual Document Retrieval tasks. Here, we report the **MAP@10** scores for retrieval of each task. For continually trained methods, we use the latest model after training on T5 for retrieval of all tasks. We highlight the proposed QDC-based approaches in purple.

Method	T1 - MS MARCO	T2 - NQ	T3 - Hotpot QA	T4 - FEVER	T5 - FiQA2018	Average
Joint Training	32.2	42.6	63.9	69.7	26.5	47.0
FT	28.6	42.8	50.6	62.7	27.0	42.3
FT + QDC	31.6	44.8	59.3	69.9	27.0	46.5
FT + KD	30.9	44.9	57.4	58.1	26.9	43.6
FT + KD + QDC	32.4	46.2	61.2	68.5	26.9	**47.0**
FT (with re-indexing)	27.4	37.8	54.9	67.9	27.0	43.0
FT + KD (with re-indexing)	27.7	38.9	56.0	67.1	26.9	43.3

Table 4.9: Performance comparison of different approaches. We report the nDCG@10 scores for retrieval of each task. Here, we do not consider continual training and evaluate the performance on each task separately. †: results excerpted from [166].

Method	Continual	T1 - MS MARCO	T2 - NQ	T3 - Hotpot QA	T4 - FEVER	T5 - FiQA2018	Avg. Score
BM25†	✗	22.8	32.9	60.3	75.3	23.6	43.0
DPR†	✗	17.7	47.4	39.1	56.2	11.2	34.3
ANCE†	✗	38.8	44.6	45.6	66.9	29.5	45.1
ColBERT†	✗	40.1	52.4	59.3	77.1	31.7	52.1
Joint Training (Nomic)	✗	39.2	50.7	71.7	75.1	33.8	**54.1**
FT + KD + QDC (Nomic)	✓	39.3	54.1	69.3	73.6	34.3	**54.1**

setting, we base the comparison on the nomic model. Benchmarking the performance of these other retriever architectures by continually training them in the proposed continual setting could be interesting to explore in future works.

Retrieval examples We present some retrieval results using the continually fine-tuned model in tables 4.10 and 4.11 for old tasks (MS MARCO and Hotpot QA). We show that using the proposed QDC method with fine-tuned model retrieves more relevant documents from the corpus for a given query.

4.6 Conclusion

In this work, we study how continually training embedding models on query-document pairs from new datasets over time could affect the retrieval performance across all seen tasks. We observe forgetting on old tasks in CDR and show that using knowledge distillation on the query and document embeddings

Table 4.10: Examples of top-1 retrieved document for a given query using fine-tuned model and with the proposed QDC (FT+QDC) for **MS MARCO** dataset using the continually trained model after task 5.

Query	Document retrieved with FT	Document retrieved with FT+QDC
do vhi swiftcare do blood tests?	Blood tests. Information on having blood tests and the types of blood tests you might have. Your blood sample is sent to the laboratory. A blood doctor can look at your sample under a microscope. They can see the different types of cells and can count the different blood cells.	The SwiftCare clinics charge an initial consultation fee of 85 euro, with additional charges for tests and procedures. For example, an x-ray at the clinics costs 65 euro, blood tests range from 30 to 50 euro and complex suturing costs 50 euro. The Swiftcare clinics are staffed by doctors with significant experience in general practice and emergency care, according to VHI. The clinics are run as a joint initiative between the VHI and The Well, a primary care care medical company.
the miners state bank routing number	Search all THE MINERS STATE BANK routing numbers in the table below. Use the Search box to filter by city, state, address, routing number. Click on the routing number link in the table below to navigate to it and see all the information about it (address, telephone number, zip code, etc.).	The Miners State Bank Routing Number the miners state bank routing aba number 091109253 routing number is a 9-digit number designed and assigned to The Miners State Bank by The American Bankers Association (ABA) to identify the financial institution upon which a payment was drawn.
cadillac alternator price	1 On average, a car alternator prices are going to range anywhere from 66*to*320. 2 This is not going to include the labor costs. 3 When you factor in labor costs, itâ safe to add another 100*to*275.	A Cadillac De Ville Alternator Replacement costs between 378*and*860 on average. Get a free detailed estimate for a repair in your area. A Cadillac De Ville Alternator Replacement costs between 378*and*860 on average.
door knocker definition	Definition of 'knocker'. knocker. A knocker is a piece of metal on the front door of a building, which you use to hit the door in order to attract the attention of the people inside.	Door Knocker definition. An act of physical violence performed on a person (usually a woman) who is wearing huge hoop earrings. A cob of dried corn employed by pranksters on Mischief evening to toss at people's doors.....notable for loud BLANG!! or KABAAM!! Not what title really seems.

can reduce the forgetting. We discuss the issue of non-compatibility between query and indexed corpus embeddings due to *embedding drift* after the embedding model is updated on a new task. We propose a novel method to avoid this issue by estimating the drift of queries from old to new embedding space and then compensating the estimated drift to project the queries to old embedding space at test time. This enables compatibility since the indexed embeddings were extracted from the old model and thus we compute similarities for retrieval with queries and corpus in the same embedding space. We establish a continual training benchmark with five large-scale datasets and demonstrate that the proposed QDC approach outperforms other approaches. We also show that re-indexing based approach does not perform well despite being very expensive.

We believe that enabling compatibility in continually trained retrieval embedding models will benefit several practical document retrieval applications like RAG systems where the retriever embedding model could be continually updated to add new knowledge over time. We hope that our approach and findings will encourage further research and more extensive benchmarks on continual document retrieval.

Table 4.11: Examples of top-1 retrieved document for a given query using fine-tuned model and with the proposed QDC (FT+QDC) for **Hotpot QA** dataset using the continually trained model after task 5.

Query	Document retrieved with FT	Document retrieved with FT+QDC
This singer of A Rather Blustery Day also voiced what hedgehog?	Pinocchio (singer) Pinocchio is a fictional, animated French character and singer.	A Rather Blustery Day A Rather Blustery Dayis a whimsical song from the Walt Disney musical film featurette, Winnie the Pooh and the Blustery Day. It was written by Robert & Richard Sherman and sung by Jim Cummings as Pooh.
What WB supernatrual drama series was Jawbreaker star Rose Mcgowan best known for being in?	Charmed (season 4) The fourth season of Charmed, an American supernatural drama television series, began airing on October 4, 2001 on The WB. Airing on Thursdays at 9:00 pm, the season consisted of 22 episodes and concluded its airing on May 16, 2002. This season also saw the introduction of Rose McGowan as Paige Matthews—half-sister to Prue, Piper and Phoebe—and a slight alteration of the opening credits, due to the third season departure of Shannen Doherty as Prue. Paramount Home Entertainment released the complete fourth season in a six-disc boxed set on February 28, 2006.	Rose McGowan Rose Arianna McGowan (born September 5, 1973) is an Italian-born American actress, film producer, director and singer. She is best known to television audiences for having played Paige Matthews in The WB supernatural drama series Charmedfrom 2001 to 2006.
In which role did Caroline Carver played in a 1999 Hallmark Entertainment made-for-TV fantasy movie?	The Magical Legend of the Leprechauns The Magical Legend of the Leprechauns is a 1999 Hallmark Entertainment made-for-TV fantasy movie. It stars Randy Quaid, Colm Meaney, Kieran Culkin, Roger Daltrey, Caroline and Whoopi Goldberg. The film contains two main stories that eventually Carver intertwine: the first being the story of an American businessman who visits Ireland and encounters magical leprechauns and the second, a story of a pair of star-crossed lovers who happen to be a fairy and a leprechaun, belonging to opposing sides of a magical war. It contains many references to Romeo and Juliet such as two lovers taking poison and feuding clans.	Caroline Carver (actress) Caroline Carver (born 1976) is an English actress, screenwriter, and producer best known for roles such as Princess Jessica in the TV film The Magical Legend of the Leprechauns(1999), Ingrid in The Aryan Couple(2004), and Sandy in My First Wedding(2006).
What's the name of the fantasy film starring Sarah Bolger, featuring a New England family who discover magical creatures around their estate?	Fredegar Bolger Fredegar FattyBolger is a fictional character in J. R. R. Tolkien's fantasy novel The Lord of the Rings.	Sarah Bolger Sarah Lee Bolger (born 28 February 1991) is an Irish actress. She is best known for her roles in the films In America, Stormbreaker, and The Spiderwick Chronicles, as well as her award-winning role as Lady Mary Tudor in the TV series The Tudors, and for guest starring as Princess Aurora in Once Upon a Time.

5 Covariances for Free: Exploiting Mean Distributions for Training-free Federated Learning*

5.1 Introduction

Motivated by results from transfer learning [52], several recent works on FL have studied the impact of using pre-trained models and observe that it can significantly reduce the impact of data heterogeneity [21, 88, 106, 125, 135, 156, 163, 164, 189]. An important finding in several of these works is that sending local class means to the server instead of raw features is more efficient in terms of communication costs, eliminates privacy concerns, and is robust to gradient-based attacks [21, 215]. The authors of [164] used pre-trained models to compute and then share class means as the representative of each class, and in [88] the authors showed that aggregating local means into global means and setting them as classifier weights (FedNCM) achieves very good performance without any training. FedNCM incurs very little communication cost and enables stable initialization. Recently, the authors of Fed3R [38] explored the impact of sharing second-order feature statistics from clients to server to solve the ridge regression problem [16] in FL and improve over FedNCM. The sharing of second-order statistics has also previously been explored for classifier calibration after federated optimization [106].

Although it is evident that exploiting second-order feature statistics results in better and more stable classifiers, it poses new problems. Notably, sharing second-order statistics for high-dimensional features from clients to the server significantly increases communication overhead and exposes clients to privacy risks [38, 106]. To reap the benefits of second-order client statistics, while at the same time mitigating these risks, in this chapter we propose Federated learning with COvariances for Free (FedCOF) which only communicates class means from clients to the server (see table 5.1). We show that, from just these class means and the mathematical relationship between their covariance and the class covariance matrices, we can compute a provably unbiased estimator of global

*This chapter is based on a publication in Advances in Neural Information Processing Systems, 2025 [46].

Table 5.1: FedNCM [88] shares only class means $\hat{\mu}_{k,c}$ and has minimal communication. Fed3R [38] requires sum of class features B_k and feature matrix G_k from all clients, thereby increasing the communication cost by d^2K. We propose FedCOF, which shares only class means and estimates a global class covariance $\hat{\Sigma}_c$ to initialize the classifier weights. Note that only a small subset of all classes are present in each client. For simplicity, we show the upper bound of communication cost here where C' denotes the maximum number of classes present in a single client.

Method	Client Shares	Server Uses	Comm. Cost
FedNCM	$\{\hat{\mu}_{k,c}, n_{k,c}\}_{c=1}^{C'}$	$\{\{\hat{\mu}_{k,c}, n_{k,c}\}_{c=1}^{C'}\}_{k=1}^{K}$	$dC'K$
Fed3R	G_k, B_k	$\{G_k, B_k\}_{k=1}^{K}$	$(dC' + d^2)K$
FedCOF	$\{\hat{\mu}_{k,c}, n_{k,c}\}_{c=1}^{C'}$	$\{\{\hat{\mu}_{k,c}, n_{k,c}\}_{c=1}^{C'}\}_{k=1}^{K}, \{\hat{\Sigma}_c\}_{c=1}^{C'}$	$dC'K$

class covariances on the server. FedCOF is a training-free method that exploits pre-trained feature extractors. It uses the same communication budget as FedNCM while delivering performance comparable to or even superior to Fed3R (see fig. 5.1). Additionally, we show how to improve classifier initialization using only within-class covariances, setting the classifier weights based on aggregated class means and our estimated class covariances. To summarize, our main contributions are:

- We propose FedCOF, a training-free method that exploits a *provably unbiased estimator of class covariances*, which requires only class means from clients, thus avoiding the need to share second-order statistics, reducing communication costs and mitigating privacy concerns.
- We show how FedCOF leverages *within-class covariances*, estimated solely from client means, to initialize the global classifier, yielding more stable and better-conditioned solutions than using both within- and between-class scatter matrices.
- We validate FedCOF across multiple FL benchmarks, including the non-iid iNaturalist-Users-120K dataset, achieving state-of-the-art results with lower communication cost than methods using second-order statistics, and showing superior performance to recent federated prompt-tuning approaches while also serving as an effective initialization for subsequent federated optimization methods such as fine-tuning and linear probing.

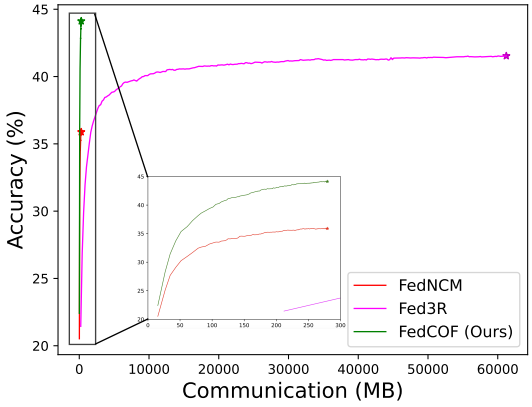


Figure 5.1: Accuracy vs. communication cost using pre-trained MobileNetv2 on iNaturalist-Users-120K. FedCOF outperforms Fed3R while having the same communication cost as FedNCM.

5.2 Related Work

Federated learning. FL focuses on neural network training in distributed environments [188, 201]. Initial works like FedAvg [115] proposed training by averaging of distributed models. Later works focus more on non-iid settings, where data among the clients is more heterogeneous [68, 92, 93, 181]. FedNova [182] normalizes local updates before averaging to address objective inconsistency. Scaffold [69] employs control variates to correct drift in local updates. FedProx [91] introduces a proximal term in local objectives to stabilize the learning process. [139] proposed use of adaptive optimization methods, such as Adagrad, Adam and Yogi, at the server side. While CCVR [106] proposed a classifier calibration method that aggregates class means and covariances from clients, other recent works [32, 75, 95, 127] proposed using fixed classifiers inspired by the neural collapse phenomenon. After federated training with fixed classifiers, FedBABU [127] proposed to fine-tune the classifiers and SphereFed [32] proposed a closed-form classifier calibration.

FL with pre-trained models. While conventional FL methods start training from scratch without any pre-training, we focus on the FL setting using pre-trained models. Multiple works propose using pre-trained weights for FL which reduces the impact of client data heterogeneity and achieves faster model convergence [21, 125, 135, 156, 164, 189]. Federated prompt-tuning

methods [189] proposed to reduce the communication cost by tuning only prompts in a federated manner using pre-trained ViT models. Recently, it has been shown that *training-free methods* using pre-trained networks, achieve strong performance without any training by exploiting feature class means [88] or second-order feature statistics [38]. Another recent work [5] proposed sharing Gaussian mixture models (multiple sets of means and covariances) for each class from clients to server for one-shot FL. In this work, we propose a training-free method with pre-trained models that estimates class covariances from only client means for initializing the global classifier.

5.3 Preliminaries

In the FL setting we assume K clients, each with a local dataset $D_k = (X_k, Y_k)$, for $k \in \{1, ..., K\}$, and denote by N the total number of images across all clients. The model is composed of a feature extractor f, parameterized by θ, which maps images to d-dimensional embeddings, and a classifier $h : \mathbb{R}^d \to \mathbb{R}^C$, parameterized by W, where C is the total number of classes. In federated optimization, each client trains its local model on its private data D_k and transmits only intermediate information – such as parameter updates or feature statistics – while a central server aggregates these signals to minimize a global objective, without revealing raw data [78].

With the growing availability of high-quality pre-trained models, recent works have focused on scenarios in which all clients are initialized with the same pre-trained feature extractor [21, 38, 88, 125, 164, 189]. Notably, *training-free* methods [38, 88] – in which only the global classifier is initialized using client feature statistics, without training the feature extractor – achieve strong performance. They often outperform federated full fine-tuning methods, such as FedAdam and FedAvg, which train client backbones and classifiers, starting from the same shared pre-trained feature extractor but with randomly initialized classifiers, at a fraction of the communication and computation costs. Moreover, this training-free initialization can serve as an effective starting point, improving the performance of federated full fine-tuning. We now discuss recent training-free methods.

Federated NCM. FedNCM [88] employs a Nearest Class Mean (NCM) classifier, where the global linear classifier weights for class c (denoted as W_c) are initialized as $\hat{\mu}_c / \|\hat{\mu}_c\|$. Here, $\hat{\mu}_c$ represents the global class mean aggregated

from the local client class means $\hat{\mu}_{k,c}$ as follows:

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{k=1}^{K} n_{k,c}\, \hat{\mu}_{k,c}; \quad \hat{\mu}_{k,c} = \frac{1}{n_{k,c}} \sum_{x \in X_{k,c}} f(x), \tag{5.1}$$

where $X_{k,c}$ is a subset of X_k having images of class c, $n_{k,c}$ refers to number of images in $X_{k,c}$, and $N_c = \sum_{k=1}^{K} n_{k,c}$ is the number of images of class c across all clients.

Federated Ridge Regression. While FedNCM exploits only class means, Fed3R [38] recently proposed using ridge regression which requires second-order feature statistics from all clients to initialize the global classifier, leading to improved performance compared to FedNCM. The ridge regression problem aims to find the optimal weights that minimize the following objective:

$$W^* = \arg \min_{W \in \mathbb{R}^{d \times C}} \|Y - F^\top W\|^2 + \lambda \|W\|^2, \tag{5.2}$$

where $F \in \mathbb{R}^{d \times N}$ is the feature matrix extracted using a pre-trained model and $Y \in \mathbb{R}^{N \times C}$ contains one-hot encoded labels for the N features over C classes. The closed-form solution is given by:

$$W^* = (G + \lambda I_d)^{-1} B, \tag{5.3}$$

where $G = FF^\top \in \mathbb{R}^{d \times d}$, $B = FY \in \mathbb{R}^{d \times C}$, $I_d \in \mathbb{R}^{d \times d}$ is the identity matrix, and $\lambda \in \mathbb{R}$ is a hyperparameter. In Fed3R, each client k computes two local matrices $G_k = F_k F_k^\top \in \mathbb{R}^{d \times d}$ and $B_k = F_k Y_k \in \mathbb{R}^{d \times C}$, where F_k and Y_k are the feature matrix and the labels of client k, and then sends them to the global server. The server aggregates these matrices as $G = \sum_{k=1}^{K} G_k, \quad B = \sum_{k=1}^{K} B_k$ to compute W^*, which is normalized and used to initialize the global classifier.

5.4 Federated Learning with COvariances for Free (FedCOF)

In this section, we derive our FedCOF method which is based on a provably unbiased estimator of class covariances and requires transfer of only class means from clients to server.

5.4.1 Motivation

Communication cost. While Fed3R is more effective than FedNCM, it requires each client to send an additional $d \times d$ matrix, significantly increasing the communication overhead by d^2K compared to FedNCM which only shares the class means (see table 5.1). Fed3R scales linearly with number of clients and quadratically with the feature dimension. FedPFT [5] shares multiple sets of $(\mu_{k,c}, \Sigma_{k,c})$ from clients, further increasing communication costs. Considering cross-device FL settings [68], having millions of clients, the communication cost required for these methods would be enormous.

Recent works [79, 189] focus on parameter-efficient federated fine-tuning where the per-round communication costs are significantly reduced, enabling applications in low-bandwidth communication settings. For example, recent work [79] shows that using pre-trained language models (RoBERTa-base) can yield performance similar to full fine-tuning while using rank-1 LoRA updates and thereby reducing communication cost by 99.8%. Sharing a similar goal of reducing communication costs, we propose an unbiased covariance estimator without sharing client covariances for training-free FL.

Potential privacy concerns. Sharing only class means provides a higher level of data privacy compared to sharing raw data, as prototypes represent the mean of feature representations. It is not easy to reconstruct exact images from prototypes with feature inversion attacks [106]. As a result, sharing class means is common in many recent works [88, 156, 163, 164]. On the other hand, Fed3R shows that sharing second-order statistics improves performance compared to sharing class means. However, this could expose the feature distribution of clients to the server since all clients employ the same frozen pre-trained model to extract features [38]. Sharing covariances makes clients more vulnerable to attacks if secure aggregation protocols are not implemented [14].

5.4.2 Estimating Covariances Using Only Client Means

Our method leverages the statistical properties of sample means to derive an unbiased estimator of the class population covariance based only on per-client class means (see fig. 5.2). We model the global features of each class c as drawn from a multivariate distribution with mean μ_c and covariance Σ_c, and assume that the local features for class c computed by each client using the shared *frozen* pre-trained model are iid. Under this assumption, class features in a client form a random sample from the class population.

Let $\{F_{k,c}^j\}_{j=1}^{n_{k,c}}$ be a random sample from a multivariate population with mean μ_c and covariance Σ_c, where $F_{k,c}^j$ is the j-th feature vector of class c

assigned to the client k and $n_{k,c}$ is the number of elements of class c in the client k. Assuming that the per-class features $F_{k,c}^j$ in each client are iid in the initialization, then the sample mean of the features for class c

$$\overline{F}_{k,c} = \frac{1}{n_{k,c}} \sum_{j=1}^{n_{k,c}} F_{k,c}^j, \tag{5.4}$$

is distributed with mean and covariance given by:

$$\mathbb{E}[\overline{F}_{k,c}] = \mu_c \qquad \text{Var}[\overline{F}_{k,c}] = \frac{\Sigma_c}{n_{k,c}} \tag{5.5}$$

Proof To prove this, we fix the class c and omit the dependencies on c for simplicity. Thus, we write $n_{k,c} = n_k$, $F_{k,c}^j = F_k^j$, $\overline{F}_{k,c} = \overline{F}_k$, and $\mu_c = \mu$, $\Sigma_c = \Sigma$.

Since $\{F_k^j\}_{j=1}^{n_k}$ is a random sample from a multivariate distribution with mean μ and covariance Σ, and the per-class features F_k^j in each client are i.i.d at initialization, it follows that:

$$\mathbb{E}[F_k^j] = \mu \qquad \text{Var}[F_k^j] = \Sigma, \quad \forall j \tag{5.6}$$

By computing the expectation of $\overline{F}_k$ and using the linearity of expectation, we obtain:

$$\mathbb{E}[\overline{F}_k] = \mathbb{E}[\frac{1}{n_k} \sum_{j=1}^{n_k} F_k^j] = \frac{1}{n_k}\mathbb{E}[F_k^1] + \ldots + \frac{1}{n_k}\mathbb{E}[F_k^{n_k}] = \frac{1}{n_k}(n_k\mu) = \mu,$$

where in the last equality we used eq. (5.6). Thus the expectation of the sample mean is μ, which completes the first part of the proof.

Next, we show that the variance of the sample mean is $\frac{\Sigma}{n_k}$. By computing the variance of $\overline{F}_k$ and using the fact that the variance scales by the square of the constant, we obtain:

$$\begin{aligned}\text{Var}[\overline{F}_k] &= \text{Var}[\frac{1}{n_k} \sum_{j=1}^{n_k} F_k^j] \\ &= \frac{1}{n_k^2}\left(\text{Var}[F_k^1] + \ldots + \text{Var}[F_k^{n_k}]\right) + \frac{1}{n_k^2} \sum_{i=1}^{n_k} \sum_{\substack{j=1 \\ j\neq i}}^{n_k} \text{Cov}[F_k^i, F_k^j].\end{aligned}$$

By the independence assumption of $\{F_k^j\}_{j=1}^{n_k}$, the cross terms $\text{Cov}[F_k^i, F_k^j] = 0$ for $i \neq j$. Applying eq. (5.6), we have:

$$\text{Var}[\overline{F}_k] = \frac{1}{n_k^2}\left(\text{Var}[F_k^1] + \ldots + \text{Var}[F_k^{n_k}]\right) = \frac{1}{n_k^2}\left(n_k\Sigma\right) = \frac{\Sigma}{n_k}$$

This well-known result implies that the variance of sample means over subsets of size $n_{k,c}$ reflects the underlying class covariance. In principle, by assigning multiple subsets of $n_{k,c}$ features to a single client and computing the empirical covariance of their means, one could recover Σ_c, since $\Sigma_c = n_{k,c}\text{Var}[\overline{F}_{k,c}]$.

However, in federated learning data are assigned only once to each client, and there are K clients in the federation, each with $n_{k,c}$ features and $n_{i,c} \neq n_{j,c}$ for $i \neq j$. To estimate the population covariance Σ_c, we need an estimator that accounts for the contributions of all K clients. In the following proposition, we propose such an estimator.

Proposition 1 *Let K be the number of clients, each with $n_{k,c}$ features, and let C be the total number of classes. Let $\hat{\mu}_c = \frac{1}{N_c}\sum_{j=1}^{N_c} F^j$ be the unbiased estimator of the population mean μ_c and $N_c = \sum_{k=1}^{K} n_{k,c}$ be the total number of features for a single class. Assuming the features for class c are iid across clients at initialization, the estimator*

$$\hat{\Sigma}_c = \frac{1}{K-1}\sum_{k=1}^{K} n_{k,c}(\overline{F}_{k,c} - \hat{\mu}_c)(\overline{F}_{k,c} - \hat{\mu}_c)^\top \tag{5.7}$$

is an unbiased estimator of the population covariance Σ_c, for all $c \in 1, \ldots, C$.

Proof To prove this, we fix the class c and omit the dependencies on c for clarity. So we write $n_{k,c} = n_k$, $\overline{F}_{k,c} = \overline{F}_k$, $N_c = N$, $\hat{\mu}_c = \hat{\mu}$, $\hat{\Sigma}_c = \hat{\Sigma}$, $\mu_c = \mu$, and $\Sigma_c = \Sigma$. By the definition of an unbiased estimator, we need to show that:

$$\mathbb{E}[\hat{\Sigma}] = \mathbb{E}\left[\frac{1}{K-1}\sum_{k=1}^{K} n_k(\overline{F}_k - \hat{\mu})(\overline{F}_k - \hat{\mu})^\top\right] = \Sigma.$$

By the linearity of the expectation, the definition of sample mean $\overline{F}_k = \frac{1}{n_k}\sum_{j=1}^{n_k} F_k^j$, and the definition of global class mean $\hat{\mu} = \frac{1}{N}\sum_{k=1}^{K}\sum_{j=1}^{n_k} F_k^j$, we have:

$$\mathbb{E}[\hat{\Sigma}] = \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k \mathbb{E}[\overline{F}_k \overline{F}_k^\top] - \sum_{k=1}^{K} n_k \mathbb{E}[\overline{F}_k \hat{\mu}^\top] - \sum_{k=1}^{K} n_k \mathbb{E}[\hat{\mu} \overline{F}_k^\top] + \sum_{k=1}^{K} n_k \mathbb{E}[\hat{\mu}\hat{\mu}^\top]\right)$$

$$= \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k \mathbb{E}[\overline{F}_k \overline{F}_k^\top] - 2\mathbb{E}[(\sum_{k=1}^{K}\sum_{j=1}^{n_k} F_k^j)\hat{\mu}^\top] + \sum_{k=1}^{K} n_k \mathbb{E}[\hat{\mu}\hat{\mu}^\top]\right)$$

$$= \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k \mathbb{E}[\overline{F}_k \overline{F}_k^\top] - 2N\mathbb{E}[\hat{\mu}\hat{\mu}^\top] + \sum_{k=1}^{K} n_k \mathbb{E}[\hat{\mu}\hat{\mu}^\top]\right). \quad (5.8)$$

By applying the variance definition, along with the expectation and variance of the sample mean (see Equation (5.5)), we obtain:

$$\mathbb{E}[\overline{F}_k \overline{F}_k^\top] = \mathrm{Var}[\overline{F}_k] + \mathbb{E}[\overline{F}_k]\mathbb{E}[\overline{F}_k]^\top = \frac{\Sigma}{n_k} + \mu\mu^\top. \quad (5.9)$$

Now, by considering the right term in eq. (5.8), since $\hat{\mu}$ is an unbiased estimator of the population mean, then $\mathbb{E}[\hat{\mu}] = \mu$. Moreover, since we assume that the features for a single class across clients are i.i.d at initialization, we can use re-use the result in Equation (5.5) by considering the all class features as a random sample of size N from a population with mean μ and variance Σ. Consequently, the global sample mean $\hat{\mu}$ is has variance $\mathrm{Var}[\hat{\mu}] = \frac{\Sigma}{N}$. Then

$$\mathbb{E}[\hat{\mu}\hat{\mu}^\top] = \mathrm{Var}[\hat{\mu}] + \mathbb{E}[\hat{\mu}]\mathbb{E}[\hat{\mu}]^\top = \frac{\Sigma}{N} + \mu\mu^\top. \quad (5.10)$$

By using eq. (5.9) and eq. (5.10) in eq. (5.8), and recalling that $N = \sum_{k=1}^{K} n_k$, we obtain:

$$\mathbb{E}[\hat{\Sigma}] = \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k(\frac{\Sigma}{n_k} + \mu\mu^\top) - 2N(\frac{\Sigma}{N} + \mu\mu^\top) + \sum_{k=1}^{K} n_k(\frac{\Sigma}{N} + \mu\mu^\top)\right)$$

$$= \frac{1}{K-1}(K\Sigma + \mu\mu^\top N - 2\Sigma - 2N\mu\mu^\top + (\frac{\Sigma}{N} + \mu\mu^\top)N)$$

$$= \frac{1}{K-1}(K-1)\Sigma = \Sigma.$$

Covariance shrinkage. Shrinkage [39, 174] stabilizes covariance estimation

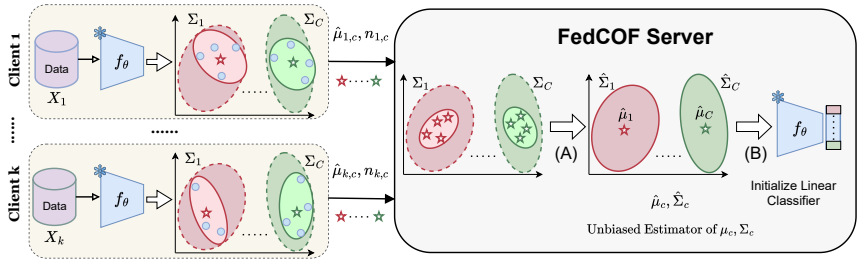


Figure 5.2: **Federated Learning with COvariances for Free (FedCOF).** Each client k communicates only its class means $\hat{\mu}_{k,c}$ and counts $n_{k,c}$. On the server side, (A) we use a provably unbiased estimator $\hat{\Sigma}_c$ (denoted by solid lines) of population covariance Σ_c (denoted by dashed lines) based on the received class means (see section 5.4.2). (B) We initialize the linear classifier using the estimated second-order statistics and remove the between-class scatter matrix as discussed in section 5.4.3.

by adding a scaled identity matrix to the covariance matrices, which regularizes small eigenvalues and reduces estimator variance. This is particularly helpful when the number of samples is smaller than the feature dimensionality, leading to low-rank covariances, and has been recently adopted in continual learning methods leveraging feature covariances [48,109]. In the FL setup, the covariance estimation using a limited number of clients may poorly estimate the population covariance Σ_c. So, we perform shrinkage to better estimate the class covariances from the client means as follows:

$$\hat{\Sigma}_c = \frac{1}{K-1} \sum_{k=1}^{K} n_{k,c} (\hat{\mu}_{k,c} - \hat{\mu}_c)(\hat{\mu}_{k,c} - \hat{\mu}_c)^\top + \gamma I_d, \tag{5.11}$$

where $\hat{\mu}_{k,c} = \overline{F}_{k,c}$ represents a realization of client means and $\gamma > 0$ is the shrinkage factor.

Impact of number of clients. The quality of estimated covariances depends on the number of clients. More clients will give more means and improve the estimate compared to fewer clients. While realistic settings have thousands of clients [59,68], there can be FL settings with fewer. In such cases, we propose to sample multiple means from each client to increase number of means used for covariance estimation. This can be done by randomly sampling subsets of features in each client and computing a mean from each subset. We validate this approach experimentally (see fig. 5.5).

From a theoretical perspective, although the estimator $\hat{\Sigma}_c$ is unbiased in eq. (5.7), this only ensures that its expected value equals the true covariance Σ_c – not that any individual estimate is accurate. Its variance still depends on the number of independent means K sampled – that is the number of clients in the federation K which contains class c.

Therefore, increasing K leads to a tighter concentration of $\hat{\Sigma}_c$ around its expectation, reducing the overall mean square error (MSE) – the sum of variance and squared bias (which remains zero since the estimator is unbiased) – between the estimator and the true covariance matrix. Moreover, where the number of clients is small relative to the feature dimension d, the estimate may be ill-conditioned. The shrinkage term γI_d in eq. (5.11) improves numerical stability in these settings, trading a small amount of bias for a significant reduction in variance – even when only a few clients are available.

Sampling strategy. We propose using a simple multiple mean sampling strategy following sampling without replacement. We consider the number of means $\mathcal{M}$ to sample as a fixed number which is a hyperparameter. For each class, we take disjoint random sets from $n_{k,c}$ samples in a client and compute the mean for these subsets. We take disjoint sets to avoid computing similar sample means. The only condition we enforce is that clients use atleast 2 samples to compute a mean. If a client does not have atleast $2\mathcal{M}$ samples, we send less than $\mathcal{M}$ sample means. For instance, if a client has 3 samples, we compute and share a single mean.

5.4.3 Classifier Initialization with Estimated Covariances

Having derived how to compute class covariances from client means, we now show how FedCOF leverages the estimated covariances to initialize classifier weights.

Proposition 2 *Let $F \in \mathbb{R}^{d\times N}$ be a feature matrix with empirical global mean $\hat{\mu}_g \in \mathbb{R}^d$, and $Y \in \mathbb{R}^{N\times C}$ be a label matrix. The optimal ridge regression solution $W^* = (G + \lambda I_d)^{-1}B$, where $B \in \mathbb{R}^{d\times C}$ and $G \in \mathbb{R}^{d\times d}$ can be written in terms of class means and covariances as follows:*

$$B = [\hat{\mu}_c N_c]_{c=1}^{C}\,, \tag{5.12}$$

$$G = \sum_{c=1}^{C}(N_c - 1)\hat{S}_c + \sum_{c=1}^{C} N_c(\hat{\mu}_c - \hat{\mu}_g)(\hat{\mu}_c - \hat{\mu}_g)^\top + N\hat{\mu}_g\hat{\mu}_g^\top \tag{5.13}$$

where the first two terms $\sum_{c=1}^{C}(N_c-1)\hat{S}_c$ *and* $\sum_{c=1}^{C} N_c(\hat{\mu}_c-\hat{\mu}_g)(\hat{\mu}_c-\hat{\mu}_g)^\top$ *represents the within-class and between class scatter respectively, while* $\hat{\mu}_c$, $\hat{S}_c$ *and* N_c, *denote the empirical mean, covariance and sample size for class* c, *respectively.*

Proof The first part, regarding eq. (5.12), follows directly. From the ridge regression solution, $B = FY$, which is obtained by summing the features for each class and arranging them into the columns of a matrix. This results in the product of class means and samples per class.

Now, for computing the matrix G, we proceed with the definition of the global sample covariance:

$$\hat{S} = \frac{1}{N-1}(F-\overline{F})(F-\overline{F})^\top = \frac{1}{N-1}\left(FF^\top - F\overline{F}^\top - \overline{F}F^\top + \overline{F}\,\overline{F}^\top\right),$$

where $\overline{F} = \left(\frac{1}{N}\sum_{j=1}^{N} F^j\right)\mathbf{1}^\top = \hat{\mu}_g\mathbf{1}^\top \in \mathbb{R}^{d\times N}$ is the matrix obtained by replicating the global mean N times in each column and $\mathbf{1}\in\mathbb{R}^{N\times 1}$ is a column vector of ones. Recalling that $G = FF^\top$, we have:

$$\hat{S} = \frac{1}{N-1}(G - F\mathbf{1}\hat{\mu}_g^\top - \hat{\mu}_g\mathbf{1}^\top F^\top + \hat{\mu}_g\mathbf{1}^\top\mathbf{1}\hat{\mu}_g^\top) = \frac{1}{N-1}(G - 2F\mathbf{1}\hat{\mu}_g^\top + N\hat{\mu}_g\hat{\mu}_g^\top)$$

since $F\mathbf{1}\hat{\mu}_g^\top = \hat{\mu}_g\mathbf{1}^T F^\top$ and $\mathbf{1}^T\mathbf{1} = N$.

Now, since $F\mathbf{1} = \sum_{j=1}^{N} F^j$, we can obtain the matrix G as:

$$\begin{aligned} G &= (N-1)\hat{S} + 2\left(\sum_{j=1}^{N} F^j\right)\hat{\mu}_g^\top - N\hat{\mu}_g\hat{\mu}_g^\top \\ &= (N-1)\hat{S} + 2N\hat{\mu}_g\hat{\mu}_g^\top - N\hat{\mu}_g\hat{\mu}_g^\top \\ &= (N-1)\hat{S} + N\hat{\mu}_g\hat{\mu}_g^\top \end{aligned} \tag{5.14}$$

It is a well known result that the global covariance can be expressed as:

$$\hat{S} = \frac{1}{N-1}\left(\sum_{c=1}^{C}(N_c-1)\hat{\Sigma}_c + \sum_{c=1}^{C} N_c(\hat{\mu}_c-\hat{\mu}_g)(\hat{\mu}_c-\hat{\mu}_g)^T\right),$$

Replacing the global covariance $\hat{S}$ in eq. (5.14), we obtain the final expression

for G as:

$$G = \sum_{c=1}^{C}(N_c - 1)\hat{S}_c + \sum_{c=1}^{C} N_c(\hat{\mu}_c - \hat{\mu}_g)(\hat{\mu}_c - \hat{\mu}_g)^\top + N\hat{\mu}_g\hat{\mu}_g^\top$$

Table 5.2: Comparison of classifiers across datasets with and without G_{btw} in the centralized setting

Classifier	G_{with}	G_{btw}	Dataset	Train Acc	Test Acc
Ridge Regression	✓	✓	CIFAR-100	60.0	57.1
Proposed Classifier	✓	✗	CIFAR-100	60.4	**57.3**
Ridge Regression	✓	✓	Imagenet-R	52.8	37.6
Proposed Classifier	✓	✗	Imagenet-R	53.4	**38.6**
Ridge Regression	✓	✓	CUB200	92.0	50.4
Proposed Classifier	✓	✗	CUB200	91.3	**53.7**
Ridge Regression	✓	✓	Cars	85.9	41.4
Proposed Classifier	✓	✗	Cars	86.2	**44.8**

Table 5.3: Performance comparison of different FL setups using G_{btw} and G_{with} across datasets.

FL Setup	CIFAR-100	ImageNet-R	CUB200	CARS
Using G_{btw}	52.8	34.8	49.7	33.7
Using G_{btw} + G_{with}	56.3	36.8	51.6	42.4
Using G_{with}	56.3	37.2	53.5	44.6

To analyze the impact of the two scatter matrices, we first consider the centralized setting in table 5.2 and empirically find that using only within-class scatter matrix performs better than using total scatter matrix in eq. (5.13). We then analyze their spectral properties via the condition number, defined for a matrix G as $\mathcal{K}(G) = \lambda_{\max}(G)/\lambda^+_{\min}(G)$, where $\lambda_{\max}(G)$ and $\lambda^+_{\min}(G)$ are the largest and smallest non-zero eigenvalue, respectively. For a SqueezeNet backbone in the centralized setting, we observe that G_{btw} is highly ill-conditioned, with condition numbers $\mathcal{K}(G_{btw})$ of 3.0×10^7, 2.5×10^7, 2.2×10^7, 1.3×10^7 on CUB, Cars, ImageNet-R and CIFAR-100, respectively, while G_{with} is much better conditioned ($\mathcal{K}(G_{with})$: 4.5×10^3, 2.5×10^4, 8.2×10^2, 6.3×10^3). Including G_{btw} can cause numerical instability due to its poor conditioning, leading the classifier to overfit directions with small eigenvalues that capture noise or dataset-specific artifacts. We also observe in table 5.2 that, while both methods achieve similarly high training accuracies, Ridge Regression consistently under-

Algorithm 5 FedCOF: Federated Learning with COvariances for Free

Client-Side (Client k):

Input: C: set of all classes, f_θ: pre-trained model, $X_{k,c}$: samples of class c in client k, $n_{k,c}$: number of samples in $X_{k,c}$

for $c = 1$ to C **do**

$\hat{\mu}_{k,c} = \frac{1}{n_{k,c}} \sum_{x \in X_{k,c}} f_\theta(x)$

end for

Send the class means $\hat{\mu}_{k,c}$ and sample counts $n_{k,c}$ to the Server

Server-Side:

Input: $\hat{\mu}_{k,c}, n_{k,c}$ sent from K clients, $\gamma > 0$

for $c = 1 \dots C$ **do**

$\hat{\mu}_c = \frac{1}{N_c} \sum_{k=1}^{K} n_{k,c} \hat{\mu}_{k,c}$; $N_c = \sum_{k=1}^{K} n_{k,c}$

$\hat{\Sigma}_c = \frac{1}{K-1} \sum_{k=1}^{K} n_{k,c} (\hat{\mu}_{k,c} - \hat{\mu}_c)(\hat{\mu}_{k,c} - \hat{\mu}_c)^\top + \gamma I_d$, (eq. (5.11))

end for

$\hat{\mu}_g = \frac{1}{N} \sum_{c=1}^{C} N_c \hat{\mu}_c, \qquad N = \sum_{c=1}^{C} N_c$

$B = [\hat{\mu}_c N_c]_{c=1}^{C}$,

$\hat{G} = \sum_{c=1}^{C} (N_c - 1)\hat{\Sigma}_c + N \hat{\mu}_g \hat{\mu}_g^\top$

$W^* = \hat{G}^{-1} B$, (eq. (5.15))

Normalize W^*: $W_c^* \leftarrow W_c^* / \|W_c^*\|; c = 1, \dots, C$

performs on the test set. This suggests that incorporating G_{btw} introduces an overfitting effect, as the classifier learns directions that do not transfer well to unseen samples.

Finally, we also empirically analyze in table 5.3 the impact of removing the between-class covariances in a Federated scenario using SqueezeNet. Here in the FL setup we again clearly see the negative effect of incorporating between-class scatter statistics.

As a result, we propose to remove the between-class scatter from eq. (5.13) and initialize the linear classifier at the end of the pre-trained network using the within-class covariances $\hat{\Sigma}_c$ which are estimated from client means using eq. (5.11), as follows:

$$W^* = \hat{G}^{-1} B; \qquad \hat{G} = \sum_{c=1}^{C} (N_c - 1)\hat{\Sigma}_c + N \hat{\mu}_g \hat{\mu}_g^\top. \tag{5.15}$$

Theoretically, we observe that a similar approach is used in Linear Discriminant Analysis [41], which employs only within-class covariances for finding optimal weights. By removing between-class scatter, we propose a more effective classifier initialization than Fed3R (which uses G from eq. (5.13) and considers both within- and between-class scatter matrices).

We summarize in algorithm 5 how we estimate the covariance matrix for each class using only the client means and use the estimated covariances to initialize the classifier as in eq. (5.15). The normalization of the weights accounts for class imbalance in the entire dataset.

Impact of the iid assumption. FedCOF initializes the classifier with the class covariance estimator (see eq. (5.11)), unbiased only under the assumption of *iid pre-trained features per class* (section 5.4.2). In FL each client has its own data, typically distributed in a statistically heterogeneous or class-imbalanced manner according to a Dirichlet distribution [58]. As a result, each client has data from different set of classes, resulting in non-iid data distributions across clients. However, note that the samples belonging to the same class in different clients are sampled from the same distribution. We exploit this fact in FedCOF. We later show empirically that our method can be successfully applied to non-iid FL scenarios involving thousands of heterogeneous clients on iNaturalist-Users-120K [59]. We hypothesize that this can be attributed to the strong generalization capabilities of pre-trained models.

We now analyze the bias of the estimator under non-iid assumptions for the same class and theoretically quantify the bias when the i.i.d assumption is violated. Under the i.i.d. assumption, the features from the same class assigned to clients can be treated as random samples from the *same* population distribution with mean μ_c and covariance Σ_c. For simplicity, focusing on a single class and dropping the class subscript c, the population distribution has mean μ and covariance Σ. As a result, recalling eq. (5.9), we can write:

$$\mathbb{E}[\overline{F}_k\overline{F}_k^\top] = \mathrm{Var}[\overline{F}_k] + \mathbb{E}[\overline{F}_k]\mathbb{E}[\overline{F}_k]^\top = \frac{\Sigma}{n_k} + \mu\mu^\top,$$

where n_k is the number of samples assigned to client k, and $\overline{F}_k$ is the sample mean for client k

Now, if the *i.i.d assumption is violated* the local features assigned to each client can be viewed as random samples drawn from different client population distributions, each characterized by a mean μ_k and covariance Σ_k, with $\mu_i \neq \mu_j$ and $\Sigma_i \neq \Sigma_j$ for $i \neq j$, and $i, j = 1, \ldots, K$. In this case:

$$\mathbb{E}[\overline{F}_k\overline{F}_k^\top] = \mathrm{Var}[\overline{F}_k] + \mathbb{E}[\overline{F}_k]\mathbb{E}[\overline{F}_k]^\top = \frac{\Sigma_k}{n_k} + \mu_k\mu_k^\top. \tag{5.16}$$

To compute the expectation of the estimator $\mathbb{E}[\hat{\Sigma}]$, we follow the same procedure used to prove proposition in eq. (5.7) up to eq. (5.8):

$$\mathbb{E}[\hat{\Sigma}] = \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k\mathbb{E}[\overline{F}_k\overline{F}_k^\top] - 2N\mathbb{E}[\hat{\mu}\hat{\mu}^\top] + \sum_{k=1}^{K} n_k\mathbb{E}[\hat{\mu}\hat{\mu}^\top]\right). \tag{5.17}$$

Assuming the global feature dataset, regardless of client assignment, is a random

sample from the population with mean μ and covariance Σ, we can write:

$$\mathbb{E}[\hat{\mu}\hat{\mu}^\top] = \mathrm{Var}[\hat{\mu}] + \mathbb{E}[\hat{\mu}]\mathbb{E}[\hat{\mu}]^\top = \frac{\Sigma}{N} + \mu\mu^\top. \tag{5.18}$$

Substituting eq. (5.18) and eq. (5.16) into eq. (5.17), and recalling that $N = \sum_{k=1}^{K} n_k$, we obtain:

$$\begin{aligned}
\mathbb{E}[\hat{\Sigma}] &= \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k(\frac{\Sigma_k}{n_k} + \mu_k\mu_k^\top) - 2N(\frac{\Sigma}{N} + \mu\mu^\top) + \sum_{k=1}^{K} n_k(\frac{\Sigma}{N} + \mu\mu^\top)\right) \\
&= \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k(\frac{\Sigma_k}{n_k} + \mu_k\mu_k^\top) - \Sigma - N\mu\mu^\top\right) \\
&= \frac{1}{K-1}\sum_{k=1}^{K}(\Sigma_k - \frac{\Sigma}{K}) + \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k\mu_k\mu_k^\top - \sum_{k=1}^{K} n_k\mu\mu^\top\right) \\
&= \frac{1}{K-1}\sum_{k=1}^{K}(\Sigma_k - \frac{\Sigma}{K}) + \frac{1}{K-1}\sum_{k=1}^{K} n_k(\mu_k\mu_k^\top - \mu\mu^\top) \\
&= \frac{1}{K-1}\sum_{k=1}^{K}(\Sigma_k - \frac{\Sigma}{K}) + \frac{1}{K-1}\sum_{k=1}^{K} n_k(\mu_k - \mu)(\mu_k - \mu)^\top,
\end{aligned}$$

where in the last step we used that $\sum_{k=1}^{K} n_k\mu_k = N\mu$.

The bias of the estimator is thus given by:

$$\mathrm{Bias}(\hat{\Sigma}) = \mathbb{E}[\hat{\Sigma}] - \Sigma = \frac{1}{K-1}\sum_{k=1}^{K}(\Sigma_k - \Sigma) + \frac{1}{K-1}\left(\sum_{k=1}^{K} n_k(\mu_k - \mu)(\mu_k - \mu)^\top\right). \tag{5.19}$$

Note that if each client population covariance Σ_k is equal to the global population covariance Σ, and the mean of each client μ_k is equal to the population mean, then the bias is zero (i.e., the estimator is unbiased). However, the bias formula reveals that when the distribution of a class within a client differs from the global distribution of the same class, our estimator introduces a systematic bias. This situation can arise in the *feature-shift* setting, in which each client is characterized by a different domain. We later evaluate FedCOF under the feature-shift setting [93] to quantify how this bias affects performance in table 5.7.

FedCOF in multiple rounds. While the proposed estimator requires class means from all clients in a single round, this might not be realistic in settings in which clients appear in successive rounds based on availability. For multi-round classifier initialization (see FedCOF in fig. 5.3 before fine-tuning), the server uses all class means and counts received from all clients seen up to the current round and stores the accumulated means and counts for future rounds. As a result, FedCOF uses statistics from all clients seen up to the current round, similar to Fed3R. This can be easily implemented by ensuring on the client side that each client transfers statistics to the server only once, avoiding repeated transfer of statistics. FedCOF converges when all clients are seen at least once, with total communication cost equal to the single round-case (see section 5.5.4 for details on communication cost).

Privacy aspects. FedCOF requires each client to send its class means and frequencies (see eq. (5.7)), similar to CCVR [106]. While this avoids sharing feature-level data, it does not guarantee strong privacy and may introduce concerns, particularly because it complicates the secure aggregation protocol applicable in Fed3R. Nonetheless, as discussed in section 5.6, FedCOF can be combined with additional privacy-preserving mechanisms, such as adding noise to the shared statistics or adapting the method to support secure aggregation. The latter preserves FedCOF's performance but increases communication overhead to the level of Fed3R.

5.5 Experiments

5.5.1 Experimental Setup

Datasets. We use the following datasets for the main experiments in this chapter:

- **CIFAR-100** [81] has 100 classes provided in 50k training and 10k testing images.
- **ImageNet-R (IN-R)** is composed of 30k images covering 200 ImageNet classes. ImageNet-R [54] is an out-of-distribution dataset and proposed to evaluate out-of-distribution generalization using ImageNet pre-trained weights. It contains data with multiple styles like cartoon, graffiti and origami which is not seen during pre-training.
- **CUB200** [178] is a fine-grained dataset and has 200 classes of different bird species provided in 5994 training and 5794 testing images.

- **Stanford Cars** [80] has 196 classes of cars with 8144 training images and 8041 test images.

- **iNaturalist-Users-120k** [58] is a real-world, large-scale dataset [173] proposed by [58] for federated learning and contains 120k training images of natural species taken by citizen scientists around the world, belonging to 1203 classes spread across 9275 clients.

In datasets like ImageNet-R and CARS, we also face class-imbalanced situations where there is a significant class-imbalance at the global level. We distribute the first 4 datasets to 100 clients using a highly heterogeneous Dirichlet distribution ($\alpha = 0.1$) following standard practice [58, 88].

Implementation details. We use three models: namely SqueezeNet [60] following [88] and [125], MobileNetV2 [148] following [38,59], and ViT-B/16 [34]. All models are pre-trained on ImageNet-1k [28]. We use the FLSim library. We use $\gamma = 1$ for all experiments with SqueezeNet and ViT-B/16, and $\gamma = 0.1$ for all experiments with MobileNetV2 due to very high dimensionality d of the feature space. We compare to *FedCOF Oracle* in which real class covariances are shared from clients and aggregated in server instead of using our estimated covariances. For all experiments, we set the client participation in each round to 30%, and we show the training-free methods in multiple rounds in figs. 5.3 and 5.4. We discuss computation of communication costs for all methods in section 5.5.4.

Here, we provide details on learning rate (lr) used for all fine-tuning experiments with FedAdam. For ImageNet-R and Stanford Cars, we use a lr of 0.0001 for both server and clients for FedNCM, Fed3R and FedCOF initializations. For CUB200, we use a server lr of 0.00001 and client lr of 0.00005 for Fed3R and FedCOF, while for FedNCM, we use a higher lr of 0.0001 for clients. For random classifier initialization with all datasets, we use a higher lr of 0.001 for clients and lr of 0.0001 for server. We use 1 local epoch, for all fine-tuning experiments on 4 datasets. After training-free classifier initialization, we fine-tune the models for 100 rounds. When starting from random classifier initialization, we train more for 200 rounds. When training with FedAvg and random classifier initialization, we use a client lr of 0.005 for all datasets other than inat-120K.

For the linear probing (LP) experiments for the 4 datasets other than iNat-120K, with FedAvg we train for 200 rounds with 1 local epoch and use a client lr of 0.01 and server lr of 1.0 for FedNCM. For Fed3R and FedCOF initializations, we use a client lr of 0.001 and a server lr of 1.0. For LP experiments on iNat-120K, we use 3 local epochs, 30% client participation and train for 5000 rounds. For iNat-120K, we use a client lr of 0.001 for FedAvg-LP without

Table 5.4: Evaluation of different *training-free methods* using 100 clients for first four datasets and 9275 pre-defined clients on iNat-120K across 5 random seeds. We show the total communication cost (in MB) from all clients to server. We also show the FedCOF oracle in which full class covariances are shared from clients to server. Best results from each section in **bold**.

		SqueezeNet ($d = 512$)		MobileNetv2 ($d = 1280$)		ViT-B/16 ($d = 768$)	
	Method	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)
CIFAR-100	FedNCM [88]	41.5±0.1	**5.9**	55.6±0.1	**14.8**	55.2±0.1	**8.9**
	Fed3R [38]	**56.9**±0.1	110.2	62.7±0.1	670.1	**73.9**±0.1	244.8
	FedCOF	56.1±0.2	**5.9**	**63.5**±0.1	**14.8**	73.2±0.1	**8.9**
	FedCOF Oracle (Full Covs)	56.4±0.1	3015.3	63.9±0.1	18823.5	73.8±0.1	6780.0
IN-R	FedNCM [88]	23.8±0.1	**7.1**	37.6±0.2	**17.8**	32.3±0.1	**10.7**
	Fed3R [38]	37.6±0.2	111.9	46.0±0.3	673.1	**51.9**±0.2	246.6
	FedCOF	**37.8**±0.4	**7.1**	**47.4**±0.1	**17.8**	51.8±0.3	**10.7**
	FedCOF Oracle (Full Covs)	38.2±0.1	3645.7	48.0±0.3	22758.8	52.7±0.1	8197.4
CUB200	FedNCM [88]	37.8±0.3	**4.8**	58.3±0.3	**12.0**	75.7±0.1	**7.2**
	Fed3R [38]	50.4±0.3	109.6	58.6±0.2	667.3	77.7±0.1	243.1
	FedCOF	**53.7**±0.3	**4.8**	**62.5**±0.4	**12.0**	**79.4**±0.2	**7.2**
	FedCOF Oracle (Full Covs)	54.4±0.1	2472.1	63.1±0.5	15432.7	79.6±0.2	5558.6
Cars	FedNCM [88]	19.8±0.2	**5.4**	30.0±0.1	**13.5**	26.2±0.4	**8.1**
	Fed3R [38]	39.9±0.2	110.2	41.6±0.1	668.8	47.9±0.3	244.0
	FedCOF	**44.0**±0.3	**5.4**	**47.3**±0.5	**13.5**	**52.5**±0.3	**8.1**
	FedCOF Oracle (Full Covs)	44.6±0.1	2767.3	47.2±0.3	17275.7	53.1±0.1	6222.5
iNat-120K	FedNCM [88]	21.2±0.1	**111.8**	36.0±0.1	**279.5**	53.9±0.1	**167.7**
	Fed3R [38]	32.1±0.1	9837.3	41.5±0.1	61064.1	62.5±0.1	22050.2
	FedCOF	**32.5**±0.1	**111.8**	**44.1**±0.1	**279.5**	**63.1**±0.1	**167.7**
	FedCOF Oracle (Full Covs)	32.4±0.1	57k	43.6±0.1	358k	62.9±0.1	128k

classifier initialization, a client lr of 0.0005 for FedNCM and client lr of 0.00001 for Fed3R and FedCOF.

The FedCOF Oracle (Sharing Full Covariances). Similar to CCVR [106], we aggregate the class covariances from clients as follows:

$$\hat{\Sigma}_c = \sum_{k=1}^{K} \frac{n_{k,c}-1}{N_c-1}\hat{\Sigma}_{k,c} + \sum_{k=1}^{K} \frac{n_{k,c}}{N_c-1}\hat{\mu}_{k,c}\hat{\mu}_{k,c}^T - \frac{N_c}{N_c-1}\hat{\mu}_c\hat{\mu}_c^T. \tag{5.20}$$

For the oracle setting of FedCOF, we use the aggregated class covariance from eq. (5.20) and apply shrinkage to obtain $\hat{\Sigma}_c + \gamma I_d$ and use it in eq. (5.15) instead of using our estimated covariances.

5.5.2 Experimental Results

We compare the performance of existing training-free methods and the proposed method in table 5.4 using pre-trained Squeezenet, Mobilenetv2 and ViT-B/16 models. We observe that Fed3R [38] using second-order statistics

outperforms FedNCM [88] significantly ranging from 0.3% to 21% across all datasets. However, Fed3R requires a higher communication cost compared to FedNCM. In real-world iNat-120K benchmark, Fed3R needs $61k$ MB compared to 280 MB for FedNCM (see fig. 5.1), which is 218 times higher. FedCOF performs better than Fed3R in most settings despite having the same communication cost as FedNCM. FedCOF achieves similar performance as the oracle setting using aggregated class covariances requiring very high communication, which validates the effectiveness of the proposed covariance estimator.

FedCOF maintains similar accuracy with Fed3R on CIFAR-100 and ImageNet-R, with an improvement of about 1% when using MobileNetv2. FedCOF outperforms Fed3R on CUB200 and Cars. On CUB200, FedCOF outperforms Fed3R by 3.3%, 3.9% and 2.2% using SqueezeNet, MobileNetv2 and ViT-B/16 respectively. FedCOF improves over Fed3R in the range of 4.1% to 5.7% on Cars. On iNat-120K, FedCOF improves over Fed3R by 0.4%, 2.6% and 0.6% using different models. When comparing FedCOF with FedNCM – both with equal communication costs and same strategy in clients – one can observe that the usage of second order statistics derived only from the class means of clients leads to large performance gains, e.g. 24% using SqueezeNet and 26% using ViT-B/16 on Cars, above 8% using all architectures on large-scale iNat-120K.

Comparison with communication-efficient methods. We consider the recent Probabilistic Federated Prompt-Tuning (PFPT) approach that aggregates trainable prompt parameters from clients and tunes them in a federated manner while keeping the backbone fixed [189]. We compare PFPT, FedAvg-PT, and FedProx-PT (prompt-tuning variants of FedAvg and FedProx) with the proposed FedCOF. Similar to PFPT [189], we use a pre-trained ViT-B/32, assign class samples to clients using a Dirichlet distribution ($\alpha = 0.1$), and use 3 random seeds. We use the same training hyperparameters as PFPT. We show in table 5.5 that FedCOF is much more communication efficient compared to PFPT and achieves higher performance without any training. We also observe that prompt-tuning methods perform poorly on fine-grained datasets. We discuss more on this in section 5.5.4.

Table 5.5: Comparison of FedCOF with federated prompt-tuning methods using ViT-B/32.

	CIFAR-100		IN-R		CUB200		Cars	
Method	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)
FedAvg-PT [189]	74.5±0.5	884.7	47.6±1.3	1622.0	37.0±2.0	1622.0	13.6±1.7	1592.5
FedProx-PT [189]	73.6±0.4	884.7	47.9±0.5	1622.0	38.5±0.8	1622.0	13.7±1.5	1592.5
PFPT [189]	75.1±0.5	846.5	50.7±0.2	1794.4	38.6±0.9	1765.5	12.9±1.1	1736.1
FedCOF	**75.3**±0.1	**8.9**	**54.9**±0.2	**10.7**	**65.0**±0.1	**7.2**	**50.4**±0.1	**8.1**

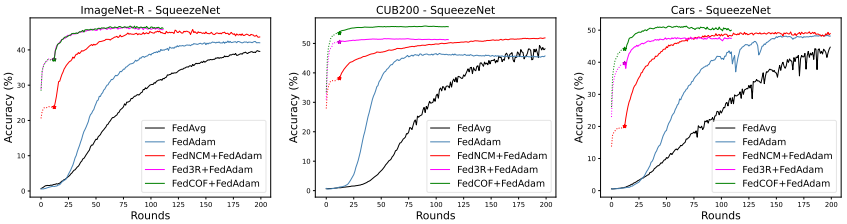


Figure 5.3: Performance comparison when initializing with different methods and fine-tuning with FedAdam [139] and FedAvg [115]. We also compare with FedAdam and FedAvg using a pre-trained backbone and random classifier initialization. The training-free initialization stages are shown as dotted lines and stars represents the start of fine-tuning stages. Accuracies are averaged over 3 seeds.

Table 5.6: Comparison with training-based FL baselines (FedAvg and FedAdam) using pre-trained SqueezeNet. For training-based methods, we consider 100 rounds of training for fair comparison and report accuracy of 3 random seeds.

Method	Training	ImageNet-R	CUB200	Cars
FedAvg	✓	30.0±0.6	30.3±6.7	24.9±1.6
FedAdam	✓	38.8±0.6	46.4±0.8	41.8±0.6
FedNCM	✗	23.8±0.1	37.8±0.3	19.8±0.2
Fed3R	✗	37.6±0.2	50.4±0.3	39.9±0.2
FedCOF	✗	37.8±0.4	53.7±0.3	44.0±0.3
FedNCM+FedAdam	✓	44.7±0.1	50.2±0.2	48.7±0.2
Fed3R+FedAdam	✓	45.9±0.3	51.2±0.3	47.4±0.4
FedCOF+FedAdam	✓	**46.0**±0.4	**55.7**±0.4	**49.6**±0.6

Comparison with full fine-tuning methods. We compare training-free methods with FL baselines like FedAvg and FedAdam with randomly initialized classifier and pre-trained backbone in table 5.6. We use adaptive optimizer, FedAdam [139] since it performs better than most other optimizers as shown in [125]. Without any training, FedCOF outperforms FedAvg in all settings and FedAdam by 7.3% on CUB200 and 2.2% on Cars, and achieves competitive performance in ImageNet-R. We show in fig. 5.3 how FedCOF starts from a very high accuracy compared to FedAdam and further improves on fine-tuning.

Analysis of fine-tuning and linear probing. While FedCOF achieves high accuracy without training, we show in fig. 5.3 that further fine-tuning the model with FL achieves better and faster convergence compared to federated optimization from scratch. We show the performance of fine-tuning after

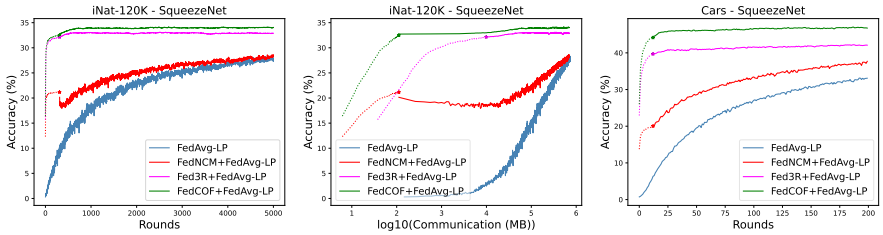


Figure 5.4: Performance of different classifier initialization methods when linear-probing with FedAvg [115]. FedAvg-LP (in blue) uses random classifier initialization and a pre-trained backbone. The training-free initialization stage is shown in dotted lines, stars represents the start of linear probing.

training-free classifier initialization in fig. 5.3. These training-free methods end after all clients appear at least once to share their local statistics. We fine-tune the models after FedCOF and Fed3R for 100 rounds since they achieve fast convergence, while we train for 200 rounds for FedAdam, FedAvg, and fine-tuning after FedNCM which takes longer to converge. Fine-tuning after FedCOF starts at a higher accuracy and converges faster compared to FedNCM. Although FedCOF and Fed3R converge similarly on ImageNet-R, FedCOF+FedAdam achieves better accuracy than Fed3R+FedAdam on CUB200 and Cars. We see in table 5.6 that all training-free approaches followed by fine-tuning outperform FedAdam and FedAvg with random classifier initialization.

Following [88] and [125], we perform federated linear probing (LP) of the models using FedAvg after classifier initialization with training-free methods. In FedAvg-LP, we perform FedAvg and learn only the classifier weights of all client models. Linear probing requires much less computation compared to fine-tuning the entire model and were found to be effective with pre-trained models. We observe in fig. 5.4 that linear probing after FedCOF improves significantly compared to FedNCM and Fed3R using ViT-B/16 on Cars and SqueezeNet on iNat-120K. On the real-world dataset iNat-120K, FedAvg-LP with random classifier initialization achieves 27.3% after 5000 rounds while FedCOF+FedAvg-Lp achieves 34% in less than 1000 rounds. We plot accuracy versus communication in fig. 5.4 (center) to demonstrate the advantage of FedCOF over other methods.

Experiments on Feature Shift Settings. Following [93], we perform experiments with MobileNetv2 in a non-iid feature shift setting on the DomainNet [133] dataset. DomainNet contains data from six different domains: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch. We use the top 10

Table 5.7: Comparison of different training-free methods using MobileNetV2 on the feature shift setting on DomainNet. We show the total communication cost (in MB) from all clients to server.

Method	Acc (↑)	Comm. (↓)
FedNCM	65.8	0.3
Fed3R	81.9	39.6
FedCOF	74.1	0.3
FedCOF (2 class means per client)	76.5	0.6
FedCOF (10 class means per client)	78.8	3.1

Table 5.8: Comparison of FedCOF with CCVR across datasets and models.

		SqueezeNet (d=512)		MobileNetv2 (d=1280)		ViT-B/16 (d=768)	
Dataset	Method	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)
CIFAR-100	CCVR	**57.5**±0.2	3015.3	59.6±0.2	18823.5	72.3±0.2	6780.0
	FedCOF	56.1±0.2	**5.9**	**63.5**±0.1	**14.8**	**73.2**±0.1	**8.9**
IN-R	CCVR	36.4±0.2	3645.7	41.9±0.2	22758.8	49.3±0.2	8197.4
	FedCOF	**37.8**±0.4	**7.1**	**47.4**±0.1	**17.8**	**51.8**±0.3	**10.7**
CUB200	CCVR	51.2±0.1	2472.1	61.6±0.2	15432.7	78.7±0.4	5558.6
	FedCOF	**53.7**±0.3	**4.8**	**62.5**±0.4	**12.0**	**79.4**±0.2	**7.2**
Cars	CCVR	40.9±0.4	2767.3	36.0±0.4	17275.7	49.4±0.4	6222.5
	FedCOF	**44.0**±0.3	**5.4**	**47.3**±0.5	**13.5**	**52.5**±0.3	**8.1**

most common classes of DomainNet for our experiments following the setting proposed by [93]. We consider six clients where each client has i.i.d. data from one of the six domains. As a result, different clients have data from different feature distributions. We show in table 5.7 how training-free methods perform in feature shift settings and the accuracy to communication trade-offs.

Fed3R achieves better overall performance then FedCOF, likely due to its use of exact class covariance, avoiding the bias that FedCOF introduces. However, FedCOF achieves comparable results while significantly reducing communication costs. FedNCM perform worse than FedCOF at the same communication budget. When we increase the number of means sampled from each client, the performance of our approach improves. This is due to the fact that our method suffers with low number of clients (only 6 in this experiments) and sampling multiple means helps.

Adapting CCVR for Classifier Initialization. Since CCVR [106] was originally proposed to calibrate classifiers after training, we adapt it to our setting and use it as an initialization method. CCVR is similar to the FedCOF Oracle in terms of communication cost as it shares class means and covariances from all clients to the server. CCVR aggregates the class covariances and means

Table 5.9: Comparison of different training-free methods using SqueezeNet with training-based Fed-LP (federated linear probing with FedAvg [115] starting with pre-trained model and random classifier initialization) across 5 random seeds. FedNCM, Fed3R and the proposed FedCOF does not involve any training. We show the total communication cost (in MB) from all clients to server. The best results from each section are highlighted in **bold**.

	CIFAR-100		ImageNet-R		CUB200		CARS		iNat-120K	
Method	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)	Acc (↑)	Comm. (↓)
Fed-LP	**59.9**±0.2	2458	**37.8**±0.3	4916	46.8±0.8	4916	33.1±0.1	4817	28.0±0.6	1.6×10^6
FedNCM	41.5±0.1	**5.9**	23.8±0.1	**7.1**	37.8±0.3	**4.8**	19.8±0.2	**5.4**	21.2±0.1	**111.8**
Fed3R	56.9±0.1	110.2	37.6±0.2	111.9	50.4±0.3	109.6	39.9±0.2	110.2	32.1±0.1	9837.3
FedCOF	56.1±0.2	**5.9**	**37.8**±0.4	**7.1**	**53.7**±0.3	**4.8**	**44.0**±0.3	**5.4**	**32.5**±0.1	**111.8**

to obtain a global class distribution at the server, and then trains a linear classifier on Gaussian features sampled from aggregated class distributions from clients. We show in table 5.8 that the proposed FedCOF outperforms CCVR in most settings despite having significantly lower communication cost.

Comparison of training-free methods with linear probing. We also compare with our approach with the training-based federated linear probing without any initialization (where we perform FedAvg and learn only the classifier weights of models) and show in table 5.9 that FedCOF is more robust and communication-efficient compared to federated linear probing across several datasets. We follow the same settings as in table 5.4. For first 4 datasets, we perform federated linear probing for 200 rounds with 30 clients per round using FedAvg with a client learning rate of 0.01. For iNat-120k, we train more for 5000 rounds.

5.5.3 Ablation Studies

Impact of number of clients and data heterogeneity. We analyze in fig. 5.5 (left), the performance of FedCOF with varying number of clients and data heterogeneity. We observe that the performance of FedCOF improves with increasing number of clients and decreasing heterogeneity. This is due to the fact that more clients provides more class means and more uniform data distribution gives better representative local means. While more clients are favourable for FedCOF, it still performs well and outperforms FedNCM significantly in the setting with 10 clients and high data heterogeneity.

Multiple class means per client. We analyze FL settings with fewer clients ranging from 10 to 50 in fig. 5.5 (center) and show that sharing multiple class means from each client improves the accuracy. Using only 10 clients,

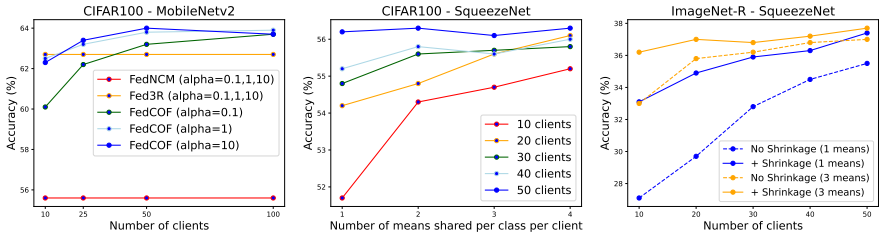


Figure 5.5: Ablation experiments: (left) Change in performance with varying number of clients and data heterogeneity. (center) Sharing multiple class means per client improves FedCOF performance. (right) Impact of shrinkage with varying number of clients and sampled means per client.

sharing 2 class means per client improves the accuracy by 2.6%.

Impact of shrinkage. We show in fig. 5.5 (right) that using shrinkage improves the covariance estimates thereby improving the accuracy. We observe that shrinkage consistently improves performance, especially when the number of sampled means is small. When the number of means is low relative to the feature dimension ($d = 512$ in SqueezeNet), the covariance estimate becomes ill-conditioned, and shrinkage stabilizes it. However, as more class means are sampled per client or the number of clients increases, the benefit of shrinkage diminishes, since the total number of means approaches the feature dimension, making the estimate more stable.

Sampling multiple class means. We perform multiple class means sampling per client using ImageNet-R and show in fig. 5.6 (left) that using FedCOF with more class means shared from each client improves the performance. We also show in fig. 5.6 (center) the total number of means used per class on an average in fig. 5.6 (left) to perform the covariance estimation. The number of means used to estimate each class covariance is less than the total number of clients due to the class-imbalanced or dirichlet distribution used to sample data for clients. This is due to the fact that not all classes are present in all clients.

In most settings we observe that the largest performance improvement occurs when increasing the number of class means per client in scenarios with fewer clients. For instance, with only 10 clients, accuracy improves from 33.1% to 37.0% when increasing from 1 to 4 means per client. In contrast, when 50 clients are available, the improvement is marginal (from 37.4% to 37.8%). This trend is consistent with the theoretical insights: when fewer clients are available, sampling more means per client increases the number of independent statistics,

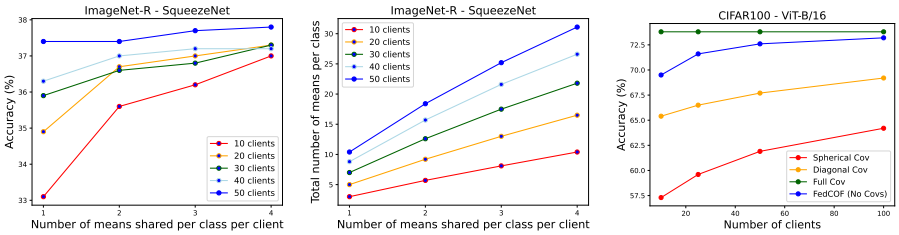


Figure 5.6: (left) Analysis of FedCOF performance with multiple class means per client on ImageNet-R. (center) Total number of means per class on average that are used to estimate the covariance for FedCOF in fig. 5.6 (left). (right) Performance comparison of FedCOF with full, diagonal, and spherical covariance matrix communication.

Table 5.10: Ablation showing the impact of using shrinkage in FedCOF using a pre-trained SqueezeNet.

Dataset	$\gamma = 0$	$\gamma = 0.01$	$\gamma = 0.1$	$\gamma = 1$	$\gamma = 10$
ImageNet-R	36.53	36.98	36.96	37.25	36.07
CUB200	51.08	51.07	51.81	53.57	53.50

thereby reducing variance and improving the quality of the covariance estimate.

Communicating diagonal or spherical covariances. While communicating diagonal or spherical covariances (mean of the diagonal covariance) from clients to server and then estimating the global class covariance from them can significantly reduce the communication cost, such estimates of global class covariance is poor compared to FedCOF. We show in fig. 5.6 (right) that FedCOF outperforms these covariance sharing baselines when communicating spherical or diagonal covariances.

Impact of Shrinkage. We analyze the impact of shrinkage on the estimated class covariances in FedCOF when using a pre-trained SqueezeNet in table 5.10. We use a shrinkage $\gamma = 1$ for our experiments with SqueezeNet and ViT-B/16. We observe that shrinkage yields marginal improvement for ImageNet-R and a bit more significant improvement in accuracy of 2.5% on CUB200. This can be attributed to the few-shot settings where the covariance estimation is not very good due to the lack of data and thus fewer clients having access to each of the classes. The use of shrinkage in FedCOF stabilizes and improves the covariance estimation leading to improved accuracy especially in few-shot settings.

Table 5.12: Total means shared per class on average (the number of clients each class is present in, on average).

Dataset	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$
ImageNet-R	36.1	17.4	11.7	4.0
Cars	24.5	13.4	9.6	3.6

Severe imbalance settings. We further evaluate FedCOF in settings with more severe class imbalance or heterogeneity across clients (Dirichlet distribution with $\alpha = 0.05, 0.01$) in table 5.11 and show that the performance of FedCOF drops a bit with very severe heteregeneity as expected.

Table 5.11: Analysis of FedCOF performance across settings with varying heterogeneity.

Class means shared	ImageNet-R (100 clients)				Cars (100 clients)			
per client	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$
1	38.4	37.3	36.4	33.5	45.0	44.5	43.1	39.9
2	38.2	37.8	37.4	35.9	44.9	44.5	43.8	42.5
3	38.1	37.5	37.5	36.7	44.9	44.7	44.2	43.2
4	38.3	37.7	38.2	37.4	45.1	44.9	44.4	43.6

We show the impact of class imbalance in different settings in table 5.12. Using the most extreme setting ($\alpha = 0.01$), each class is present on 4 clients on an average. Although the extreme settings are less unrealistic, we analyze and evaluate FedCOF in those settings. For instance, the drop in performance of FedCOF on Cars from 44.5 to 39.9 is due to the fact that the global covariance is estimated from around 3.6 sample means instead of around 13.4 sample means. This scenario faces the same issue as fewer clients due to the severe class imbalance. Here, we show that sampling multiple class means per client improves the performance in the extreme settings mitigating the bias in these settings.

5.5.4 Communication Costs

When computing communication costs we consider that the pre-trained models are on the clients similar to [38] and do not need to be communicated. All parameters are considered to be 32-bit floating point numbers (i.e. 4 bytes) in all our analyses and experiments.

For *training-free* methods, each client sends its local statistics to the server only once, as in [38, 88]. In the multi-round federated learning setting, this

can be efficiently implemented by ensuring on the client-side that each client only shares its local statistics during its first participation round. This avoids repeated communication of statistics from clients for all training-free methods. Following [38], the communication from server to client is not considered since the clients does not need to receive any updates from the server as the client models are not updated in the training-free initialization of the global classifier. Thus, the communication cost is the same for a single round setting with full client participation and the setting with multiple rounds having partial client participation in each round.

Let C_k denote the number of classes present in client k. Due to the non-iid Dirichlet data distribution, $C_k < C$, where C is the total number of classes in the dataset. As a result, the communication cost varies across clients. Defining $M = \sum_{k=1}^{K} C_k$ as the total number of class means shared from the K clients and assuming that each class mean is a d-dimensional feature vector, the communication cost of each evaluated approach is given by:

- FedNCM shares a total of M means from K clients. Cost $= Md \cdot 32$.
- Fed3R shares a total of M sum of class features and a total of K matrices of d^2 dimensionality from K clients. Cost $= (Md + Kd^2) \cdot 32$.
- FedCOF shares a total of M means from K clients. Cost $= Md \cdot 32$.
- CCVR and FedCOF-Oracle shares a total of M means and M covariance matrices of d^2 dimensionality from K clients. Cost $= M \cdot (d + d^2) \cdot 32$.

For the non-iid sampling in experiments reported in table 5.4, the values of M for CIFAR-100, ImageNet-R, CUB200, Cars and iNat-120K are 2870, 3470, 2353, 2634 and 54590, respectively.

Instead, for *federated finetuning*, the communication cost is calculated as follows. Let E, C and $H = d \cdot C$ denote the sizes of the feature extractor, the total number of classes and the classifier head, respectively. Assuming that $s < K$ is the number of clients participating in each round and T is the total number of communication rounds, the communication cost for federated *full-finetuning* is $2 \cdot (E + d \cdot C) \cdot T \cdot s \cdot 32$ and the cost for federated *linear probing* will be $2 \cdot d \cdot C \cdot T \cdot s \cdot 32$.

For *federated prompt-tuning* methods [189], considering the classifier weights $H = dC$, T rounds of communication, and s clients per round, the communication costs are:

- FedAvg-PT and FedProx-PT share the classifier weights $H = d \cdot C$ and the fixed size prompt pool P for each client of size d at each round. Cost $= 2 \cdot (d \cdot C + P \cdot d) \cdot T \cdot s \cdot 32$.

- PFPT shares the classifier weights $H = d \cdot C$ and a variable size prompt pool P_i for each client at each round i. Considering the sum of all P_i across all T rounds as $\sum_{i=1}^{T} P_i$, the communication cost is $2 \cdot (d \cdot C \cdot T + d \cdot \sum_{i=1}^{T} P_i) \cdot s \cdot 32$.

For the experiments reported in table 5.5, we follow the same training settings as [189] and train for 120 rounds with $s = 10$ clients participating per round. We use a fixed prompt pool size $P = 20$ for FedAvg-PT and FedProx-PT and a variable prompt pool size which is optimized over rounds for PFPT. The sums of the variable prompt pool sizes $\sum_{i=1}^{T} P_i$ across all rounds for CIFAR-100, ImageNet-R, CUB200 and Cars are 1777, 5205, 4736 and 4736 respectively.

Using PFPT [189] on CIFAR-100, the prompt pool size starts from 20 and ends at 10 after 120 rounds leading to improved performance and reduced communication compared to FedAvg-PT and FedProx-PT. This is consistent with the results from [189]. However, we observe in our experiments on out-of-distribution datasets like ImageNet-R and fine-grained datasets like CUB200 and Cars that the prompt pool size increases over rounds leading to increased communication costs with respect to FedAvg-PT and FedProx-PT. We also see in table 5.5 that all existing prompt-tuning methods performs very poorly on fine-grained datasets like CUB200 and Cars.

5.6 Improving Privacy Preservation in FedCOF

In this section we discuss additional privacy techniques to prevent the sharing of client class-wise count statistics and class means.

5.6.1 Adding Random Noise to Protect Class-wise Statistics

Our method requires transmitting class-wise statistics to compute the unbiased estimator of the population covariance (eq. (5.7)) and classifier initialization, similar to other methods in federated learning [88,106]. In general, transmitting the class-wise statistics may raise privacy concerns, since each client could potentially expose its class distribution. Inspired by differential privacy [36], we propose perturbing the class-wise statistics of each client with different types and intensities of noise, before transmission to the global server. This analysis allows us to evaluate how robust FedCOF is to variations in class-wise statistics and whether noise perturbation mechanisms can effectively hide the true client class statistics. Specifically, we propose perturbing the class-wise statistics as

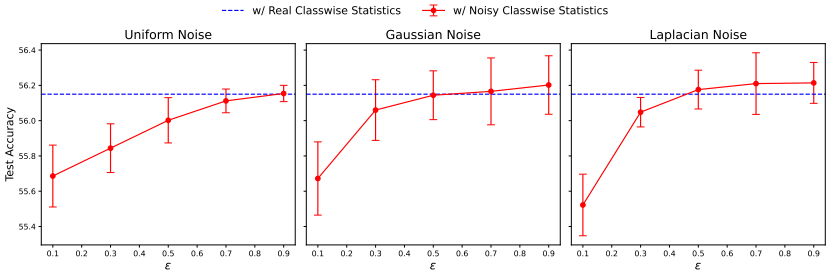


Figure 5.7: Performance of FedCOF with noisy class statistics on CIFAR-100 using SqueezeNet. The number of clients is fixed at 100 and classes are distributed using a Dirichlet distribution with $\alpha = 0.1$. Results are averaged over five random seeds, each generating different noise in client statistics, and the standard deviation is reported. FedCOF demonstrates robustness to uniform, Gaussian, and Laplace perturbations in class statistics, with performance showing a slight drop as noise, parameterized by ϵ, increases. Lower ϵ corresponds to higher noise levels in the class statistics.

follows:

$$\widetilde{n}_{k,c} = \max(n_{k,c} + \sigma_{\epsilon}^{\text{noise}}, 0) \tag{5.21}$$

where $\sigma_{\epsilon}^{\text{noise}}$ is noise added to the statistics, and ϵ is a parameter representing the noise intensity. The max operator clips the class statistics to zero if the added noise results in negative values, which is expected to happen in federated learning with highly heterogeneous client distributions. When clipping is applied, the client does not send the affected class statistic and class mean, and the server excludes them from the computation of the unbiased estimator.

We consider three types of noise:

- *Uniform noise*: $\sigma_{\epsilon}^{\text{unif}} \sim \mathcal{U}(-(1-\epsilon)n_{k,c}, +(1-\epsilon)n_{k,c})$, proportional to the real class statistics.
- *Gaussian noise*: $\sigma_{\epsilon}^{\text{gauss}} \sim \mathcal{N}(0, \frac{1}{\epsilon})$, independent of the real class statistics.
- *Laplacian noise* $\sigma_{\epsilon}^{\text{laplace}} \sim \mathcal{L}(0, \frac{1}{\epsilon})$, which is also independent of the real class statistics.

Lower ϵ values correspond to higher levels of noise in the statistics.

In fig. 5.7, we show that the performance of FedCOF is robust with respect to the considered noise perturbation, varying the intensity of $\epsilon \in$

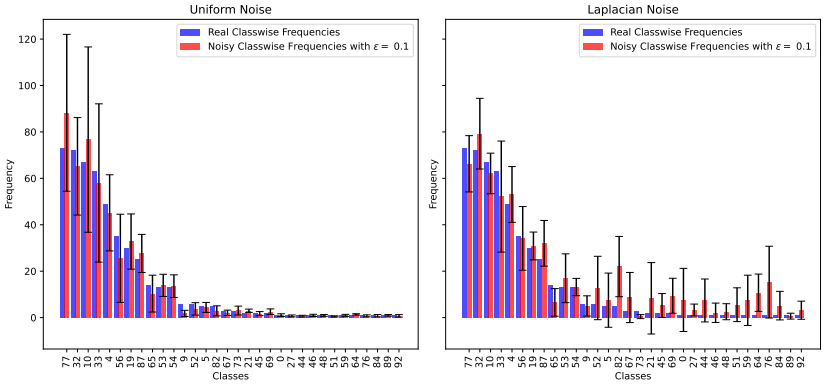


Figure 5.8: Class frequency distributions for a single client under different noise types: uniform noise (left) and Laplacian noise (right) on CIFAR-100. Both noise types are applied to the real class statistics with the highest noise intensity ($\epsilon = 0.1$). The bar heights represent the average class frequencies, and the error bars indicate the standard deviation across 5 seeds. Real class-wise frequencies and their noisy counterparts are shown for comparison.

$\{0.1, 0.3, 0.5, 0.7, 0.9\}$. These results suggest that a differential privacy mechanism can be implemented to mitigate privacy concerns arising from the exposure of client class-wise frequencies. In fig. 5.8, we provide a qualitative overview of how the proposed Laplacian and uniform noise perturbation affect class-wise distributions.

5.6.2 FedCOF with Secure Aggregation Protocols

To incorporate secure aggregation protocols with FedCOF, we propose to use pairwise masking between clients following [14]. Each client i generates a random noise vector $z_{i,j}$ for every other client j. Then, each of these clients i computes a perturbation vector $p_{i,j} = z_{i,j} - z_{j,i}$. Similarly, $p_{j,i} = z_{j,i} - z_{i,j}$ resulting in $p_{j,i} = -p_{i,j}$. These perturbation vectors are added to the client statistics before sharing them to the server. When all client statistics are added in the server, the noise cancels out across clients and the server can use the aggregate statistics.

To use such a secure aggregation protocol with FedCOF, the communication cost will increase and become similar to Fed3R. We proceed in two steps:

1. **Secure aggregation of class means and counts.** In this phase, each client sends only class means and counts, adding perturbation vectors to the means and perturbation scalars to the counts to ensure privacy. The server then computes the global class means from the aggregated statistics and sends them back to the clients. Specifically each client k sends:

$$u_{k,c} = q_k + n_{k,c}, \qquad v_{k,c} = p_k + \mu_{k,c} \times n_{k,c},$$

where $n_{k,c}$ and $\mu_{k,c}$ are the original class counts and means; $p_k = \sum_{i \in K, i \neq k} p_{k,i}$ is the total perturbation vector for client k, and $\sum_{k=1}^{K} p_k = 0$ (similarly q_k is a perturbation scalar such that $\sum_{k=1}^{K} q_k = 0$).

The server aggregates these quantities, and since the perturbations cancel out, it can compute:

$$\mu_c = \frac{\sum_{k=1}^{K} v_{k,c}}{\sum_{k=1}^{K} u_{k,c}}, \qquad N_c = \sum_{k=1}^{K} u_{k,c},$$

and sends (μ_c, N_c) back to all clients.

2. **Secure aggregation of class-covariance terms.** Each client computes

$$s_{k,c} = (N_c - 1) n_{k,c} (\mu_{k,c} - \mu_c)(\mu_{k,c} - \mu_c)^\top,$$

and forms

$$S_k = \sum_{c=1}^{C} s_{k,c} + M_k,$$

where M_k is a random matrix such that $\sum_{k=1}^{K} M_k = 0$.

After receiving all S_k, the server sums them, and since the noise terms cancel out, it obtains:

$$\sum_{k=1}^{K} S_k = \sum_{k=1}^{K} \sum_{c=1}^{C} (N_c - 1) n_{k,c} (\mu_{k,c} - \mu_c)(\mu_{k,c} - \mu_c)^\top.$$

The final global matrix is then estimated as:

$$\begin{aligned}\hat{G} &= \frac{1}{K-1}\left(\sum_{k=1}^{K} S_k + \sum_{c=1}^{C} K(N_c-1)\gamma I_d\right) + N\mu_g\mu_g^T \\ &= \frac{1}{K-1}\left(\sum_{k=1}^{K}\sum_{c=1}^{C}(N_c-1)n_{k,c}(\mu_{k,c}-\mu_c)(\mu_{k,c}-\mu_c)^\top\right) \\ &\qquad + \frac{K}{K-1}\sum_{c=1}^{C}(N_c-1)\gamma I_d + N\mu_g\mu_g^\top \\ &= \sum_{c=1}^{C}(N_c-1)\left[\frac{1}{K-1}\sum_{k=1}^{K} n_{k,c}(\mu_{k,c}-\mu_c)(\mu_{k,c}-\mu_c)^\top + \gamma I_d\right] + N\mu_g\mu_g^\top.\end{aligned}$$

where $N = \sum_{c=1}^{C} N_c$ and $\mu_g = \frac{\sum_{c=1}^{C}\mu_c}{N}$. This results in the exact same formulation of the matrix $\hat{G}$ as computed in the proposed FedCOF.

Although this increases training-free communication costs which becomes equivalent to Fed3R, FedCOF offers benefits in terms of accuracy and faster convergence which also reduces overall communication costs when fine-tuning after FedCOF initialization.

5.7 Conclusion

In this work we proposed FedCOF, a novel training-free approach for federated learning with pre-trained models. By leveraging the statistical properties of client class sample means, we showed that second-order statistics can be estimated using only class means from clients, thus reducing communication costs. We derived a provably unbiased estimator of population class covariances, enabling accurate estimation of a global covariance matrix. By applying shrinkage to the estimated class covariances and removing between-class scatter matrices, the server can effectively use this global covariance to initialize a global classifier. Our experiments demonstrated that FedCOF outperforms FedNCM [88] by significant margins while maintaining the same communication cost. Additionally, FedCOF delivers competitive or even superior results to Fed3R [38] across model architectures and benchmarks while substantially reducing communication costs. Finally, we showed that FedCOF outperforms federated prompt-tuning methods and serves as a more effective starting point for improving the convergence of federated fine-tuning and linear probing

methods.

Limitations. The quality of our estimator depends on the number of clients, as shown in fig. 5.5 where using multiple class means per client helps with fewer clients. Another limitation is the assumption that samples of the same class are iid across clients, which is, however, an assumption underlying most of federated learning. We discuss the bias in our estimator in non-iid settings.

6 Conclusions and Future Work

6.1 Conclusions

In this thesis, we have investigated several challenges and proposed novel approaches for continual and federated learning. We studied exemplar-free continual learning in two prominent settings: classifier-incremental learning and continual representation learning. We also explored exemplar-free continual learning for information retrieval using a new continual document retrieval benchmark. Finally, we studied training-free federated learning using pre-trained models. The major contributions from the thesis can be summarized as follows:

- **Chapter 2: Exploiting the Heterogeneity of Class Distributions in Exemplar-Free Continual Learning.** In this chapter, we have explored the classifier-incremental setting where the feature extractor is kept frozen after the first big task and only the classifier is continually updated. We emphasized the need to look beyond euclidean distance comparisons for classification since the feature space for new classes is not isotropic. We proposed a Mahalanobis distance-based classifier and demonstrated that it outperforms several competitive methods for EFCIL. The proposed method is widely applicable across other realistic settings like starting from a strong pre-trained model or in a few-shot continual learning setting.

- **Chapter 3: Resurrecting Old Classes with New Data for Exemplar-Free Continual Learning.** In this chapter, we have investigated the issue of semantic drift in feature representations of old classes when the entire network is updated on a new task. While exemplars from old tasks could be used to estimate the positions of old class prototypes in the updated space, this is not feasible in the exemplar-free setting. As an alternative approach, we proposed to adversarially perturb the current task samples to generate pseudo-exemplars for each old class

which could be then used to estimate the updated positions of those class prototypes. This approach is computationally cheap and achieves superior performance compared to several competitive methods across datasets.

- **Chapter 4: Enabling Compatibility in Continual Learning of Retrieval Embedding Models.** In this chapter, we have explored continual learning for document retrieval using pre-trained embedding models in a task-incremental setting. We discussed on the need to enable compatibility between embeddings of corpus documents indexed using an old task model and the query embeddings using the latest model. We proposed a novel approach of query drift compensation where the query embeddings of old tasks are moved back to their old embedding space by drift vector compensation. This enables re-indexing free document retrieval which outperforms other baselines and even re-indexing based approaches on the proposed continual document retrieval benchmark.
- **Chapter 5: Exploiting Mean Distributions for Training-free Federated Learning.** In this chapter, we have explored training-free FL methods using pre-trained models where we exploit the feature distributions from clients to obtain a strong global classifier at the server without any training. We proposed an approach of estimating the global class covariances at the server without sharing the covariances from the clients by using an unbiased covariance estimator which needs only the class means from clients. The proposed classifier initialization method reduces communication costs significantly while still achieving similar to better performance over existing methods across several benchmarks.

6.2 Future Work

The contributions of this thesis could be further explored in future works. Some promising directions could be as follows:

- In the continual representation learning setting, modeling the change in class covariances for old classes on updating the feature extractor (in addition to ADC) could enable the usage of a more effective Mahalanobis distance-based classifier instead of using a NCM classifier. This could potentially lead to improved continual classification performance.
- More advanced adversarial attack methods could be explored for ADC, which are more resistant to systems having strong adversarial defense

mechanisms. This will ensure that the attacks are successful and we always have some pseudo-exemplars for estimating prototype drift.

- A more comprehensive benchmark for continual document retrieval having more number of tasks with more domain shift would be very useful for the community to further research and develop more robust approaches for CDR. This would require more publicly available retrieval datasets having a significant amount of training query-document pairs.
- For training-free federated learning, future works can explore how to more accurately estimate covariances from means in feature-shift settings where the client feature distributions are significantly different. Beyond classifier initialization, the covariances estimated using FedCOF could also be used as a regularizer to guide the federated training updates (low-rank or full-rank). The proposed FedCOF estimator could be further explored in other settings involving Vision-Language Models (VLMs) or in unsupervised or self-supervised settings with no class labels which could use cluster centroids instead of class means.

Publications

1. **Goswami, D.**, Liu, Y., Twardowski, B., Van De Weijer, J. (2023). Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In The Thirty-seventh Annual Conference on Neural Information Processing Systems (**NeurIPS**).

2. Liu, Y., Cong, Y., **Goswami, D.**, Liu, X., Van De Weijer, J. (2023). Augmented box replay: Overcoming foreground shift for incremental object detection. In Proceedings of the IEEE/CVF international conference on computer vision (**ICCV**).

3. **Goswami, D.**, Soutif-Cormerais, A., Liu, Y., Kamath, S., Twardowski, B., Van De Weijer, J. (2024). Resurrecting old classes with new data for exemplar-free continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR**).

4. **Goswami, D.**, Twardowski, B., Van De Weijer, J. (2024). Calibrating higher-order statistics for few-shot class-incremental learning with pre-trained vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (**CVPRW**).

5. Gomez-Villa, A., **Goswami, D.**, Wang, K., Bagdanov, A. D., Twardowski, B., van de Weijer, J. (2024). Exemplar-free continual representation learning via learnable drift compensation. In European Conference on Computer Vision (**ECCV**).

6. **Goswami, D.**, Wang, L., Twardowski, B., van de Weijer, J. (2025). Query Drift Compensation: Enabling Compatibility in Continual Learning of Retrieval Embedding Models. Conference on Lifelong Learning Agents (**CoLLAs)**.

7. **Goswami, D.**, Magistri, S., Wang, K., Twardowski, B., Bagdanov, A. D., van de Weijer, J. (2025). Covariances for Free: Exploiting Mean Distributions for Training-free Federated Learning. In The Thirty-ninth Annual Conference on Neural Information Processing Systems (**NeurIPS**).

Under Review

1. Yu, L., Tao, Z., **Goswami, D.**, Yao, H., Twardowski, B., Van de Weijer, J., Xu, C. (2024). Exploiting the semantic knowledge of pre-trained text-encoders for continual learning.

2. Liu, Y., Hong, Q., Huang, L., Gomez-Villa, A., **Goswami, D.**, Liu, X., Van de Weijer, J. Tian, Y. (2025). Continual Learning for VLMs: A Survey and Taxonomy Beyond Forgetting.

Bibliography

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[2] Afra Feyza Akyürek, Ekin Akyürek, Derry Wijaya, and Jacob Andreas. Subspace regularizers for few-shot class incremental learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

[4] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[5] Mahdi Beitollahi, Alex Bie, Sobhan Hemati, Leo Maxime Brunswic, Xu Li, Xi Chen, and Guojun Zhang. Foundation models meet federated learning: A one-shot feature-sharing method with privacy and performance guarantees. *Transactions on Machine Learning Research*, 2025.

[6] Eden Belouadah and Adrian Popescu. Deesil: Deep-shallow incremental learning. In *European Conference on Computer Vision (ECCV) Workshops*, 2018.

[7] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *International Conference on Computer Vision (ICCV)*, 2019.

[8] Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR*

conference on Research and development in information retrieval, pages 192–199, 2000.

[9] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

[10] Prashant Shivaram Bhat, Bahram Zonooz, and Elahe Arani. Consistency is the key to further mitigating catastrophic forgetting in continual learning. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

[11] Prashant Shivaram Bhat, Bahram Zonooz, and Elahe Arani. Task-aware information routing from common representation space in lifelong learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[12] Niccoló Biondi, Federico Pernici, Matteo Bruni, Daniele Mugnai, and Alberto Del Bimbo. Cl2r: Compatible lifelong learning representations. *ACM Transactions on Multimedia Computing, Communications and Applications*, 18(2s):1–22, 2023.

[13] Niccolò Biondi, Federico Pernici, Simone Ricci, and Alberto Del Bimbo. Stationary representations: Optimally approximating compatibility and implications for improved model replacements. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28793–28804, 2024.

[14] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[15] Jorg Bornschein, Alexandre Galashov, Ross Hemsley, Amal Rannen-Triki, Yutian Chen, Arslan Chaudhry, Xu Owen He, Arthur Douillard, Massimo Caccia, Qixuan Feng, et al. Nevis' 22: A stream of 100 tasks sampled from 30 years of computer vision research. *Journal of Machine Learning Research*, 24(308):1–77, 2023.

[16] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[17] Yinqiong Cai, Keping Bi, Yixing Fan, Jiafeng Guo, Wei Chen, and Xueqi Cheng. L2r: Lifelong learning for first-stage retrieval with backward-compatible representations. In *Proceedings of the 32nd ACM International*

Conference on Information and Knowledge Management, page 183–192, 2023.

[18] Debora Caldarola, Barbara Caputo, and Marco Ciccone. Improving generalization in federated learning by seeking flat minima. In *European Conference on Computer Vision*, pages 654–672. Springer, 2022.

[19] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European Conference on Computer Vision (ECCV)*, 2018.

[20] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174, 2018.

[21] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[22] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. 2023.

[23] Zhixiang Chi, Li Gu, Huan Liu, Yang Wang, Yuanhao Yu, and Jin Tang. Metafscil: a meta-learning approach for few-shot class incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[24] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[25] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2021.

[26] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *International Conference on Computer Vision (ICCV)*, 2021.

[27] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 2000.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[30] Akshay Raj Dhamija, Touqeer Ahmad, Jonathan Schwan, Mohsen Jafarzadeh, Chunchun Li, and Terrance E Boult. Self-supervised features improve open-world learning. *arXiv preprint arXiv:2102.07848*, 2021.

[31] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] Xin Dong, Sai Qian Zhang, Ang Li, and HT Kung. Spherefed: Hyperspherical federated learning. In *European Conference on Computer Vision*, 2022.

[33] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[34] 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 (ICLR)*, 2021.

[35] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[36] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[37] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[38] Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Accelerating heterogeneous federated learning with closed-form classifiers. *Proceedings of the International Conference on Machine Learning*, 2024.

[39] Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

[40] Thomas Gerald and Laure Soulier. Continual learning of long topic sequences in neural information retrieval. In *Advances in Information Retrieval*, pages 244–259, 2022.

[41] Benyamin Ghojogh and Mark Crowley. Linear and quadratic discriminant analysis: Tutorial. *arXiv preprint arXiv:1906.02590*, 2019.

[42] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008*, 2018.

[43] Alex Gomez-Villa, Dipam Goswami, Kai Wang, Andrew D. Bagdanov, Bartlomiej Twardowski, and Joost van de Weijer. Exemplar-free continual representation learning via learnable drift compensation. In *Computer Vision – ECCV 2024*, pages 473–490, 2025.

[44] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICML)*, 2015.

[45] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[46] Dipam Goswami, Simone Magistri, Kai Wang, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost van de Weijer. Covariances for free: Exploiting mean distributions for training-free federated learning. In

The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[47] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bartlomiej Twardowski, and Joost van de Weijer. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28525–28534, 2024.

[48] Dipam Goswami, Bartłomiej Twardowski, and Joost Van De Weijer. Calibrating higher-order statistics for few-shot class-incremental learning with pre-trained vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[49] Dipam Goswami, Liying Wang, Bartłomiej Twardowski, and Joost van de Weijer. Query drift compensation: Enabling compatibility in continual learning of retrieval embedding models. In *Conference on Lifelong Learning Agents*, 2025.

[50] Samantha Guerriero, Barbara Caputo, and Thomas Mensink. Deepncm: Deep nearest class mean classifiers. *International Conference on Learning Representations Workshop (ICLR-W)*, 2018.

[51] Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, et al. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *arXiv preprint arXiv:2310.19923*, 2023.

[52] Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[54] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *International Conference on Computer Vision (ICCV)*, 2021.

[55] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[56] Jingrui Hou, Georgina Cosma, and Axel Finke. Advancing continual lifelong learning in neural information retrieval: definition, dataset, framework, and empirical evaluation. *Information Sciences*, 687:121368, 2025.

[57] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[58] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[59] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 76–92. Springer, 2020.

[60] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. 2016.

[61] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[62] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[63] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

[64] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023.

[65] Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

[66] Kishaan Jeeveswaran, Prashant Bhat, Bahram Zonooz, and Elahe Arani. Birt: Bio-inspired replay in vision transformers for continual learning. In *International Conference on Machine Learning (ICML)*, 2023.

[67] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[68] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[69] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[70] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih. Dense passage retrieval for open-domain question answering. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6769–6781. Association for Computational Linguistics (ACL), 2020.

[71] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[72] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[73] Gyuhak Kim, Bing Liu, and Zixuan Ke. A multi-head model for continual learning via out-of-distribution replay. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

[74] Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. Open-world continual learning: Unifying novelty detection and continual learning. *Artificial Intelligence*, 338:104237, 2025.

[75] Seongyoon Kim, Minchan Jeong, Sungnyun Kim, Sungwoo Cho, Sumyeong Ahn, and Se-Young Yun. Feddr+: Stabilizing dot-regression with global feature distillation for federated learning. *arXiv preprint arXiv:2406.02355*, 2024.

[76] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2017.

[77] Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q Weinberger. Incdsi: Incrementally updatable document retrieval. In *International conference on machine learning*, pages 17122–17134. PMLR, 2023.

[78] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[79] Jabin Koo, Minwoo Jang, and Jungseul Ok. Towards robust and efficient federated low-rank adaptation with heterogeneous clients. *arXiv preprint arXiv:2410.22815*, 2024.

[80] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision (ICCV-W) Workshops*, 2013.

[81] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[82] Shakti Kumar and Hussain Zaidi. Gdc-generalized distribution calibration for few-shot learning. *arXiv preprint arXiv:2204.05230*, 2022.

[83] Lilly Kumari, Shengjie Wang, Tianyi Zhou, and Jeff A Bilmes. Retrospective adversarial replay for continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[84] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2016.

[85] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 2019.

[86] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[87] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[88] Gwen Legate, Nicolas Bernier, Lucas Caccia, Edouard Oyallon, and Eugene Belilovsky. Guiding the last layer in federated learning with pretrained models. In *Advances in Neural Information Processing Systems*, 2023.

[89] Gwen Legate, Lucas Caccia, and Eugene Belilovsky. Re-weighted softmax cross-entropy to control forgetting in federated learning. In *Proceedings of The 2nd Conference on Lifelong Learning Agents*, 2023.

[90] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[91] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[92] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[93] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations*, 2021.

[94] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

[95] Zexi Li, Xinyi Shang, Rui He, Tao Lin, and Chao Wu. No fear of classifier biases: Neural collapse inspired federated learning with synthetic and fixed classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5319–5329, 2023.

[96] Zhizhong Li and Derek Hoiem. Learning without forgetting. *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2017.

[97] Davis Liang, Peng Xu, Siamak Shakeri, Cicero Nogueira dos Santos, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. Embedding-based zero-shot retrieval through query generation. *arXiv preprint arXiv:2009.10270*, 2020.

[98] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in Neural Information Processing Systems*, 35:17612–17625, 2022.

[99] Huan Liu, Li Gu, Zhixiang Chi, Yang Wang, Yuanhao Yu, Jun Chen, and Jin Tang. Few-shot class-incremental learning via entropy-regularized data-free replay. In *European Conference on Computer Vision (ECCV)*, 2022.

[100] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 226–227, 2020.

[101] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[102] Yuyang Liu, Yang Cong, Dipam Goswami, Xialei Liu, and Joost van de Weijer. Augmented box replay: Overcoming foreground shift for incremental object detection. In *International Conference on Computer Vision (ICCV)*, 2023.

[103] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning (CoRL)*, 2017.

[104] Jesús Lovón-Melgarejo, Laure Soulier, Karen Pinel-Sauvagnat, and Lynda Tamine. Studying catastrophic forgetting in neural ranking models. In *Advances in Information Retrieval*, pages 375–390, 2021.

[105] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021.

[106] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 2021.

[107] Chunwei Ma, Zhanghexuan Ji, Ziyun Huang, Yan Shen, Mingchen Gao, and Jinhui Xu. Progressive voronoi diagram subdivision enables accurate data-free class-incremental learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[108] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICML)*, 2018.

[109] Simone Magistri, Tomaso Trinci, Albin Soutif, Joost van de Weijer, and Andrew D. Bagdanov. Elastic feature consolidation for cold start exemplar-free incremental learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[110] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942, 2018.

[111] Tamasha Malepathirana, Damith Senanayake, and Saman Halgamuge. Napa-vq: Neighborhood-aware prototype augmentation with vector quantization for continual learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[112] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[113] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation. *Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2022.

[114] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Elsevier, 1989.

[115] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[116] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. Dsi++: Updating transformer memory with new documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8198–8213, 2023.

[117] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.

[118] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[119] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[120] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

[121] Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*, 2022.

[122] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, 2023.

[123] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning (ICML)*, 2017.

[124] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[125] John Nguyen, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? exploring the impact of pre-training and initialization in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[126] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.

[127] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

[128] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[129] Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E. Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[130] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[131] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[132] Can Peng, Kun Zhao, Tianren Wang, Meng Li, and Brian C Lovell. Few-shot class-incremental learning from an open-set perspective. In *European Conference on Computer Vision (ECCV)*, 2022.

[133] 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/CVF international conference on computer vision, pages 1406–1415, 2019.

[134] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Winter Conference on Applications of Computer Vision (WACV)*, 2023.

[135] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[136] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

[137] Vivek Ramanujan, Pavan Kumar Anasosalu Vasu, Ali Farhadi, Oncel Tuzel, and Hadi Pouransari. Forward compatible training for large-scale embedding retrieval systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19386–19395, 2022.

[138] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[139] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

[140] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[141] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[142] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

[143] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[144] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3403–3411. PMLR, 2021.

[145] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 2015.

[146] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations (ICLR)*, 2019.

[147] Dawid Rymarczyk, Joost van de Weijer, Bartosz Zieliński, and Bartlomiej Twardowski. Icicle: Interpretable class incremental continual learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[148] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[149] Simon Schrodi, David T. Hoffmann, Max Argus, Volker Fischer, and Thomas Brox. Two effects, one trigger: On the modality gap, object bias, and information imbalance in contrastive vision-language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[150] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[151] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom

Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[152] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR) workshops*, 2014.

[153] Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards backward-compatible representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6368–6377, 2020.

[154] Wuxuan Shi and Mang Ye. Prototype reminiscence and augmented asymmetric knowledge aggregation for non-exemplar class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[155] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *AAAI Conference on Artificial Intelligence*, 2021.

[156] Aliaksandra Shysheya, John F Bronskill, Massimiliano Patacchiola, Sebastian Nowozin, and Richard E Turner. Fit: Parameter efficient few-shot transfer learning for personalized and federated image classification. In *The Eleventh International Conference on Learning Representations*, 2022.

[157] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[158] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2021.

[159] James Seale Smith, Junjiao Tian, Shaunak Halbe, Yen-Chang Hsu, and Zsolt Kira. A closer look at rehearsal-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2410–2420, 2023.

[160] Albin Soutif-Cormerais, Marc Masana, Joost Van de Weijer, and Bartłomiej Twardowski. On the importance of cross-task features for class-incremental learning. *International Conference on Machine Learning (ICML) Workshops*, 2021.

[161] Andreas Peter Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *Transactions on Machine Learning Research*, 2022.

[162] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[163] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. Fedproto: Federated prototype learning across heterogeneous clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[164] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in Neural Information Processing Systems*, 2022.

[165] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[166] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[167] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, 2018.

[168] Marco Toldo and Mete Ozay. Bring evanescent representations to life in lifelong class incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[169] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations (ICLR)*, 2020.

[170] John W. Tukey. *Exploratory data analysis*. Addison-Wesley series in behavioral science : quantitative methods. Addison-Wesley, 1977.

[171] Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069, 2020.

[172] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[173] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[174] John Van Ness. On the dominance of non-parametric Bayes rule discriminant algorithms in high dimensions. *Pattern Recognition*, 1980.

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, and Aidan N Gomez. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[176] Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, et al. Continual learning: Applications and the road forward. *Transactions on Machine Learning Research*, 2024.

[177] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[178] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[179] Timmy ST Wan, Jun-Cheng Chen, Tzer-Yi Wu, and Chu-Song Chen. Continual learning for visual search with backward consistent feature embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16702–16711, 2022.

[180] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European Conference on Computer Vision (ECCV)*, 2022.

[181] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[182] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7611–7623. Curran Associates, Inc., 2020.

[183] Kai Wang, Luis Herranz, and Joost van de Weijer. Continual learning in cross-modal retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3628–3638, 2021.

[184] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[185] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[186] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision (ECCV)*, 2022.

[187] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[188] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2):513–535, 2023.

[189] Pei-Yau Weng, Minh Hoang, Lam Nguyen, My T Thai, Lily Weng, and Nghia Hoang. Probabilistic federated prompt-tuning with non-iid and imbalanced data. *Advances in Neural Information Processing Systems*, 37:81933–81958, 2024.

[190] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[191] Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern recognition*, 2008.

[192] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021.

[193] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[194] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. In *International Conference on Learning Representations (ICLR)*, 2021.

[195] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[196] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pages 1154–1156, 2021.

[197] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[198] Wang Yining, Wang Liwei, Li Yuanzhi, He Di, Chen Wei, and Liu Tie-Yan. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th annual conference on learning theory*, 2013.

[199] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[200] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016.

[201] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[202] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[203] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[204] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[205] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[206] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: a python toolbox for class-incremental learning. *SCIENCE CHINA Information Sciences*, 2023.

[207] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[208] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[209] Da-Wei Zhou, Han-Jia Ye, Liang Ma, Di Xie, Shiliang Pu, and De-Chuan Zhan. Few-shot class-incremental learning by sampling multi-phase tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[210] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Co-transport for class-incremental learning. In *ACM International Conference on Multimedia*, 2021.

[211] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[212] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[213] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[214] Kai Zhu, Kecheng Zheng, Ruili Feng, Deli Zhao, Yang Cao, and Zheng-Jun Zha. Self-organizing pathway expansion for non-exemplar class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[215] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 2019.