Efficient Lifelong Learning in Deep Neural Networks: Optimizing Architecture, Training, and Data

Sanket Vaibhav Mehta

CMU-LTI-23-016

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Emma Strubell (Chair), Carnegie Mellon University
William W. Cohen, Carnegie Mellon University & Google DeepMind
Aditi Raghunathan, Carnegie Mellon University
Dani Yogatama, University of Southern California & Reka

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Keywords: Lifelong Learning, Continual Learning, Meta Learning, Catastrophic Forgetting, Negative Interference, Forward Transfer, Pre-training, Flat Minima, Sharpness-Aware Minimization, Mixture of Experts

To Mom, Dad, and my dear sister, who made this day possible.

Abstract

The prevalent machine learning paradigm involves training a separate model for every new task given a static dataset. In contrast, humans accumulate knowledge over time, and the *lifelong learning* paradigm seeks to emulate this process by enabling systems to learn continuously from a stream of tasks, retaining past knowledge for efficient future learning. This paradigm also offers advantages such as avoiding periodic model training, potentially reducing computational and energy requirements, and promoting environmentally friendly Green AI. In modern machine learning, deep neural networks, while powerful, face challenges like *catastrophic forgetting* (losing knowledge from previous tasks during new task learning) and *negative interference* (previously learned knowledge hindering new task learning). These issues arise from the *stability-plasticity dilemma*, which necessitates finding the right balance between preserving past knowledge (stability) and acquiring new knowledge (plasticity). *Efficient* lifelong learning systems must address this dilemma, along with other considerations like supporting online data streams, utilizing small and fixed memory buffer capacity (if any), and learning from unlabeled data streams.

In this thesis, we derive inspiration from the biological learning process and recent progress in deep learning to enable *efficient lifelong learning systems*. We propose injecting inductive biases into the three main components of data-driven machine learning: *model* (architecture & initialization), *training* (objective & optimization), and *data*. This thesis is structured into three parts, each corresponding to one of these components. In the first part, we explore the role of *pre-trained initializations*, revealing their implicit alleviation of forgetting compared to random ones. Next, we design a *parameter-efficient expert architecture* that dynamically expands learning capacity to address the stability-plasticity dilemma. In the second part, we demonstrate that explicit *optimization for flat minima* improves network stability and introduce a *meta-learning objective* for stability-plasticity balance. The third part delves into lifelong semi-supervised learning, addressing the stability-plasticity dilemma by *rehearsing pseudo-labeled data*. We conclude by examining pre-training from the perspective of lifelong learning, showcasing enhancements by applying the above-developed strategies to the *(continual) pre-training of models*.

Acknowledgments

Embarking on the challenging journey of a Ph.D. has been a valuable experience for me, contributing to both my professional and personal growth. Through consistent work on various research projects, I have learned the art of research, honing skills like problem selection, scoping, scientific rigor, technical writing, and giving talks. While I do not claim mastery, I recognize this continuous learning process. My perspective on the Ph.D. journey is one of lifelong learning and meta learning, culminating in the attainment of a Doctor of Philosophy in AI. My academic journey, unlike many others, was not smooth sailing. It took an unconventional turn marked by the unexpected loss of my advisor, an abrupt stop in funding, and the onset of the COVID-19 pandemic, all in a single month. This led to a sudden halt in my Ph.D. progress without a clear way forward. Reflecting on completing this thesis, I have undergone a transformative journey, navigating from my lowest point to the finish line. This experience has underscored the optimism that arises from overcoming adversity, finding brightness in the aftermath of difficulties, and embracing hidden opportunities—echoing the timeless wisdom that "every cloud has a silver lining." I am profoundly grateful to those who played a pivotal role in guiding me toward this finish line. I wish to express sincere appreciation to my mentors, collaborators, mentees, friends, chingoos, and family. Without their unwavering support and encouragement, this thesis would not have come to fruition, and for that, I am deeply thankful.

First and foremost, I would like to thank my advisor, Emma Strubell, whose mentorship has been instrumental in shaping me into a well-rounded researcher. Emma is an excellent researcher, and her work in the field of Green AI has been groundbreaking and a constant inspiration for me. Her dedication and diligence as an advisor are commendable, and I consider myself fortunate to benefit from her guidance, expertise, and unwavering support throughout this academic journey. The knowledge I have gained from her extends beyond the confines of a single paragraph. She has also been a kind and caring mentor. During the transitional period following the loss of my previous advisor, she graciously accepted me as her first student at CMU, providing support as I navigated through uncertainties. She actively encouraged and guided me to mentor junior students to cultivate essential mentorship skills while providing invaluable support to junior colleagues. She genuinely cares for her students' well-being, and her efforts to organize outdoor social activities inspired my interest in both running and climbing. As an advisor-student duo, we participated in the CMU Annual Random Distance Run 2022, covering 1.75 miles, and secured the third position. The influence she has had on my academic and personal development is immeasurable, and I am grateful for having such an exceptional advisor during my doctoral studies.

I am very thankful to my committee members William W. Cohen, Dani Yogatama, and Aditi Raghunathan for their valuable feedback and comments. Special thanks to William for thoroughly reviewing this thesis, providing insightful suggestions for additional analyses, and identifying writing issues that I may have overlooked. I am excited about potential future collaborations with this stellar committee.

I express my deepest gratitude to my late advisor, Jaime Carbonell. I vividly recall his words after the Ph.D. admission committee meeting: "Congratulations, Sanket! I championed you! Good luck with your Ph.D., and never let me down!" Throughout my master's and early Ph.D., he patiently listened to my naive ideas as I grasped machine learning basics and offered insightful feedback. Jaime's sudden passing in February 2020 during my first Ph.D. year was the toughest moment in my graduate school journey. Everything came to a standstill with no clear direction ahead. In those tough times, I recalled his dedication, joining our meetings from his hospital bed — a symbol of responsibility. If he could endure pain without complaining, why should I complain about my challenges? This experience taught me a profound life lesson in resilience and gratitude. One of his final emails became my guiding light: "The best way you can help is continuing your excellent work and striving to finish your studies." As I write this, the emotion is overwhelming. I wish he could be here to witness my graduation moment, but the universe had different plans. Jaime, you were an extraordinary person, a fantastic advisor, and a lifelong mentor. Thank you for everything! You will never be forgotten, always missed. Your student forever!

I thank Barnabás Póczos for valuable advice during my master's studies, emphasizing collaborative learning with peers. I have carried this lesson into my graduate school and plan to keep it forever. I extend my thanks to Jamie Callan, Robert E. Frederking, Yiming Yang, Jay Yoon Lee, Sarath Chandar, and Zachary Lipton for their support and help after the loss of Jaime, leading up to my transition to Emma as my advisor. Next, I would like to thank those who played a major role in getting me to CMU. Special thanks to Ritwik Sinha, my undergraduate intern mentor at Adobe Research, who equipped me with foundational research skills and supported me throughout my academic applications. I appreciate the pivotal role of Shriram Revankar, my manager at Adobe Research, for providing a full-time opportunity and instilling a research philosophy. Thanks to Vishwa Vinay and Sunav Choudhary, peers at Adobe Research, for their research insights and ongoing support. Dhaval Patel, my undergraduate thesis advisor, deserves acknowledgment for imparting valuable lessons on good science. Special thanks to Sateesh Kumar Peddoju for enabling me to lead the ACM Student Chapter during my undergraduate years at IIT Roorkee, a leadership experience that prepared me for team projects in graduate school.

During grad school, I worked on various projects and gained valuable knowledge through collaboration with peers. I am grateful to Jay-Yoon Lee for guiding me in my master's and to CMU collaborators: Zirui Wang, Sang Keun Choe, Harsh Jhamtani, Jonathan Francis, Rajshekhar Das, Saurabh Garg, Bhargavi Paranjape, Sumeet Singh, and my mentees. Next, I am thankful to my collaborators from Mila and Meta: Sarath Chandar, Darshan Patil, Shagun Sodhani, Mojtaba Faramarzi, Pranshu Malviya, Mohamed Abdelsalam, Janarthanan Janarthanan. Special thanks to Yi Tay, Jai Gupta, Jinfeng Rao, and Donald Metzler for hosting me at Google, and the rest of my Google collaborators: Vinh Q. Tran, Tal Schuster, Mostafa Dehghani, Dara Bahri, Kai Hui, Mihir Kale, Ankur Parikh, Vamsi Aribandi, and Marc Najork. Despite the ups and downs in our relationship, I appreciate the anonymous reviewers for their actionable feedback, especially recognizing the JMLR Action Editor and reviewers.

I express gratitude to all the students at CMU whom I had the privilege of mentoring. Witnessing their joy upon submitting their initial papers has been the most rewarding aspect of being a mentor. Guiding several students and projects has allowed me to delve into research fundamentals, discern successful approaches, and understand the ingredients for initiating and completing projects. This has shaped my Ph.D. journey with a meta-learning perspective. Beyond professional development, mentoring has positively influenced my personal growth. In my first year of Ph.D. at CMU, I mentored visiting students from South Korea for Jaime. Jimin Sun and Hwijeen Ahn from this batch later joined the master's program at LTI, shaping my health and fitness habits. Jimin influenced my lifestyle positively, introducing me to activities such as yoga, running, and occasional cheat days with dosa and mango lassi. Hwijeen introduced me to biking by lending me his vintage 1986 Schwinn Super Sport bike for a summer. Clara Na elevated my running experience by guiding me in long-distance training and live-tracking my completion of the Pittsburgh half marathon. Jared Fernandez initiated me into indoor climbing, while Sophia Roshal guided the outdoor top roping sessions. Health enthusiasts Jimin Mun and Simran Khanuja initially aimed for strength training but ultimately fostered my healthy dietary habits, leading to a reduced intake of Chipotle rice bowls (NaCl) and Parle-G. Saujas Vaduguru guided me in perfecting dosa with coconut chutney, and Akhila Yerukola's preference for milkless chai prompted experiments with water-to-milk ratios. Even in my absence from Hagwon (GHC 5501), Amanda Bertsch and Sireesh Gururaja upheld high working standards, reinforcing my belief in the positive impact of collaborative learning among peers. As Lindia Tjuatja and Jared discovered the magic of masala chai and Maggi with at-home Hagwon sessions, I mastered the art of consuming cereal—and trust me, it is not about the milk or the cereal. Su Bin Jung globally launched virtual hagwon sessions on Zoom, yet Yewon Byun's extensive use significantly improved my remote mentoring skills. Furthermore, Yewon's playful use of Korean motivated me to pick up the language at LTI. While I was running, climbing, and biking, So Yeon Min introduced me to Gadi walking, which led me to discover more about my personality. Mentees kept arriving one after another, but what remained constant were the desks at GHC 5501. Gratitude goes to my office mates, Yingshan Chang and Siddhant Arora, who generously allowed me to use their desks whenever they were unoccupied. While mentoring students over the years, Sang Keun Choe regularly visited Hagwon accompanied by his dog, Betty. I have known Sang since the fall of 2018, and through our interactions, I gained insights into the intricacies of startups. It seems that Hagwon has functioned as a startup incubator all along, prioritizing research, academic pursuits, mentorship, and fostering a collaborative work environment. Lastly, my heartfelt gratitude goes to Betty, whose presence brought joy and solace to Hagwon. At my farewell, the heartfelt words from my very first mentee, Kundan Krishna, are indelibly imprinted in my mind: "While many pursue a Ph.D., only a few genuinely earn the love and respect of an entire group of juniors. You deserve every bit of it." Expressing gratitude for the countless, lifelong memories! Farewell to Hagwon; you will be sincerely missed. As I pass the GHC 5501 baton to Danny To Eun Kim, I wish him good luck.

I thank all of the SLAB, YES, COMEDY, and Jaime's lab members who made my CMU journey so memorable and gave me so much to learn. With gratitude to my labmates mentioned earlier, I extend my appreciation to others: Jeremiah Milbauer, for his insightful conversations and broad knowledge, remaining a teammate at CMU or Google; Nupoor Gandhi, for sharing valuable lessons about Indian tourism; Zhisong Zhang, for meticulously reviewing my manuscripts; Hao Zhu, for climbing lessons and introducing me to my first Jain hotpot; Vidhi Jain, for engaging discussions on Jain philosophy and appealing to my Jain sensibilities; Xuhui Zhou, for maintaining a friendly demeanor despite my occasional lack of honesty; Vijay Viswanathan, for the tip that Patel Brothers offers the cheapest Parle-G, ensuring an unlimited supply; Josh Zhanson, for engaging in CV discussions; Zhiruo Wang, for the counter++; Rosa Vitiello, for being patient during my random CLAW lab visits; Cathy Jiao, for Vancouver whereabouts; Jing Yu Koh, our local tourist guide for EMNLP@Singapore; Ashique KhudaBukhsh, for passionate CSS discussions and constant support following Jaime's passing. A big shout-out to Yonatan Bisk, Maarten Sap, Daniel Fried, and Sherry Tongshuang Wu, who, along with Emma, organized our lab meetings and sponsored many social gatherings. Special mention to Emma and Yonatan for semester-long soft-skills-focused lab meetings. Going forward I am sad to miss Yonatan's lessons on proper English usage during our lab meetings. However, with our discussions on flat minima, the prospect of a Flat DOSA remains alive. Next, special thanks to Stacey Young, Kate Schaich, and Tessa Samuelson who facilitated navigating the academic aspects of LTI, SCS, and CMU. I would like to thank Boeing and DSO Labs Singapore for funding my graduate school studies, particularly recognizing Tom Vu, Luo Qi Chan, Jing Lim, and Chieu Hai Leong.

I am profoundly grateful for my friends, whose constant support has made my Ph.D. journey happier and more enjoyable. Living with my roommate, the tech and policy nerd Divyansh Kaushik, I gained some expertise in Punjabi cooking, from patiently slow-cooking Dal Makhani for 6 hours to realizing that Ghee is all you need. Beyond the culinary adventures, our conversations delved into the science of machine learning, shaping my research taste—zeroing in on fundamental questions often overlooked. With Kundan Krishna, from his Pittsburgh debut to my farewell, we shared countless moments, my favorite being educational 'Chai Pe Charcha' sessions with Parle-G and Rusk on the side. With my earlier office mates, Sai Krishna Rallabandi and Khyathi Raghavi Chandu, we engaged in numerous whiteboard brainstorming sessions and philosophical discussions, along with weekly 'pet pooja' gatherings. Among many memories, there is the unforgettable moment of Khyathi teaching me to dance at Sai-Suchi's wedding, my presence at Khyathi-Abhishek's wedding, and the delightful surprise of Sai-Suchi's daughter, Veda, visiting GHC 5501 and attending my thesis defense. A shout-out to my master's years roommate, Raghuram Mandyam Annasamy, for consistently having my back — that support holds steady even now. Special gratitude to Sreeja and Raghuram for generously hosting me during my Bay Area sojourn and providing an endless feast of delicious homemade dosas. Another earlier roommate, the chai virtuoso Ankit Parag Shah, whose influence on my chai skills is truly noteworthy.

Thanks, Jon, for leading my first camping and memorable hikes with Jimin, Jared, Clara, Nupoor, Jeremiah, Zhisong, Xinyi Wang, Ziyu Xu, Bingqing Chen, Adithya Pratapa and Torsten Wörtwein. Special thanks to Raj, Jon, Jimin, Hwijeen, Jared, and Lindia for the Choolaah gatherings, Prakhar Gupta for the pizza lessons, Aman Madaan for the Silicon Valley Vista point tour, and Lucio Dery and Shengyu Feng, for their sociable and cheerful personalities. Thanks, Sophia, Hwijeen, Clara, Jared, Kaixin Ma, Hao, and Xuhui for being my climbing squad, and Ritam Dutt and Yiyuan Li for being chai explorers. Shuyan Zhou, thanks for the invite (tfti), Sriram Narayanan, Shaily Bhatt, and Harshita Diddee for vibing on food, Patrick Fernandes and Paul Pu Liang for parties, Taro Tsuchiya for Cylab stories, Daye Nam and Minji Yoon for being Korean seon saeng nim, Jessica Huynh, Harvineet Singh, and Aayush Sharma for the apartment sublease, Leena Mathur, Saujas, Simran, Sriram and Hwijeen for the Frick Park Nine Mile Run trail, and Prateek Joshi, Dylan Sam, and Alex Wilf for being Hagwon's best partners.

A shoutout to Bhargavi and Sumeet for our first lifelong learning paper, Vidhisha Balachandran, Xiaochuang Han, and Jiateng Xie for co-TAing courses with me, Sreecharan Sankaranarayanan, Shrimai Prabhumoye, Dheeraj Rajagopal, Varun Gangal, Aakansha Naik and Abhilasha Ravinchandar as goto seniors for any queries, Shruti Palaskar for being my student contact, Biswajit Paria, Hai Pham and Amrith Setlur for being my labmates under Barnabás. Gratitude to Sumit Agarwal, Minkai Deng, and David Bick for the mentoring opportunities, Juncheng Billy Li for ML chats, Saurabh for our late-evening random walks and ML talks, Ananye Agarwal for being my late-night shuttle buddy. Thanks, Tzu-Hsiang Lin, Alankar Jain, Vaibhav, Aldrian Obaja, Gayatri Bhat, and Vasu Sharma for MLT days, Aditi Chaudhary, Nidhi Vyas, Shirley A. Hayati, Harsh, and Divyansh for memorable Brussels moments, Pratyush Maini, Adithya, Chan Young Park and Sachin Kumar for fun Singapore moments, Sachin Goyal, Brandon Trabucco, Pratyush for Navratri Bhoj, Maneesh Bilalpur, Kundan and Prakhar for Diwali celebrations, Rishabh Joshi, Sopan Khosla, Tanmay Parekh, and Mukul Bhutani for bridging Pittsburgh to Bay Area. Special thanks to my undergraduate friends for their ongoing support: Rahul Raj, Shagun Sodhani, Surendra Kumar Gadwal, and Pranay Chaudhary.

Above all, I would like to express my deepest gratitude to my dad, Vaibhav Mehta, my mom, Swati Mehta, my dear sister, Dr. Sujal Mehta, and my brother-in-law, Dr. Sahil Shah, for their unwavering love and support throughout my career. No words can truly capture the extent of their contributions. Having been away from home for over a decade, the distance became even more significant in Pittsburgh, spanning continents. Yet, they never failed to send boxes of ladoos and snacks for every Diwali, Rakhi for Raksha Bandhan, and patiently responded to my countless calls during the last 333 weeks from Pittsburgh. While others at CMU praised my positive energy levels, today I acknowledge that it is my family that has been the wellspring of my energy all these years — I was merely transferring it. Needless to say, this thesis would not have been possible without their encouragement along the way. Thank you SVM family! This thesis is dedicated to you! I trust I have done justice to this acknowledgment, and I apologize sincerely if there are any names I may have missed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade, advances in training hardware and the availability of large datasets have enabled deep neural networks to make significant progress in the field of machine learning. These networks have reached or exceeded the human-level performance in numerous natural language processing and computer vision tasks, such as machine translation (Lepikhin et al., 2021), question-answering (Du et al., 2022; Chowdhery et al., 2023), open-ended dialogue generation (Ouyang et al., 2022), object detection, and image generation (Lu et al., 2023), when evaluated on independent and identically distributed (i.i.d) holdout data. However, these networks tend to perform worse when applied to realistic situations where the data distribution changes over time (Lazaridou et al., 2021). The primary reason behind their failure is that the current approach to machine learning concentrates on isolated learning (Chen and Liu, 2018), i.e., training a separate network for each new task or set of related tasks using a stationary dataset. One way to keep these networks up-to-date is by re-training them from scratch every time new information becomes available. However, the data used for previous training may only be temporarily available due to privacy or storage limitations (Farquhar and Gal, 2018). In addition, the re-training approach can be computationally expensive, data inefficient, and time-consuming, especially for large networks. For instance, GPT-3 (Brown et al., 2020), an auto-regressive language model with 175B parameters, trained with 499B tokens, used compute equivalent to $3.14e^{23}$ floating point operations and would require 355 years and $\$4.6$M to train on a single NVIDIA Tesla V100 GPU[1].

Another approach is to update networks with new information as it arrives continuously. However, deep neural networks, and generally parametric models, are prone to the phenomenon of *catastrophic forgetting* (McCloskey and Cohen, 1989; Ratcliff, 1990; French, 1999). In this phenomenon, networks forget or overwrite previously learned knowledge as new information is incorporated into the system. Additionally, these networks may experience the phenomenon of *negative interference* (Pan and Yang, 2009; Weiss et al., 2016), in which previously learned knowledge may hinder the efficient learning of new things, resulting in increased data requirements. These two phenomena stem from the *stability-plasticity* dilemma (Mermillod et al., 2013). *Stability* relates to preserving past knowledge, and *plasticity* relates to learning new knowledge. A balance is needed, as too much stability hinders new knowledge acquisition, and too much plasticity causes previous knowledge to be forgotten. This dilemma makes it challenging for current networks to

[1] OpenAI's GPT-3 Language Model: A Technical Overview

update their knowledge and adapt efficiently to new tasks incrementally.

In contrast, we humans learn quite differently. We learn by acquiring and updating knowledge throughout our lifetime, retaining previously learned knowledge, and using it to facilitate efficient learning of new concepts and skills. Motivated by this human learning process, the *lifelong learning* (Thrun and Mitchell, 1995; Thrun, 1995; Chen and Liu, 2018) or *incremental learning* (Solomonoff et al., 1989; Syed et al., 1999; Ruping, 2001) or *never-ending learning* (Mitchell et al., 2018) or *continual learning* (Parisi et al., 2019) paradigm aims to develop systems to learn from a continuous stream of data, ideally preserving past knowledge, updating it with new information and leveraging it for subsequent learning. Also, researchers have recognized the importance of lifelong learning capability to progress toward achieving artificial general intelligence (Silver, 2011; Chen and Liu, 2018; Yogatama et al., 2019). Apart from the resemblance to biological learning, the lifelong learning paradigm also has the potential to reduce energy waste by obviating excessive model re-training and enabling environmentally friendly and sustainable Green AI (Hazelwood et al., 2018; Strubell et al., 2019; Schwartz et al., 2020).

The lifelong learning paradigm is also related to other knowledge transfer-related paradigms like transfer learning (Pan and Yang, 2009) and multi-task learning (Caruana, 1997). Unlike these two paradigms, the lifelong learning paradigm is more general; assuming sequential access to tasks, it aims to improve performance on both previous (ideally positive backward transfer or negative forgetting) and new (positive forward transfer) tasks. The contemporary transfer learning paradigm primarily focuses on uni-directional knowledge transfer from previous tasks to improve performance on a new task, even if that hurts the performance on previously learned tasks. On the other hand, multi-task learning assumes simultaneous access to data from all tasks and aims to improve performance on all tasks by enabling knowledge sharing between them. Furthermore, neural networks are shown to experience catastrophic forgetting even in single-task learning setup (Toneva et al., 2019), highlighting that the lifelong learning paradigm is not just restricted to multi-task scenarios. Even the notion of the task is very much open-ended in the lifelong learning paradigm. For example, consider a lifelong COVID-19 Named Entity Recognition (NER) tagger. There are three different manifestations of tasks – (i) classification *tasks* like entity chunking, entity detection, entity linking, co-reference resolution, and relationship extraction, (ii) NER on temporary varying *domains* of COVID-19 research articles for years 2020, 2021, 2022, 2023 (iii) NER for evolving *classes* of COVID-19 variants like COVID-Alpha, COVID-Beta, COVID-Omicron. These manifestations correspond to three prominent scenarios in lifelong learning: *task*, *domain*, and *class* incremental learning (Van de Ven and Tolias, 2019).

In addition to addressing catastrophic forgetting, there are several other goals for lifelong learning systems (Biesialska et al., 2020). Humans quickly learn new information from an ongoing conversation without clear topic boundaries (Chen and Liu, 2018). We selectively retain past experiences in our limited memory capacity to prevent forgetting and sparsely replay them as needed (Ratcliff, 1990; McGaugh, 2000). Additionally, we often learn in an unsupervised manner from our environment rather than relying on explicit supervision (Aljundi, 2019). In contrast, current lifelong learning systems (Biesialska et al., 2020) require explicit task boundaries, are data inefficient as they rely on large memory capacity, and are computationally expensive as they demand multiple passes over labeled data. To mimic human learning more effectively, it's necessary to develop lifelong learning systems that operate under more realistic assumptions and are data, memory, and computationally efficient (Farquhar and Gal, 2018).

1.1 Research Goals

In this thesis, *we aim to design efficient lifelong learning systems that alleviate catastrophic forgetting of previously learned knowledge and facilitate future learning by operating under realistic assumptions*. Inspired by the biological learning process and recent progress in deep learning, we propose injecting appropriate inductive biases into the three main components of data-driven machine learning: **model**, **training**, and **data**. By doing so, we also hope to increase efficiency in data, memory, and computational requirements for lifelong learning systems.

Model (architecture & initialization). To construct a deep neural model, we start with an *architecture* like ResNet (He et al., 2016) or Transformer (Vaswani et al., 2017), set the *initial value of the architecture's parameters*, and then use the available data to train the model and determine the optimal parameter values. Recent works (Glorot and Bengio, 2010; LeCun et al., 2015; Mishkin and Matas, 2015) have shown that the initial values of the parameters are critical for the model's learning dynamics and ultimate performance. Toward this end, various initialization schemes have been introduced for fully-connected networks (Pennington et al., 2017), residual networks (He et al., 2015), convolutional networks (Xiao et al., 2018), recurrent networks (Chen et al., 2018), and attention-based networks (Huang et al., 2020). However, these schemes are specific to a particular architecture and associated activation functions and do not transfer to variations of existing architectures or completely novel architectures. To address this limitation, a recent work (Dauphin and Schoenholz, 2019) introduces a recipe for automating the search for good initializations using task-agnostic meta-learning.

At the same time, transfer learning has shown impressive results in both computer vision (Zhuang et al., 2021) and natural language processing applications (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019). The modern transfer learning paradigm involves *pre-training* a fixed architecture, like ResNet (He et al., 2016) or BERT (Devlin et al., 2019), using copious amounts of data, and then *fine-tuning* the learned parameters on target tasks. The pre-training stage can be viewed as a data-driven method of finding a good initialization scheme, similar to how early experiences shape the development of brain structure in humans and provide a foundation for lifelong learning (Ackerman, 1992; Black et al., 2017). Given the tremendous success of pre-trained models, there has been increased interest in understanding their role in improving generalization (Erhan et al., 2010; Neyshabur et al., 2020), speed of convergence (Hao et al., 2019), successful transfer (He et al., 2019; Pruksachatkun et al., 2020), and out-of-distribution robustness (Hendrycks et al., 2020; Tu et al., 2020). Despite these efforts, **the role of pre-trained initializations in lifelong learning settings has been under-explored**[2].

In recent years, the Transformer (Vaswani et al., 2017) architecture has become a popular choice for neural network design due to its effectiveness across various modalities, including text (Devlin et al., 2019; Raffel et al., 2020), vision (Dosovitskiy et al., 2021), audio (Gong et al., 2021), and speech (Dong et al., 2018). The Transformer architecture and its variants consist of multiple layers, each comprising multi-headed self-attention and feed-forward sub-layers. While these networks are modular in terms of their layers and attention heads, they are still monolithic,

[2]In fact, one of the original motivations for transfer learning was as a way to enable lifelong learning, discussed in a NIPS-95 workshop on "Learning to Learn" (Pan and Yang, 2009).

with a fixed pre-defined network topology (e.g., the outputs of one layer feed into the next layer). The monolithic nature of these architectures makes them prone to catastrophic forgetting. Adapting them to new tasks may cause the parameters associated with previously learned tasks to be overwritten, as there is no clear separation between them. In contrast, Simon (1962) posited that most complex systems are modular and that their near-decomposability, or the presence of "modules-within-modules" (Meunier et al., 2009), is crucial for their ability to adapt quickly to changing environmental conditions while retaining previously learned knowledge. This suggests that an architecture consisting of multiple modules or experts, each specializing in a particular subset of tasks, can help mitigate the risk of catastrophic forgetting. When adapting to new tasks, only a subset of the parameters related to those tasks will be affected, while the parameters associated with previously learned tasks will be preserved. Recently, sparse expert models (Fedus et al., 2022a) like Switch Transformers (Fedus et al., 2022b), Mixture-of-Experts (Artetxe et al., 2022), ST-MoE (Zoph, 2022) have been proposed as a way to scale large language models to trillion parameters while keeping computations efficient by selecting a subset of parameters (or experts) for each example. Despite being well-suited to lifelong learning systems design, **sparse expert models have not been thoroughly studied for their ability to alleviate forgetting and facilitate lifelong learning**.

Training (optimization & objective). To train a deep neural network, we require training data, an objective function that measures the model's performance, and an optimization algorithm that searches for the optimal parameters that minimize this function. Although the optimization process focuses on minimizing the loss function on the training data, the ultimate goal is to *generalize* to holdout i.i.d. data. The training loss landscapes of the deep neural networks are complex, resulting in convergence to different global minima (or optimal parameters) based on the training dynamics. Moreover, these minima are widely different in terms of their generalization capabilities (Keskar et al., 2017). As a result, there has been a surge in research to alter the training dynamics by employing techniques like dropout (Hinton et al., 2012b; Srivastava et al., 2014), batch normalization (Ioffe and Szegedy, 2015) or optimizers like Stochastic Gradient Descent (SGD; Bottou (1999)), RMSProp (Hinton et al., 2012a), Adam (Kingma and Ba, 2014), Adafactor (Shazeer and Stern, 2018). Given the prevalence of these techniques and optimizers, it is vital to understand their implications for the stability-plasticity dilemma. Towards this end, Goodfellow et al. (2013) argues that dropout increases the optimal size of the network by regularizing and constraining the network capacity to be barely sufficient to perform the first task, thereby reducing forgetting of the previous task. Mirzadeh et al. (2020a) provides an alternative explanation that dropout learns a gating mechanism such that different network paths are active for different tasks, minimizing interference during sequential learning and retaining stability. Wei et al. (2020) shows that dropout encourages flat minima by implicitly regularizing the activation norm, and Mirzadeh et al. (2020b) argues that such flat minima reduce forgetting. Furthermore, by modifying the hyper-parameters like learning rate and batch size, one can promote flat minima (Keskar et al., 2017) and reduce forgetting (Mirzadeh et al., 2020b). With these studies establishing the connection between flat minima and forgetting, **it would be efficient to directly optimize for flat minima instead of relying on an ad-hoc approach of varying learning rate decay, batch size, and dropout regularization with the hope of converging to flat minima**.

After discussing the impact of the training procedure, we will now turn our attention to designing objective functions for continual learning. A lifelong learning system must learn from a sequence of tasks, and the objective is to minimize the average loss function of all tasks. However, due to the sequential nature of the input task stream, one may not have access to previous data. Due to this, the system only optimizes for the current task loss function (too much plasticity), resulting in a loss of previously consolidated knowledge (stability). This kind of *vanilla* sequential learning creates a stability-plasticity imbalance, and using previous task data in a multi-task learning strategy could mitigate this imbalance. The challenge of balancing stability and plasticity is also well-known in biological systems (Grossberg, 1987; Mermillod et al., 2013). To maintain this balance and, in turn, retain previously acquired knowledge, humans rely on *episodic memory* to store a *subset* of past experiences and conduct *sparse* experience rehearsal to reinforce older tasks (Ratcliff, 1990; Robins, 1995; McGaugh, 2000). Straightforward application of episodic memory for lifelong language learning requires large working memory to store past inputs and frequent replaying of them, thereby increasing storage requirements (Biesialska et al., 2020). One may understand how humans employ episodic memory and mimic it to reduce these requirements. According to the complementary learning systems (CLS; McClelland et al., 1995; O'Reilly and Norman, 2002), lifelong learning in humans necessitates two complementary learning phases: a slow-learning phase for structured knowledge and a fast-learning phase for episodic information, allowing for the gradual accumulation of knowledge and rapid adaptation to individual experiences without interference. Inspired by this work, it would be interesting to study whether **explicitly optimizing for complementary learning behaviors during lifelong learning helps reduce the size of episodic memory along with balancing stability-plasticity dilemma**.

Data (limited labeled and unlabeled). Supervised learning involves training a model on a dataset of labeled examples, each with an input and a corresponding label. Similarly, lifelong supervised learning involves training a model on a sequence of *labeled* datasets, each corresponding to a different task (Chen and Liu, 2018). However, continually assuming access to the labeled data stream is unrealistic in some lifelong learning scenarios. For instance, consider a neural corpus indexer that memorizes the document corpus to retrieve documents corresponding to the input queries (Tay et al., 2022). This indexer constitutes a well-suited candidate for lifelong learning; the indexing process is continuous and cumulative. In other words, as new documents arrive, the neural indexer needs updating. However, one may not have access to ground-truth queries corresponding to the incoming documents; nevertheless, the indexer should still be able to answer input queries for already and newly indexed documents. This application scenario motivates designing lifelong learning systems capable of continuously learning from an unlabeled data stream, like humans, who continually learn in an unsupervised manner from our surroundings rather than relying on direct supervision. Chen and Liu (2018) refers to this learning setup as *lifelong semi-supervised learning* with never-ending language learner (NELL; Mitchell et al., 2018) constituting the only known system for this setup. The NELL architecture does not utilize deep neural models, which means it does not experience the forgetting phenomenon. **Given the increasing adoption of large neural models** (Radford et al., 2018; Devlin et al., 2019; Raffel et al., 2020; Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2023; Touvron et al., 2023)**, it is crucial to examine lifelong semi-supervised learning with these models**.

Like lifelong learning, transfer learning focuses on learning generic knowledge from (potentially unlabeled) data to transfer it to subsequent tasks. Recently, large language models like RoBERTa(Liu et al., 2019b), T5 (Raffel et al., 2020), GPT-3 (Brown et al., 2020), and LLaMA 2 (Touvron et al., 2023) have become the de facto starting point for transfer learning on many NLP tasks. The basic idea is to pre-train these models on self-supervised tasks and then fine-tune them on the downstream tasks of interest. For pre-training, self-supervised tasks like masked language modeling are constructed from massive text corpora containing billions of words to learn general language patterns and features (Devlin et al., 2019). Despite sharing similarities with how humans learn language by making inferences based on the context, pre-training is still "data-hungry" (e.g., GPT-3 trained with 499B tokens) compared to humans learning a language with relatively small amounts of data. To further understand the learning dynamics of language models during pre-training, Liu et al. (2021) systematically analyses the kinds of knowledge these models acquire and what time during pre-training. Using RoBERTa, Liu et al. (2021) report that linguistic knowledge is acquired fast and stably, facts and commonsense are acquired slowly, and reasoning abilities are not stably acquired. Now let us *view the pre-training through the lens of the lifelong learning paradigm*. We receive an incoming stream of pre-training data, and we continuously update the model with the pre-training objective. Moreover, given the large size of the pre-training corpora, we rarely revisit any previously seen examples. With this view, the findings from Liu et al. (2021) suggest that these models undergo the phenomenon of forgetting during pre-training, thus, suffering from the stable acquisition of knowledge and ending up being data-hungry. It is intriguing to **examine the role of lifelong learning in pre-training, especially in alleviating forgetting and improving sample complexity**.

1.2 Thesis Overview and Contributions

In Figure 1.1, we summarize our work when applied to different scenarios:

- goals **(G)** of lifelong learning systems - reducing catastrophic forgetting (or backward transfer or stability), enabling forward transfer (or plasticity), addressing backward vs. forward transfer tradeoff (stability-plasticity dilemma), sample efficiency
- lifelong learning scenario **(S)** - task-incremental, domain-incremental, class-incremental, single-task learning
- desiderate **(D)** of lifelong learning systems - online/ offline learning, no task boundaries, a large number of tasks in sequence, limited memory, unlabeled data streams
- proposed methodology **(M)** - flat minima (pre-training and SAM), meta-learning, semi-supervised learning, mixture-of-experts
- applications **(A)** used for evaluations - text and image classification, question-answering, (continual) pre-training, memorization, document indexing and retrieval

Background. In Chapter 2, we provide an overview of the fundamental concepts related to lifelong learning, including problem formulation, different lifelong learning scenarios, performance measurement criteria, benchmarks for conducting experiments, and prominent baselines for comparison. This chapter serves as a foundation for the subsequent chapters.

G: Goal, **M:** Method, **S:** Setup, **A:** Application

Model	Initialization	**Ch3. Role of Pre-training in Lifelong Learning** (Mehta et al., 2023b) **G:** catastrophic forgetting, **M:** pre-training, **S:** task incremental, **A:** text/ image classification tasks
	Architecture	**Ch4. Dynamic Parameter-Efficient Experts for Lifelong Learning** (Byun et al., 2023) **G:** backward vs forward transfer, **M:** mixture-of-experts, **S:** domain incremental, **A:** QA
Training	Optimization	**Ch5. Lifelong Learning with Sharpness Aware Minimization** (Mehta et al., 2023b) **G:** catastrophic forgetting, **M:** sharpness-aware minimization, **S:** task incremental, **A:** text/ image classification tasks
	Objective	**Ch6. Efficient Meta Lifelong Learning with Limited Memory** (Mehta et al., 2020) **G:** backward vs forward transfer, **M:** meta-learning, **S:** class/ domain incremental, **A:** text classification/ QA
Data	Limited Labeled	**Ch7. Lifelong Semi-Supervised Learning for Updating Transformer Memory** (Mehta et al., 2023a; Mehta et al., 2022) **G:** backward vs forward transfer, **M:** semi-supervised learning, **S:** class incremental, **A:** document indexing & retrieval
	Unlabeled	**Ch8. Role of Lifelong Learning in Pre-training** (Mehta et al., 2023a; Choe et al., 2023) **G:** catastrophic forgetting and sample efficiency, **M:** sharpness-aware minimization and meta-learning, **S:** single-task, **A:** pre-training

An Introduction to Lifelong Supervised Learning (Sodhani et al., 2022)

Figure 1.1: **Thesis overview.** We summarize our work when applied to different goals **(G)** and desiderata **(D)** of lifelong learning systems, lifelong learning scenarios **(S)**, proposed methodology **(M)** to address them, and applications **(A)** used for evaluations. We view the development of **efficient** lifelong learning systems through the lens of machine learning basics—Which model **architecture** is well-suited for designing such systems? What role does model **initialization** play in alleviating forgetting? How do we address different desiderata of these systems by modifying training **objective** and **optimization/** training dynamics?
How to sample/ generate **data** for training such systems?

Part I: Model (Architecture & Initialization).

- **Initialization.** In Chapter 3, we investigate the role of pre-training in lifelong learning, specifically focusing on the issue of catastrophic forgetting (or stability). We investigate existing methods in the context of large, pre-trained models and evaluate their performance on various text and image classification tasks, including a large-scale study using a novel dataset of 15 diverse NLP tasks. Across all settings (task-incremental learning, offline learning with task boundaries, and a large number of tasks in a sequence), we observe that generic pre-training implicitly alleviates catastrophic forgetting when learning tasks sequentially compared to randomly initialized models. Moreover, *we report that "vanilla" sequential fine-tuning with pre-trained initializations outperforms sophisticated methods for reducing forgetting when applied to randomly initialized models, presenting a much simpler and more efficient solution.* We then delve deeper into *why* pre-training can help prevent forgetting in this scenario. To understand this phenomenon, we analyze the loss landscape and *infer that pre-trained initialization seems to help reduce forgetting by converging to flat minima* (Mehta et al., 2023b).

- **Architecture.** In Chapter 4, we propose Generate to Discriminate for Expert Routing (G2D*f*ER), a continual learning method that leverages modern generative language models to generate per-domain synthetic examples for purposes of domain discrimination (rather than generative replay). We then leverage this discriminator to route each example to the best expert at inference time. Concretely, for each new domain, we: (i) fine-tune a domain-specific expert model; (ii) fine-tune a generative model and sample synthetic examples; and (iii) train a domain discriminator to predict which domain a given sample is drawn from using generated samples from all seen domains. At inference time, we pass samples through our domain discriminator, which routes each sample to its corresponding expert. We show that this expert-based approach is competitive with previous state-of-the-art in domain-incremental learning and *outperforms in scenarios characterized by stringent constraints on data sharing* (Byun et al., 2023).

Part II: Training (Objective & Optimization).

- **Optimization.** In Chapter 5, we build on our earlier findings (from Chapter 3) that pre-trained initialization reduces forgetting by converging to flat minima. We propose jointly optimizing for current task loss and loss basin sharpness to explicitly encourage wider basins during sequential fine-tuning. Concretely, we use the Sharpness-Aware Minimization (SAM) procedure to seek parameters in neighborhoods with uniformly low loss values (or flat loss regions). We show that this optimization procedure leads to performance comparable to the state-of-the-art in task-incremental learning across multiple settings (offline training with task boundaries and a large number of tasks in a sequence) *without retaining an episodic memory that scales in size with the number of tasks* (Mehta et al., 2023b).
- **Objective.** In Chapter 6, we work on alleviating catastrophic forgetting (retaining stability) while enabling positive forward transfer (supporting plasticity). Driven by the CLS theory, we identify three components of an existing method—generic representations, experience rehearsal, and local adaptation, each corresponding to one of the two learning phases, that are independent rather than complementary. We introduce a "synergistic" framework that makes two learning phases complementary. We propose a novel first-order meta-learning objective that formulates the slow-learning phase as the meta-task (generic representation and experience rehearsal) and the fast-learning phase as the base task (local adaptation). Across different challenging settings (domain/class-incremental learning, online learning with no task boundaries), we show that *our framework prepares the slow-learning phase for faster local adaptation and the fast-learning phase to support reduced memory buffer size* (Mehta et al., 2020).

Part III: Data (Limited Labeled & Unlabeled).

- **Limited labeled data.** In Chapter 7, we work on a lifelong semi-supervised learning setup to learn continuously from an unlabeled data stream (in contrast to our work in the above chapters). Concretely, we consider Differentiable Search Indices (DSIs) to encode a corpus of documents in the parameters of a model and use the same model to map queries directly to relevant document identifiers. We introduce DSI++, a continual learning

challenge for DSI to continuously index new documents while being able to answer queries related to both previously and newly indexed documents. Across different model scales and document identifier representations, *we show that continual indexing of new documents leads to considerable forgetting of previously indexed documents*. Next, we introduce a parametric memory to generate pseudo-queries for the retrieval task. We supplement them during incremental indexing to *prevent forgetting older documents (stability) and enable continuous semi-supervised learning with new documents (plasticity)* (Mehta et al., 2023a, 2022).

- **Unlabeled data.** In Chapter 8, we propose viewing pre-training from a lifelong learning perspective and design methods to reduce forgetting during pre-training and enhance sample complexity. Building on the above works, in this chapter, we answer the following research questions—(i) What should be the initialization scheme for pre-training? For instance, we show that meta learning-based learned initialization does not influence forgetting behaviors during the pre-training of language models; (ii) What should be the training dynamics for pre-training? Based on our work in Chapter 5, we demonstrate that *the SAM optimization procedure helps with stable learning in a single-task learning setting (e.g., memorization task from Chapter 7), thereby generalizing to situations where data does not undergo a clear distribution shift* (Mehta et al., 2023a); and (iii) What data should be used for pre-training? Low-quality or noisy samples may hurt learning by amplifying negative interference (or forgetting); therefore, by employing a scalable meta-learning algorithm, *we reweigh (or filter) them for sample-efficient (continual) pre-training and demonstrate convincing benefits in terms of downstream performance* (Choe et al., 2023).

Chapter 2

Lifelong Learning

In this chapter, we provide an overview of the fundamental concepts related to lifelong learning, including the problem formulation (Section 2.1), performance measurement criteria (Section 2.2), benchmarks for conducting experiments(Section 2.3), and prominent baselines (Section 2.4). This chapter serves as a foundation for the subsequent chapters.

2.1 Lifelong Learning Formulation

Supervised Learning. The objective of supervised learning is to learn a function, $f_w : \mathcal{X} \to \mathcal{Y}$ such as a neural network, parameterized by $w \in \mathbb{R}^P$, that can predict an outcome $y \in \mathcal{Y}$ for a given input sample $x \in \mathcal{X}$. To do so, we assume access to some training data corresponding to a task t, $\mathcal{D}^t_{train} = \{(\boldsymbol{x}^t_i, y^t_i)\}_{i=1}^{n_t}$, consisting of n_t pairs of input samples x^t_i and their labels y^t_i, drawn i.i.d from task-specific (unknown) fixed distribution $P_t(\mathcal{X}, \mathcal{Y})$. We hope to learn f_w such that for a new pair, $(x, y) \sim P_t$, $\hat{y} \approx y$, where $\hat{y} = f_w(x)$. To learn f_w, we also need a non-negative real-valued loss function ℓ that measures how different the prediction $\hat{y}$ of f_w is from the ground truth y on the training data. Then the risk $R(f_w)$ associated with f_w is defined as the expectation of the loss function ℓ

$$R(f_w) = \mathbb{E}_{x,y\sim P_t}\left[\ell(f_w(x), y)\right], \tag{2.1}$$

with the goal to find the optimal function f_{w^*} (or the optimal parameters w^*) that minimize the risk $R(f_w)$

$$w^* = \arg\min_w R(f_w). \tag{2.2}$$

However, the risk $R(f_w)$ cannot be computed because the distribution P_t is unknown. Therefore, the Empirical Risk Minimization principle (ERM; Vapnik, 1991) is employed, which suggests minimizing the empirical risk $\hat{R}(f_w)$ instead of the true risk $R(f_w)$

$$\hat{R}(f_w; \mathcal{D}^t_{train}) = \frac{1}{n_t} \sum_{(x^t_i, y^t_i)\in\mathcal{D}^t_{train}} \ell(f_w(x^t_i), y^t_i), \tag{2.3}$$

$$w^* = \arg\min_w \hat{R}(f_w; \mathcal{D}^t_{train}). \tag{2.4}$$

Lifelong Learning. In the lifelong learning setting, the system receives an infinite stream of training examples $\mathcal{D}_{train}$. Unlike typical supervised learning, the training samples are not i.i.d and are considered to be split into $\mathcal{T}$ disjoint subsets, $\mathcal{D}_{train} = \{\mathcal{D}^1_{train}, \cdots, \mathcal{D}^{\mathcal{T}}_{train}\}$, each of which corresponds to a task t. Next, the data corresponding to each task, $\mathcal{D}^t_{train} := \{(x^t_i, y^t_i)\}^{n_t}_{i=1}$, is assumed to be drawn from a different i.i.d distribution $P_t(\mathcal{X}, \mathcal{Y})$. Lifelong learning aims to learn a single model, $f_w : \mathcal{X} \rightarrow \mathcal{Y}$ (e.g., a neural network parameterized by $w \in \mathbb{R}^P$), capable of effectively generalizing to examples from any task, despite being trained on multiple tasks in sequential order. Moreover, in an ideal lifelong learning setting, the model should learn from a stream of input examples without a task descriptor, i.e., the model is unaware of the task identifier t for a sample during training and testing. This setting is ubiquitous in practice, as environments consistently evolve without sending an explicit signal (Farquhar and Gal, 2018). Formally, the objective function is to minimize the average expected risk of all tasks in a sequence. Applying the ERM principle (Vapnik, 1991), we hope to minimize the average empirical risk of all $\mathcal{T}$ tasks

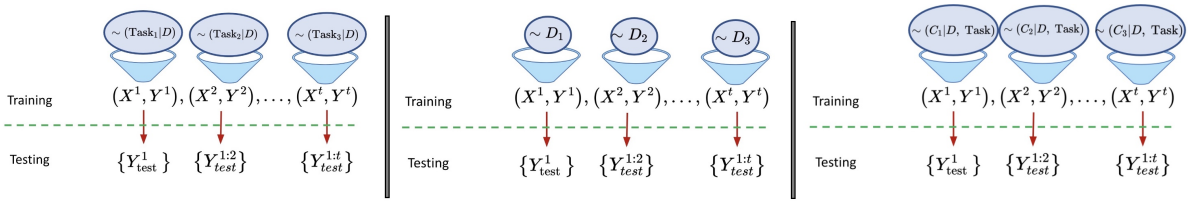
$$\hat{R}_{\mathcal{T}}(f_w; \mathcal{D}_{train}) = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \frac{1}{n_t} \sum_{(x^t_i, y^t_i) \in \mathcal{D}^t_{train}} \ell(f_w(x^t_i), y^t_i), \tag{2.5}$$

where ℓ is the task-specific loss function and $n_t = |\mathcal{D}^t_{train}|$. However, during lifelong learning, data from older tasks ($t < \mathcal{T}$) is unavailable, and so it is infeasible to minimize the average empirical risk in Equation 2.5. As a consequence, the system solely minimizes the empirical risk of the currently available task, potentially leading to the forgetting of previous tasks. The terms *stability* and *plasticity* are used to describe the preservation of past knowledge and the acquisition of new knowledge, respectively. Present neural networks exhibit excessive plasticity and insufficient stability. Maintaining a balance is crucial, as excessive stability impedes the acquisition of new knowledge, while excessive plasticity results in the forgetting of previous knowledge. This interplay between stability and plasticity constitutes the *stability-plasticity dilemma*, and our thesis focuses primarily on addressing this dilemma.

As stated above, in the lifelong learning setting, it is assumed that there are clear and well-defined boundaries between the tasks to be learned, and the system is unaware of the task identifier during training and evaluation. However, depending on the use case scenarios for lifelong learning systems, the task identifier may (or may not) be available during inference time, thereby influencing the solutions for addressing the stability-plasticity dilemma. Due to this, Van de Ven and Tolias (2019) defines three prominent scenarios in lifelong learning: *task-incremental*, *domain-incremental*, and *class-incremental* learning. These scenarios differ on whether the task identifier is available during inference time and, if it is not, whether the system should infer it. In Figure 2.1, we visualize three different scenarios and discuss them in detail in the following sections.

2.1.1 Task-Incremental Learning

Task-incremental learning is a scenario where a model is trained on a sequence of tasks with known task identifiers, making it the simplest type of lifelong learning scenario. With the task identifier always available, models can have task-specific components, such as a multi-headed output layer in deep neural networks, where each task has its output units. For instance, in the case



(a) Task-Incremental (b) Domain-Incremental (c) Class-Incremental

Figure 2.1: Overview of the three lifelong learning scenarios (adapted from Sodhani et al., 2022)

of a COVID-19 Named Entity Recognition (NER) tagger, under the task-incremental learning scenario, different (related) classification tasks like entity chunking, entity detection, entity linking, co-reference resolution, and relationship extraction would have different classification heads but may share the rest of the network between them. In this scenario, different tasks have different label spaces, i.e., $\mathcal{Y}_i \neq \mathcal{Y}_j \; \forall i, j \in \{1, \cdots, \mathcal{T}\}$. After sequential training on tasks, the model is evaluated on each task separately, meaning only the classes corresponding to that task are considered. The overall model performance is average across all tasks after being trained sequentially on all tasks (see Figure 2.1a). Although the overall performance is most commonly evaluated after the model has seen all tasks, one can also evaluate a specific task at different stages during continual learning to evaluate the model's robustness against catastrophic forgetting and negative interference.

2.1.2 Domain-Incremental Learning

In a domain-incremental learning scenario (see Figure 2.1b), a model learns from a sequence of tasks without knowing task identifiers during training and inference time. This scenario is when the underlying task structure remains consistent, but the input data distribution evolves, referred to as the "domain" of the input. For instance, in the case of the COVID-19 NER tagger, a domain-incremental learning scenario constitutes training a system sequentially on temporally evolving articles from 2020, 2021, and 2022 with the articles from every year treated as a separate task (or domain). Note that the underlying task of NER is the same across different domains; hence the label space $\mathcal{Y}$ remains fixed across all domains. Consequently, the output layer in the deep neural network is shared. However, the marginal or conditional distributions over $\mathcal{X}$ and $\mathcal{Y}$ can change, i.e., $P_i(\mathcal{X}) \neq P_j(\mathcal{X})$ and $P_i(\mathcal{Y}|\mathcal{X}) \neq P_j(\mathcal{Y}|\mathcal{X})$, $\forall i, j \in \{1, \cdots, \mathcal{T}\}$. Similar to the task-incremental learning scenario, the overall model performance is averaged across all domains after being trained sequentially on evolving domains.

2.1.3 Class-Incremental Learning

In a class-incremental learning scenario, the task identifier is unavailable during inference time. So a model must infer a task identifier to solve each task seen so far, making this the most challenging lifelong learning scenario and bringing it closer to mimicking the human learning setting. As a result, deep neural networks incorporate a single-headed output layer that is shared among all tasks. For example, a COVID-19 NER tagger must learn new classes to support evolving virus

variants (COVID-Alpha, COVID-Beta, and COVID-Omicron) as they appear, creating a class-incremental lifelong learning scenario. In this scenario, the output label space keeps evolving, i.e., $\mathcal{Y}^i \subset \mathcal{Y}^j,\ \forall i < j,\ i, j \in \{1, \cdots, \mathcal{T}\}$. During the evaluation, the model's predictions are assessed against all the classes it has encountered thus far (see Figure 2.1c). The overall performance is reported to be averaged for all tasks (each task corresponds to a subset of classes added in a sequence).

2.2 Performance Metrics in Lifelong Learning

Let $S_{t,\tau}$ denote the accuracy on task τ after training on task t. After model finishes training on the task t, we compute the *average accuracy* (A_t), *forgetting* (F_t) and *learning accuracy* (LA_t) metrics as proposed by Lopez-Paz et al. (2017); Riemer et al. (2019). F_t (also referred to as backward transfer) measures the influence of learning task t on the performance of all previously seen tasks $\tau, (1 \leq \tau < t)$. As the model learns multiple tasks in the sequence, we hope that knowledge acquired during lifelong learning aids in learning new tasks (forward transfer). LA_t measures the learning capability when the model sees the new task t (indirectly measuring the forward transfer). Say we learn the t^{th} task, then A_t, F_t and LA_t are defined as follows

$$A_t = \frac{1}{t}\sum_{\tau=1}^{t} S_{t,\tau}, \qquad F_t = \frac{1}{t-1}\sum_{\tau=1}^{t-1} \max_{\tau' \in \{1,\cdots,t-1\}} (S_{\tau',\tau} - S_{t,\tau}), \qquad LA_t = \frac{1}{t}\sum_{\tau=1}^{t} S_{\tau,\tau}. \quad (2.6)$$

2.3 Tasks and Benchmarks for Lifelong Learning

2.3.1 Text Classification

We conduct experiments on Split YahooQA (homogeneous) and 5-dataset-NLP/15-dataset-NLP (diverse). In the homogeneous scenario, tasks are generated from the same underlying distribution or dataset. In contrast, the diverse setting involves tasks that originate from different underlying distributions or datasets. We examine these two task categories to investigate potential variations in forgetting trends within this context.

Split YahooQA consists of five homogeneous text classification tasks and is built from a 10-way topic classification dataset (YahooQA; Zhang et al., 2015) by randomly splitting topics into different tasks. Each task involves 2-way text classification and comprises approximately 279k/12k examples for the training and testing sets, respectively.

5-dataset-NLP consists of five diverse text classification tasks (Zhang et al., 2015) spanning different text classification datasets: (1) news classification (AGNews), (2) sentiment analysis (Yelp, Amazon), (3) Wikipedia article classification (DBPedia) and (4) questions and answers categorization (Yahoo). To compare our framework with d'Autume et al. (2019), we follow the same data processing procedure to produce balanced datasets. For the task-incremental learning scenario, we have 115k/7.6k (train/test) examples per task. For the class-incremental learning scenario, we have 33 classes, $575,000$ training examples, and $38,000$ test examples from all tasks.

Task	Dataset/ Corpus	Domain(s)/ Text source(s)	|Train|	|Val|	|Test|	|L|	Metrics
Linguistic Acceptability	CoLA	Journal articles & books	7,695	856	1,043	2	Matthews correlation
Boolean Question Answering	BoolQ	Google queries, Wikipedia passages	8,483	944	3,270	2	Acc.
Sentiment Analysis	SST-2	Movie reviews	9,971	873	872	2	Acc.
Paraphrase Detection	QQP	Quora questions	10,794	4,044	4,043	2	Acc. & F1
Q & A Categorization	YahooQA	Yahoo! Answers	13,950	4,998	4,998	10	Acc.
Review Rating Prediction	Yelp	Business reviews	12,920	3,999	3,998	5	Acc.
Event Factuality	Decomp	FactBank	10,176	4,034	3,934	2	Acc.
Argument Aspect Detection	AAC	Web documents	10,893	2,025	4,980	3	Acc. & F1
Explicit Discourse Marker Prediction	DISCONN8	Penn Discourse TreeBank	9,647	1,020	868	8	Acc. & F1
Question Answering NLI	QNLI	Wikipedia	9,927	5,464	5,463	2	Acc.
Binary Sentence Order Prediction	RocBSO	Roc story, corpus	10,000	2,400	2,400	2	Acc.
Natural Language Inference	MNLI	speech, fiction, govt. reports	11,636	4,816	4,815	3	Acc.
Multi-choice Science QA	SciTAIL	Science exams	11,145	1,305	1,304	2	Acc.
Implicit Discourse Relation Classification	PDTB2L1	Penn Discourse TreeBank	13,046	1,183	1,046	4	Acc. & F1
Emotion Detection	Emotion	Twitter	9,600	2,000	2,000	6	Acc. & F1

Table 2.1: **15-dataset-NLP**: Task/dataset description and statistics. All tasks are either single sentence or sentence pair classification. |Train|, |Val|, and |Test| denote the number of examples in the train, validation, and test splits respectively. |L| denotes the number of classes for each task.

15-dataset-NLP One of the objectives of our thesis is to study the role of different pre-trained initializations in lifelong learning. To enable this study, we introduce 15-dataset-NLP, a novel suite of diverse tasks for lifelong learning. It consists of fifteen text classification tasks covering a broad range of domains and data sources. Although there exists a setup with 4 tasks spanning 5 datasets, 5-dataset-NLP (de Masson D'Autume et al., 2019), we show that our introduced benchmark proves more challenging (see Table 5.3 and Section 3.3.3) than the previous setup for the Transformer models (e.g., DistilBERT, BERT, RoBERTa) considered in our thesis.

This benchmark consists of single-sentence or sentence pair classification tasks. We design our benchmark from existing tasks such that (1) the overall dataset includes various domains, (2) different tasks are (dis)similar to each other, thereby, facilitating both transfer and interference phenomena. All tasks under consideration differ in dataset size (from 8.5k-400k), so for our experiments, we only use between 8.5k-14k training examples from each task. Lifelong learning from highly imbalanced data is an interesting problem, and we feel that our introduced benchmark can be used to investigate this problem as well. As our data is gathered from publicly available sources, for some tasks we do not have access to hidden test examples. In such cases, we consider dev examples as test split and sample examples from train split for validation We describe the tasks below and Table 2.1 details the evaluation metrics and train/dev/test split sizes for each task.

1. Linguistic acceptability aims at identifying whether the given sequence of words is a grammatical sentence. The Corpus of Linguistic Acceptability (*CoLA*; Warstadt et al., 2019) consists of English sentences annotated with their grammatical judgments. The data spans multiple domains, specifically books, and journal articles.
2. Boolean QA is a reading comprehension task of answering yes/no questions for a given passage. The Boolean Questions (*BoolQ*; Clark et al., 2019) data set consists of short passages with yes/no questions about the passage. The questions are sourced from anonymous Google users and paired up with passages from Wikipedia articles.
3. Sentiment analysis is a binary classification task of identifying the polarity (positive/negative sentiment) of a given text. The Stanford Sentiment Treebank (*SST-2*; Socher et al., 2013) corpus consists of sentences from Rotten Tomatoes movie reviews annotated with their sentiment.
4. Paraphrase detection aims at identifying whether two sentences are semantically equivalent. The Quora Question Pairs (*QQP*) corpus consists of question pairs from *Quora*[1] website annotated for semantic equivalence of question pairs.
5. Q&A categorization is a topic classification task of categorizing question and answer text pairs into existing topics. The Yahoo! Answers Comprehensive Questions and Answers (*YahooQA*; Zhang et al., 2015) corpus contains data corresponding to the ten largest categories from Yahoo! Webscope program.
6. Review rating prediction is a five-way classification task of predicting the number of stars the user has given in a review given the corresponding text. The *Yelp* (Zhang et al., 2015) data set contains business reviews obtained from the Yelp Dataset Challenge (2015).
7. Event factuality prediction is the task of determining whether an event described in the

[1]https://www.quora.com/share/First-Quora-Dataset-Release-Question-Pairs

text occurred. The factuality annotations from the *Decomp* corpus are recast into an NLI structure and we use the modified data set from Diverse NLI Collection (Poliak et al., 2018).

8. Argument aspect mining is concerned with the automatic recognition and interpretation of arguments (assessing the stance, source, and supportability for a given topic). The Argument Aspect Corpus (*AAC*; Stab et al., 2018) has over 25,000 arguments spanning eight topics annotated with three labels (no argument, supporting argument, opposing argument). Stab et al. (2018) collected the data from web documents representing a range of genres and text types, including blogs, editorials, forums, and encyclopedia articles.
9. The explicit discourse marker prediction task aims at classifying the discourse markers between sentences. Specifically, words like 'and', 'but', 'because', 'if', 'when', 'also', 'while', and 'as' mark the conceptual relationship between sentences (*DISCONN8*) and are considered as labels for this task as discussed in (Prasad et al., 2019; Kim et al., 2020). We use examples from the Penn Discourse Treebank 3.0 marked for explicit discourse relationship for our experimentation.
10. Question-answering NLI (*QNLI*) is a task adapted from the SQuAD by converting it into the sentence pair classification task (Wang et al., 2019). QNLI is a binary classification task of detecting whether the context sentence contains the answer to the question.
11. Binary Sentence Ordering (BSO) is a binary classification task to determine the order of two sentences. This task is similar to pre-training objectives considered in recent works. We use Roc Stories (*RocBSO*; Mostafazadeh et al., 2016) corpus for constructing the data set for this task.
12. Natural language inference (NLI) is a three-way classification task of predicting whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral). The Multi-Genre Natural Language Inference (*MNLI*; Williams et al., 2018) corpus consists of sentence pairs from different sources (transcribed speech, fiction, and government report) annotated for textual entailment.
13. Multi-choice QA is a reading comprehension task wherein given a passage and question, models need to pick up the right option out of the provided ones. Khot et al. (2018) cast the multiple-choice science exam questions into an NLI structure to convert them to a binary classification task. We use the *SciTAIL* (Khot et al., 2018) data set released by them for our experimentation.
14. Implicit discourse relation classification is a common task of identifying discourse relations between two text spans or arguments. The Penn Discourse Treebank 3.0 (*PDTB3L1*; Prasad et al., 2019; Kim et al., 2020) contains a hierarchical annotation scheme (top-level senses, fine-grained level-2 senses) and we use top-level senses comprising of four labels (expansion, comparison, contingency, temporal) for our experimentation.
15. Emotion detection is a classification task of detecting the emotions from a given text snippet. We use *Emotion* data set (Saravia et al., 2018) which contains Twitter messages with six emotions: anger, fear, joy, love, sadness, and surprise.

2.3.2 Question Answering

We use three question answering datasets: SQuAD v1.1 (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017) and QuAC (Choi et al., 2018). TriviaQA has two sections, Web and Wikipedia, which we consider as separate datasets. We process the datasets to follow the same setup as d'Autume et al. (2019). Our processed dataset includes SQuAD with $90,000$ training and $10,000$ validation examples, TriviaQA (Web) with $76,000$ training and $10,000$ validation examples, TriviaQA (Wikipedia) with $60,000$ training and $8,000$ validation examples and QuAC with $80,000$ training and $7,000$ validation examples.

2.3.3 Image Classification

We perform our experiments on 5-dataset-CV (diverse) and Split CIFAR-50/ Split CIFAR-100 (homogeneous).

5-dataset-CV consists of five diverse 10-way image classification tasks - CIFAR-10 (Krizhevsky and Hinton, 2009), MNIST (LeCun, 1998), Fashion-MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011), and notMNIST (Bulatov, 2011). It is one of the largest datasets used for lifelong learning experiments (Ebrahimi et al., 2020) with overall 180.9k train examples (see Table 2.2 for task-specific statistics).

Split CIFAR-50 takes the first 50 classes of the CIFAR-100 image classification dataset (Krizhevsky and Hinton, 2009) and randomly splits them into five homogeneous 10-way classification tasks. Each task contains $5,000/1,000$ (train/test) examples. We built this dataset as a homogeneous counterpart to 5-dataset-CV by mimicking its task structure (10 classes/task) and the number of tasks. Further, we note that Split CIFAR-50 (10 classes/ task) is more challenging than Split MNIST/ CIFAR-10 (2 classes/ task) because there are more classes per task. Therefore, in this work, we prefer Split CIFAR-50 over MNIST/CIFAR-10 for our experimentation.

Split CIFAR-100 splits the CIFAR-100 dataset into 20 disjoint 5-way classification tasks, with each task containing $2,500/500$ (train/test) examples. Due to the large number of tasks in this dataset, it is regarded as one of the most challenging and realistic CV benchmarks for lifelong learning (Chaudhry et al., 2018b).

2.4 Prominent Baselines for Lifelong Learning

Various approaches have been developed for lifelong learning, tailored to different scenarios and system requirements (Sodhani et al., 2022). These approaches fall into several categories: (i) parameter-based regularization, (ii) episodic memory-based, (iii) test-time adaptation using episodic memory, and (iv) optimization-based approaches. In the subsequent sections, we will delve into notable methods within each category, which we consider in this thesis for comparing our approaches.

Dataset	|Train|	|Val|	|Test|	|L|
MNIST	51,000	9,000	10,000	10
notMNIST	15,526	2,739	459	10
Fashion-MNIST	9,574	1,689	1,874	10
CIFAR10	42,500	7,500	10,000	10
SVHN	62,269	10,988	26,032	10

Table 2.2: **5-dataset-CV** statistics. |Train|, |Val|, and |Test| denote the number of examples in the train, validation, and test splits, respectively. |L| denotes the number of classes for each task.

2.4.1 Parameter-Based Regularization Approaches

Parameter-based regularization methods aim to prevent catastrophic forgetting in neural networks by introducing a penalty term to the task-specific objective, without the need for retaining samples from prior tasks. While a basic approach involves minimizing the drift between parameters w_1 and w_2 post task 2 training, this can significantly limit network capacity during sequential training. To address this, advanced techniques selectively apply regularization, allowing flexibility for some parameters while constraining others based on their relevance to past tasks. The crux of these techniques lies in efficiently measuring the importance of each parameter.

Elastic Weight Consolidation (EWC; Kirkpatrick et al., 2017) is the canonical approach in this category. It tackles forgetting by restricting updates to parameters important for previously learned tasks, as determined by the Fisher information matrix F diagonal terms. Specifically, when training on a current task t, EWC tries to minimize the distance from previous optimal parameters that correspond to each task $1 \leq \tau < t$, as follows

$$\hat{R}_t(f_w; \mathcal{D}^t_{train}, w^*_1, \cdots, w^*_{t-1}) = \underbrace{\frac{1}{n_t} \sum_{(x^t_i, y^t_i) \in \mathcal{D}^t_{train}} \ell\big(f_w(x^t_i), y^t_i\big)}_{\text{empirical risk of the current task } t} + \underbrace{\alpha \sum_{\tau=1}^{t-1} \sum_{j=1}^{N} F^\tau_j (w_j - w^*_{\tau,j})^2}_{\text{penalty term for prior tasks, } 1 \leq \tau < t}, \tag{2.7}$$

where N is the total number of parameters ($|w|$), α is the regularization strength, w^*_τ are the optimal parameters after training on the prior task τ, and F^τ is the Fisher information matrix calculated as the point estimate at the end of training on task τ. Formally, the F^τ_j is defined as:

$$F^\tau_j = \mathbb{E}\big[\nabla_{w_j} \log f_w(x)^2 |_{w^*_{\tau,j}}\big]. \tag{2.8}$$

EWC necessitates retaining all Fisher matrices from previous tasks, denoted as F^τ, and the optimal parameters w^*_τ for those tasks. This results in a memory complexity that scales linearly with the number of tasks $\mathcal{T}$, specifically $\mathcal{O}(N\mathcal{T})$. EWC can be employed to mitigate forgetting across all lifelong learning scenarios.

Online EWC Schwarz et al. (2018) introduces a modified version of EWC, known as Online EWC, designed for situations that require continuous learning from an online stream of data, especially in scenarios where task identifiers are unknown. In this variation, an overall Fisher is computed online instead of storing the Fisher information for all previous tasks. Let w^*_{t-1} and $\tilde{F}^{t-1}$ represent the optimal parameters and overall Fisher after learning task $t-1$. After learning task t, the overall Fisher is updated as follows

$$\tilde{F}^t = \gamma \tilde{F}^{t-1} + F^t, \tag{2.9}$$

where γ serves as a decay parameter, diminishing the importance of parameters associated with previous tasks. This modification eliminates the need for the task identifier during training and, due to γ, gracefully forgets older tasks by down-weighting their importance values. Graceful forgetting is crucial in lifelong learning to create space for new tasks, preventing capacity issues discussed in Kirkpatrick et al. (2017) when the model reaches its capacity limit.

2.4.2 Episodic Memory-Based Approaches: Data-Based Regularization

In the preceding section, we explored parameter-based regularization methods, which aim to prevent significant deviations of model parameters from their initialization while optimizing the current task loss. In contrast, this section delves into data-based regularization methods, aiming to induce a similar effect as parameter-based approaches. We specifically concentrate on episodic memory-based approaches in lifelong learning. Episodic memory ($\mathcal{M} = \cup_{\tau<t}\mathcal{M}_\tau$) retains a subset of observed examples from all prior tasks $\tau < t$ and is employed to mitigate forgetting of previous tasks while learning the current task t. Different approaches within this category differ in terms of how episodic memory is employed—either to derive implicit constraints on current task gradients (Chaudhry et al., 2019) or explicit constraints on current task gradients (Lopez-Paz et al., 2017; Chaudhry et al., 2018b).

Episodic Replay (ER; Chaudhry et al., 2019) The most basic strategy for utilizing episodic memory involves replaying examples, denoted as $\mathcal{M}_\tau$, from each previously learned task τ while learning the current task t. Formally, the loss functions are defined as follows

$$L_{\text{TASK}}(w; \mathcal{D}^t_{train}) = \frac{1}{n_t} \sum_{(x^t_i, y^t_i) \in \mathcal{D}^t_{train}} \ell(f_w(x^t_i), y^t_i), \tag{2.10}$$

$$L_{\text{REP}}(w; \mathcal{M}) = \frac{1}{n_{re}} \sum_{\tau=1}^{t-1} \sum_{(x^\tau_i, y^\tau_i) \in \mathcal{M}_\tau} \ell(f_w(x^\tau_i), y^\tau_i), \tag{2.11}$$

where L_{TASK} represents the loss on samples from the current task t, L_{REP} represents the replay loss on samples corresponding to prior tasks, as sampled from the episodic memory $\mathcal{M}$, and $n_{re} = |\mathcal{M}|$. For ER, the joint optimization problem covering both the current task loss and replay loss is defined as

$$w^*_t = \arg\min_w \quad \alpha_1 L_{\text{TASK}}(w; \mathcal{D}^t_{train};) + \alpha_2 L_{\text{REP}}(w; \mathcal{M}). \tag{2.12}$$

Typically, in Equation 2.12, α_1 and α_2 are assigned the value of 1. Chaudhry et al. (2019) demonstrates that this straightforward strategy outperforms other sophisticated algorithms in lifelong learning performance.

Averaged Gradient Episodic Memory (A-GEM; Chaudhry et al., 2018b) imposes the constraint that the average replay loss, computed over the previous tasks using the episodic memory, does not increase

$$\min_{w} L_{\text{TASK}}(w; \mathcal{D}^t_{train}) \quad \text{s.t.} \quad L_{\text{REP}}(w; \mathcal{M}) \leq L_{\text{REP}}(w^*_{t-1}; \mathcal{M}) \quad \text{where } \mathcal{M} = \cup_{\tau<t}\mathcal{M}_\tau\,. \tag{2.13}$$

The optimization problem corresponding to Equation 2.13 is defined as

$$\min_{\tilde{g}} \frac{1}{2}||g - \tilde{g}||_2^2 \quad \text{s.t.} \quad \tilde{g}^\top g_{REP} \geq 0. \tag{2.14}$$

Here, g_{REP} denotes a gradient computed using a batch of replay examples, randomly sampled from the episodic memory $\mathcal{M}$ across all previously encountered tasks. Rather than solving Equation 2.13 using a quadratic program, it can be solved using only the inner product between the gradients of L_{TASK} (g) and L_{REP} (g_{REP}). When the gradient of the current task g violates the constraint, the projected gradient $\tilde{g}$ is computed as follows

$$\tilde{g} = g - \frac{g^\top g_{REP}}{g_{REP}^\top g_{REP}} g_{REP}\,. \tag{2.15}$$

For the joint optimization problem in Equation 2.12, A-GEM algorithm reduces to setting the importance weights α_1 and α_2 as follows

$$\alpha_1 = 1, \quad \alpha_2 = \mathbf{I}_{\langle g, g_{REP}\rangle \leq 0}\left(-\frac{g^\top g_{REP}}{g_{REP}^\top g_{REP}}\right), \tag{2.16}$$

where $\mathbf{I}_C$ is the indicator function which evaluates to 1 if C holds and otherwise to 0. Note that both ER and A-GEM are applicable in all three lifelong learning scenarios, as well as in situations where one needs to learn from an online or offline stream of data.

2.4.3 Test-Time Adaptation-Based Approaches

In the previous section, we focus on approaches that use episodic memory during training. However, lifelong learning is continuous, and there might not be a clear boundary between training and evaluation. Consequently, it is reasonable to consider access to episodic memory even during evaluation, and we will explore approaches addressing this aspect in this section.

Memory-based Parameter Adaptation (MbPA; Sprechmann et al., 2018) employs two components: a slow-learning parametric neural network and a fast-adapting non-parametric episodic memory. The parametric network f_w consists of an embedding network h_{emb} and a task network g_θ, where $p(y|x; emb, \theta) = f_w(x) = g_\theta(h_{emb}(x))$. Unlike previous memory-based methods,

instances are stored as key-value pairs in the episodic memory $\mathcal{M}_t = (h_i, v_i)$ during training, where $h_i = h_{emb}(x_i)$ and $v_i = y_i$. Also, the memory is not utilized for training beyond populating it with observed examples.

During inference, the episodic memory facilitates instance-based (local) adaptation of the parametric network, hence, the approach is called Memory-based Parameter Adaptation (MbPA). Concretely, the encoding of the current input $h_{emb}(x)$ is used to retrieve k nearest neighbors from the episodic memory, $\mathcal{M}_k = \{(h_i, v_i, s_i)\}_{i=1}^{k}$, where the weight s_i gauges the proximity of the i-th example to $h_{emb}(x)$ using the kernel function

$$\text{kern}(h_i, h_{emb}(x)) = \frac{1}{\epsilon + ||h_i - h_{emb}(x)||_2^2}\,. \tag{2.17}$$

The local adaptation involves adjusting the task network parameters θ for examples x to minimize the weighted average negative likelihood over the retrieved k neighbors, with the update rule defined as

$$\tilde{\theta}^{x_i} = \arg\min_{\theta} -\sum_{i=1}^{k} s_i \log p(y_i|x_i; emb, \theta)\,. \tag{2.18}$$

Improved Memory-based Parameter Adaptation (MbPA++; de Masson D'Autume et al., 2019) notices that in MbPA the episodic memory contains keys from the embedding network at different points during the training. They argue that this results in the embedding network drifting over time, and the key of the test examples is closer to that of the recently seen examples. To circumvent this issue, they suggest freezing the embedding network. Specifically, MbPA++ consists of three main components: (i) a predictor network f_w, (ii) a key network g_ϕ, and (iii) a memory module $\mathcal{M}$. The goal is to train f_w to generalize on examples across all tasks as in Equation 2.5.

To learn a generic representation, MbPA++ utilizes any state-of-the-art text encoder, such as BERT (Devlin et al., 2019), to initialize both predictor network f_w and key network g_ϕ. At each time step, the model receives a training example $(\boldsymbol{x}_i^t, y_i^t) \in \mathcal{D}_{train}^t$ and updates parameter w by optimizing the task loss

$$L_{\text{TASK}}(w; x_i^t, y_i^t) = \ell(f_w(x_i^t), y_i^t). \tag{2.19}$$

To determine if the training example should be added to the memory module $\mathcal{M}$, a Bernoulli random variable is drawn with pre-set probability, which is used to control the memory size.

For experience rehearsal (or episodic replay), a subset $\mathcal{S}$ of $\mathcal{M}$ is randomly selected, based on a set ratio of replay examples to learning new examples (i.e. revisit n_{re} examples for every n_{tr} training examples). To avoid catastrophic forgetting, the model then updates the following replay loss to adapt w towards prior tasks (similar to Equation 2.11)

$$L_{\text{REP}}(w; \mathcal{S}) = \frac{1}{n_{re}} \sum_{(x,y)\in\mathcal{S}} \ell(f_w(x), y). \tag{2.20}$$

At inference time, the key network g_ϕ, which is fixed during training, is used to encode example inputs as keys to obtain the k nearest neighbor context $\mathcal{N}_{x_i}$ of the i-th testing example x_i.

n_l local adaptation gradient updates are then performed to achieve task-specific finetuning using the following objective

$$L_{\text{LA}}(\tilde{w}^{x_i}; w, \mathcal{N}_{x_i}) = \frac{1}{k} \sum_{(x,y)\in\mathcal{N}_{x_i}} \ell(f_{\tilde{w}^{x_i}}(x), y) + \lambda_l \|\tilde{w}^{x_i} - w\|_2^2, \tag{2.21}$$

where λ_l is a hyperparameter. The predictor network $f_{\tilde{w}^{x_i}}$ is then used to output the final prediction for the i-th testing example x_i.

2.4.4 Optimization-Based Approaches

Another avenue in lifelong learning research explores the optimization perspective of the problem. Mirzadeh et al. (2020b) demonstrated that the geometric nature of the local minima reached by the learning algorithm influences the extent of catastrophic forgetting. Let w_i^* denote the minimum achieved after sequential training on the i-th task, and $L_j(w_i)$ represent the loss of the j-th task with parameters w_i. Forgetting F_1 of the first task after training on the second task is defined as

$$F_1 = L_1(w_2^*) - L_1(w_1^*). \tag{2.22}$$

According to the second-order Taylor expansion of $L_1(w_2^*)$ around w_1^*

$$L_1(w_2^*) \approx L_1(w_1^*) + \frac{1}{2}(w_2^* - w_1^*)^\top \nabla^2 L_1(w_1^*)(w_2^* - w_1^*). \tag{2.23}$$

By using the above approximation to $L_1(w_2^*)$, Mirzadeh et al. (2020b) derived an upper bound for F_1 in terms of the maximum eigenvalue (λ_1^{max}) of $\nabla^2 L_1(w_1^*)$

$$\begin{aligned} F_1 &\approx \frac{1}{2}(w_2^* - w_1^*)^\top \nabla^2 L_1(w_1^*)(w_2^* - w_1^*), \\ &\leq \frac{1}{2}\lambda_1^{max} ||w_2^* - w_1^*||_2^2. \end{aligned} \tag{2.24}$$

.

Stable SGD (Mirzadeh et al., 2020b) As per the bound in Equation 2.24, a smaller λ_1^{max} corresponds to less forgetting. λ_1^{max} is utilized to characterize the width of local minima, where small values indicate flat minima and large values indicate sharp or narrow minima (Hochreiter and Schmidhuber, 1997). Building on this analysis, Mirzadeh et al. (2020b) proposes the Stable SGD strategy, which modifies the training process by adjusting hyperparameters such as learning rate, batch size, learning rate decay, and dropout regularization. The objective is to introduce inherent noise in the stochastic gradients, promoting convergence to wide regions within the loss landscape, thereby reducing forgetting during continual learning. This approach does not necessitate access to examples from previous tasks. Nonetheless, conducting a hyperparameter sweep in continual learning poses challenges as it is ill-defined, therefore, implementing this method across different lifelong learning scenarios is challenging.

Mode Connectivity SGD (MC-SGD; Mirzadeh et al., 2021) approach modifies the training dynamics to constrain continual minima within a low-loss valley connected to all previous minima. Let w^*_{t-1} represent the optimal solution after sequential training on $t-1$ previous tasks, and $\tilde{w}_t$ denote the minimum achieved after sequential training on the t-th task. The following objective is employed to find the optimal solution enforcing linear mode connectivity to both w^*_{t-1} and $\tilde{w}_t$:

$$w^*_t = \arg\min_w \sum_\alpha \left[L_{\text{REP}}(w^*_{t-1} + \alpha(w - w^*_{t-1}); \mathcal{M}) + L_{\text{TASK}}(\tilde{w}_t + \alpha(w - \tilde{w}_t); \mathcal{D}^t_{train}) \right], \quad (2.25)$$

where L_{TASK} represents the loss on samples from the current task t, L_{REP} represents the replay loss on samples corresponding to prior tasks, as sampled from the episodic memory $\mathcal{M}$, and α parameterizes the line connecting w to w^*_{t-1} and $\tilde{w}_t$ respectively in the first and second term. MC-SGD can be seen as a fusion of both regularization and replay-based methods in continual learning, using a small replay buffer to approximate a low-loss path for previous tasks. Thus, this approach requires the presence of previous task data in the form of episodic memory.

Part I

Model: Architecture & Initialization

Chapter 3

Initialization: Role of Pre-training in Lifelong Learning

3.1 Overview

As discussed in Section 1.1, the role of pre-trained initializations in lifelong learning settings has been under-explored. In contemporary work, Hu et al. (2021) uses pre-trained models as *feature extractors* (i.e., pre-trained weights are frozen) for task-incremental learning. The pre-trained weights are frozen in this setting, so the model undergoes no forgetting. In contrast, *fine-tuning* pre-trained weights update the pre-trained model parameters and are susceptible to severe forgetting. Pre-training followed by fine-tuning is typically the most accurate and common Transfer Learning (TL) paradigm (Peters et al., 2019) and the one we consider in this chapter. To the best of our knowledge, there is no prior work systematically analyzing the role of pre-trained initialization on catastrophic forgetting in lifelong learning scenarios.

Figure 3.1 shows that simply changing the network initialization to pre-trained initialization significantly reduces forgetting on the first task during sequential training on five tasks. This observation motivates us to ask—**Does pre-training implicitly alleviate catastrophic forgetting, and if so, why?** To answer this question, we conduct a systematic study on existing CV and NLP benchmarks and observe that pre-training indeed leads to less forgetting. We also investigate the effect of the type of pre-trained initialization by analyzing the extent to which different pre-trained Transformer language model variants (Sanh et al., 2019; Devlin et al., 2019; Liu et al., 2019b; Raffel et al., 2020) undergo forgetting, observing that increasing the capacity of the model and diversity of the pre-training corpus play an essential role in alleviating forgetting. To further stress-test these models under realistic scenarios, we introduce a dataset with 15 diverse NLP tasks (see Section 2.3.1 for more details) and observe considerable forgetting on this dataset.

We hypothesize that pre-trained weights already have a good inductive bias to alleviate forgetting implicitly. To explain this behavior, we build upon two separate observations — Hao et al. (2019); Neyshabur et al. (2020) show that in the context of TL, pre-trained weights lead to a flat loss basin when fine-tuning on a single task. Mirzadeh et al. (2020b) argues that the geometric properties of the local minima for each task play a vital role in forgetting and suggests modifying the hyper-parameters (learning rate decay, batch size, dropout) to promote flat minima.

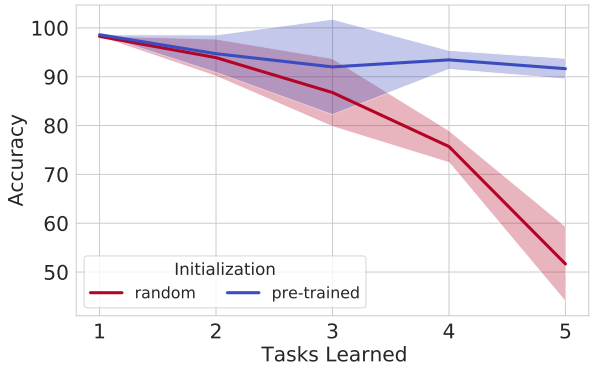


Figure 3.1: Pre-trained and randomly initialized DistilBERT on Split YahooQA dataset. Performance of the first task visualized over sequential learning of tasks (averaged over 5 runs). Both models start with approximately equal average task accuracy, but pre-trained initialization leads to significantly less forgetting.

To verify the above hypothesis, we analyze the loss landscape of the first task as the model is trained sequentially on subsequent tasks. For pre-trained initializations, we see that minima obtained after training on a sequence of tasks remain in the relatively low loss basin of the first task compared to the random initialization. Furthermore, linearly interpolating loss across sequentially trained task minima confirms that models initialized with pre-trained weights undergo a more gradual change in loss than random initialization. These observations hint at the flatness of the minima reached in the case of pre-trained initialized models. To measure the flatness of the loss landscape, we compute a sharpness metric (Keskar et al., 2017) and verify that pre-trained weights result in flatter basins compared to random weights during sequential training. These analyses help us showcase that continual training from pre-trained weights induces wide task minima, therefore, implicitly alleviating forgetting. Our main contributions can be summarized as follows

- We observe that initializing models with pre-trained weights results in less forgetting than random weights despite achieving higher performance on each task. We bolster this observation with a systematic study validating that this behavior persists across applications (NLP, CV) and prominent approaches: elastic weight consolidation (EWC; Kirkpatrick et al., 2017), average gradient episodic memory (A-GEM; Chaudhry et al., 2018b), and episodic replay (ER; Chaudhry et al., 2019).
- To understand the role of varying pre-trained initializations, we analyze a suite of pre-trained Transformer language models and showcase that model capacity and diversity of the pre-training corpus play a role in alleviating forgetting. We also show that sequential training on diverse tasks is still challenging for pre-trained initialized models by introducing a new, more challenging benchmark for lifelong learning in NLP consisting of 15 diverse NLP tasks.
- We hypothesize and verify empirically that pre-trained models alleviate forgetting as they have an implicit bias towards wider task minima. The effect of these wider minima is that changes in weights from learning subsequent tasks result in a smaller change to the current task loss, which helps reduce forgetting.

3.2 Experimental Setup

3.2.1 Problem Formulation

We consider a task-incremental learning scenario where we receive a continuum of data from different tasks sequentially: $(x_1, y_1, t_1), \cdots, (x_i, y_i, t_i), \cdots$. Each triplet (x_i, y_i, t_i) consists of a task descriptor $t_i \in \mathcal{T}$, input data $x_i \in \mathcal{D}_{t_i}$, target labels $y_i \in \mathcal{Y}_{t_i}$ and satisfies $(x_i, y_i) \overset{iid}{\sim} \mathcal{P}_{t_i}(X, Y)$. We consider an explicit task descriptor t_i because the same input x_i can appear in multiple tasks but with different labels. For example, given an article about a new technological breakthrough, we could use it to train a model for extracting entities and another model for classifying the article into different categories such as science, health, or technology. In this case, the input text x_i is the same article but the explicit task descriptor t_i is different, one for entity extraction and another for classification. Given the observed data, our goal is to learn a predictor $f_w : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{Y}$ where we want to evaluate test pairs (x, t) from previously observed tasks (backward transfer) and the current task at any time during continual learning of our model. Additionally, we assume an offline training setting, allowing for multiple training passes over a task before transitioning to the next task in a sequence.

3.2.2 Benchmarks and Task Sequences

We perform extensive experiments on widely adopted task-incremental learning benchmarks across both NLP and CV domains - **Split YahooQA** (Section 2.3.1) and **Split CIFAR-50/ Split CIFAR-100** (Section 2.3.3). These benchmarks help us evaluate our method to be consistent with the literature (Chaudhry et al., 2019; Ebrahimi et al., 2020). Most existing works consider Split MNIST, Split CIFAR-10, and Split CIFAR-100 benchmarks, which are homogeneous; different tasks in these benchmarks share the same underlying domain. Given the generic nature of the pre-trained initialization, we want to investigate forgetting when subjected to a sequence of diverse tasks. Therefore, we also consider benchmarks spanning diverse NLP and CV tasks and evaluate on **5-dataset-NLP/ 15-dataset-NLP** (Section 2.3.1) and **5-dataset-CV** (Section 2.3.3).

One of the desiderata of a lifelong learning method is to be robust to different task sequences as task ordering is unknown beforehand. Hence, we run all our experiments with five random task sequences and report average performance.

For **Split CIFAR-50/ Split CIFAR-100**, the task sequences were generated by randomly sampling classes without replacement for each task. Thus, the sequences were different for every random seed, and each method was trained and tested on the identical five sequences.

For **Split YahooQA**, we created five tasks by using disjoint groups of consecutive classes (e.g. $\{0, 1\}, \{2, 3\} \ldots$). These tasks were then randomly ordered for each task sequence, and each method was trained and tested using the identical five random sequences.

For **5-dataset-CV**, we randomly select the following dataset orders

Seq1 SVHN→notMNIST→Fashion-MNIST→CIFAR-10→MNIST

Seq2 SVHN→MNIST→notMNIST→Fashion-MNIST→CIFAR-10

Seq3 CIFAR-10→SVHN→notMNIST→Fashion-MNIST→MNIST

Seq4 notMNIST→Fashion-MNIST→CIFAR-10→SVHN→MNIST

Seq5 CIFAR-10→MNIST→notMNIST→SVHN→Fashion-MNIST

For **5-dataset-NLP**, we randomly select the following dataset orders (the first four are consistent with de Masson D'Autume et al., 2019):

Seq1 Yelp→AGNews→DBPedia→Amazon→YahooQA

Seq2 DBPedia→YahooQA→AGNews→Amazon→Yelp

Seq3 Yelp→YahooQA→Amazon→DBpedia→AGNews

Seq4 AGNews→Yelp→Amazon→YahooQA→DBpedia

Seq5 YahooQA→Yelp→DBPedia→AGNews→Amazon

For **15-dataset-NLP**, we consider following five random task orderings:

Seq1 Decomp→BoolQ→AAC→Yelp→DISCONN8→SST-2→QQP→YahooQA→QNLI →RocBSO→MNLI→SciTAIL→CoLA→PDTB2L1→Emotion

Seq2 CoLA→QQP→MNLI→QNLI→Emotion→SST-2→BoolQ→Decomp→AAC→SciTAIL →RocBSO→Yelp→PDTB2L1→YahooQA→DISCONN8

Seq3 SciTAIL→BoolQ→SST-2→AAC→DISCONN8→YahooQA→QNLI→RocBSO→PDTB2L1 →Emotion→Decomp→MNLI→QQP→CoLA→Yelp

Seq4 DISCONN8→QNLI→CoLA→YahooQA→AAC→SciTAIL→PDTB2L1→Emotion →Decomp→RocBSO→QQP→Yelp→MNLI→BoolQ→SST-2

Seq5 Emotion→SST-2→RocBSO→YahooQA→AAC→MNLI→CoLA→DISCONN8→QQP →QNLI→Decomp→PDTB2L1→SciTAIL→Yelp→BoolQ

3.2.3 Baselines

We evaluate our approach against prominent methods in the task-incremental lifelong learning scenario (Chaudhry et al., 2019; Mirzadeh et al., 2020b, 2021). Specifically, we compare with EWC from the parameter-based regularization approaches (Section 2.4.1), ER and A-GEM from the episodic memory-based approaches (Section 2.4.2), and Stable SGD and MC-SGD from the optimization-based approaches (Section 2.4.4). Excluded from the comparison are test-time adaptation-based approaches (Section 2.4.3), as a task identifier is available. Additionally, we omit architecture-based approaches that dynamically expand the network based on new tasks (Sodhani et al., 2022), as these approaches do not undergo forgetting, and this chapter primarily focuses on the role of pre-training in forgetting. Brief descriptions of the relevant baselines are provided below, with detailed information available in Section 2.4.

- **Finetune** (FT): The model is sequentially fine-tuned on each task without additional learning constraints.

- **Elastic Weight Consolidation** (EWC; Kirkpatrick et al., 2017): A parameter-based regularization approach that tries to mitigate forgetting by restricting learning to parameters important to previously learned tasks, as measured by the Fisher information matrix.
- **Averaged Gradient Episodic Memory** (A-GEM; Chaudhry et al., 2018b): A data-based regularization approach that augments the base model with an episodic memory module that retains examples from the previously seen tasks, and during training, uses these stored examples to enforce a constraint on the gradients, ensuring that the model does not forget previously learned tasks.
- **Episodic replay** (ER; Chaudhry et al., 2019): This approach involves the use of a replay buffer to store and replay past experiences during training. This enables the model to revisit and learn from previously seen examples, mitigating catastrophic forgetting and enhancing its ability to retain knowledge across multiple tasks. Following Chaudhry et al. (2019), we retain one example per task per class and randomly select examples for storage. Prabhu et al. (2020); Hussain et al. (2021) show that the straightforward ER method outperforms all of the previous methods under realistic task-incremental learning settings, and therefore, we compare our approach mainly with ER.
- **Stable SGD** (Mirzadeh et al., 2020b): This method alters the training process by adjusting hyperparameters such as learning rate, batch size, learning rate decay, and dropout regularization (see Appendix A.1 for hyperparameter sweep). The goal is to introduce inherent noise in the stochastic gradients, resulting in convergence to wide regions within the loss landscape, which in turn leads to reduced forgetting during continual learning.
- **Mode Connectivity SGD** (MC-SGD; Mirzadeh et al., 2021): This approach restricts the minima of continual learning within a region of low loss by all previous minima. MC-SGD can be viewed as a combination of both regularization and replay-based methods in continual learning as it uses a small replay buffer to approximate a low-loss path for previous tasks.

3.3 Does pre-training implicitly alleviate forgetting?

Having defined the formal problem setup, evaluation metrics, and methods for alleviating the forgetting phenomenon, in this section, we conduct experiments to tease apart the role of pre-training for lifelong learning. We are interested in answering the following questions

(Q1) How much does pre-training help in alleviating forgetting?

(Q2) Do pre-trained models undergo similar forgetting on diverse and homogeneous tasks?

(Q3) How do different pre-trained initializations affect forgetting?

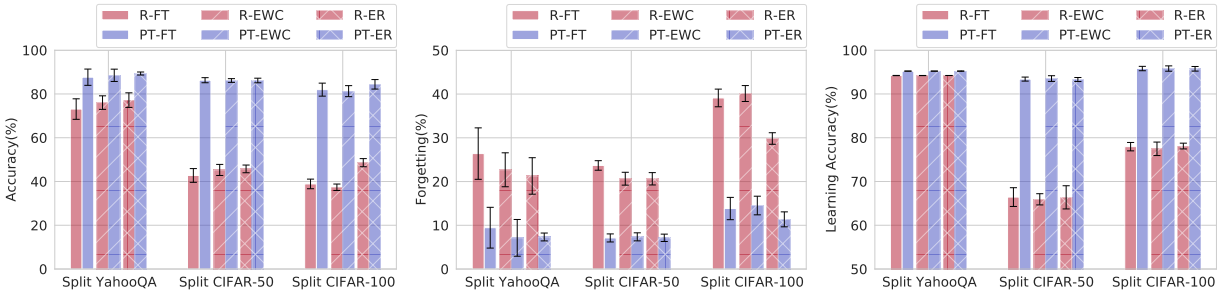
Experimental design. To answer these questions convincingly, we conduct experiments on the above-discussed CV and NLP benchmarks. We utilize the DistilBERT$_{\text{base}}$ (Sanh et al., 2019) architecture for text classification and the ResNet-18 (He et al., 2016) architecture for image classification. To isolate the effect of pre-training, we consider two variants for each of these architectures: pre-trained models (**DistilBERT-PT/ResNet-18-PT**) and randomly initialized

models (**DistilBERT-R/ResNet-18-R**). For our study, we need to ensure that there are as few confounding factors as possible. Therefore, we keep all other hyperparameters the same and vary only the initialization (for more details refer to Appendix A.1). To measure the severity of forgetting, ideally, we want sufficient training samples so that both a pre-trained model or a randomly initialized model (of the same capacity) achieves similar learning accuracy on each task. To control for this behavior we either select a large training corpus whenever available (e.g., 279k examples/task for Split YahooQA) or run our experiments for multiple epochs (5 epochs for CV benchmarks).

ImageNet pre-training corpus. For a fair comparison between pre-trained and randomly initialized models, we explicitly control for and remove the overlap between pre-training and downstream tasks. Publicly available ResNet models are pre-trained on ImageNet that overlaps with CIFAR-100 in terms of class labels. Therefore, we make sure that the subset of the ImageNet corpus we use does not have any visually and semantically overlapping classes with the CIFAR-100 dataset. We use the publicly available (Abdelsalam et al., 2021) two-level class hierarchies for ImageNet, where semantically and visually similar labels are grouped under one super-category. We iterate over all CIFAR-100 labels and drop the complete super-category from ImageNet corresponding to each of these labels. For example, CIFAR-100 contains a *castle* class and we have a *building* super-category in ImageNet that contains *castle, palace, monastery, church, etc.*. We remove all building-related labels from our pre-training dataset. In total, we remove 267 classes and pre-train the **ResNet-18-PT** model on the remaining subset of the ImageNet dataset.

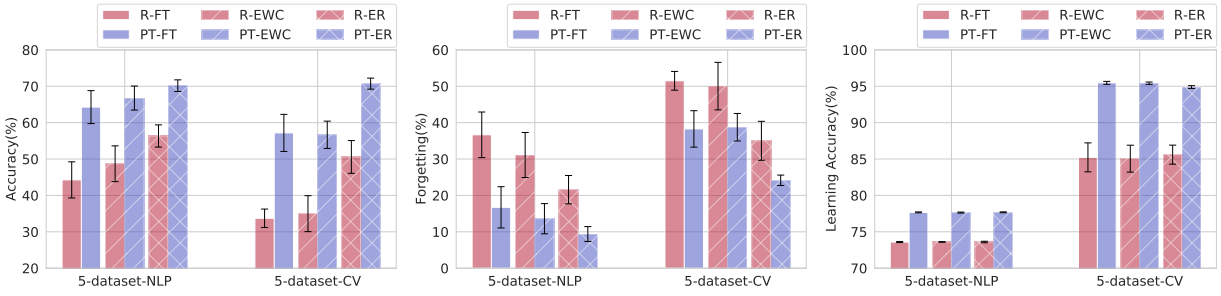
3.3.1 How much does pre-training help in alleviating forgetting?

From Figures 3.2b and 3.3b, we see that pre-trained models (ResNet-18-PT, DistilBERT-PT) undergo significantly less forgetting in comparison to models with random initialization (ResNet-18-R, DistilBERT-R). This trend holds across all three methods — FT, EWC, and ER. For text datasets (Split YahooQA, 5-dataset-NLP), we see that both models have comparable learning accuracy (see Figures 3.2c and 3.3c) and significantly less forgetting for DistilBERT-PT. This can be completely attributed to the pre-trained initialization. On 5-dataset-CV, ResNet-18-PT undergoes less forgetting (38.28) compared to ResNet-18-R (51.51). Specifically, despite task accuracy starting at a higher base for ResNet-18-PT, the forgetting value is still lower compared to ResNet-18-R models. Additionally, this effect also holds when considering a sequentially finetuned pre-trained model (with no additional regularization to alleviate forgetting) to a randomly initialized model trained with LL methods. For example, on 5-dataset-NLP, sequentially finetuning DistilBERT-PT undergoes less forgetting (16.73) compared to the competitive ER method (21.58) when applied to DistillBERT-R. This raises an interesting research direction—**explicitly focusing on learning generic features apart from just concentrating on the forgetting aspect**.



(a) Accuracy (↑) (b) Forgetting (↓) (c) Learning Accuracy (↑)

Figure 3.2: Comparing performance on **homogeneous tasks** (Split YahooQA/ CIFAR-50/ CIFAR-100) across initialization (R: random, PT: pre-trained) and methods (FT: finetune, EWC: elastic weight consolidation, ER: episodic replay) after training on the last task. ↑ indicates higher is better, ↓ indicates lower is better. All metrics are averaged over 5 random task sequences (see Equation 2.6). We observe that pre-trained models undergo significantly less forgetting in comparison to randomly initialized models.

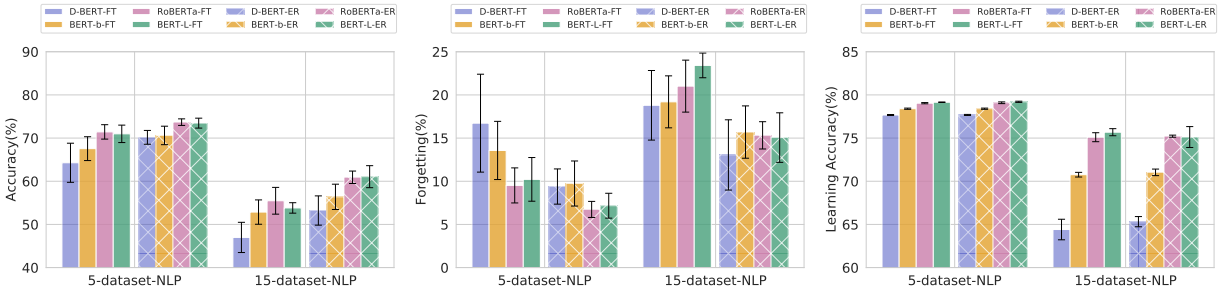


(a) Accuracy (↑) (b) Forgetting (↓) (c) Learning Accuracy (↑)

Figure 3.3: Comparing performance on **diverse tasks** (5-dataset-NLP/ CV) across initialization (R: random, PT: pre-trained) and methods (FT: finetune, EWC: elastic weight consolidation, ER: episodic replay) after training on the last task. ↑ indicates higher is better, ↓ indicates lower is better. All metrics are averaged over 5 random task sequences (see Equation 2.6). In comparison to homogeneous tasks (see Figure 3.2b), we observe that pre-trained models are more susceptible to forgetting when exposed to a diverse sequence of tasks.

3.3.2 Do pre-trained models undergo similar forgetting on diverse and homogeneous tasks?

From Figure 3.2b, we see that ResNet-18-PT does not undergo a significant amount of forgetting when sequentially fine-tuned on Split CIFAR-50, and Split CIFAR-100 (homogeneous tasks). On Split CIFAR-50, forgetting is around 7% absolute points. Surprisingly, the competitive ER method also undergoes a similar amount of forgetting, thereby raising a question about the applicability of these datasets when studying forgetting in the context of the pre-trained models. It may be possible to manually cluster tasks based upon semantic closeness, rendering severe interference to make these benchmarks more challenging (Ramasesh et al., 2021).



(a) Accuracy (↑) (b) Forgetting (↓) (c) Learning Accuracy (↑)

Figure 3.4: Comparing performance on diverse tasks (5-dataset-NLP/ 15-dataset-NLP) across different **pre-trained Transformer models** (D-BERT: DistilBERT, BERT-b: BERT-base, RoBERTa: RoBERTa-base, BERT-L: BERT-Large) and methods (FT: finetune, ER: episodic replay) after training on the last task. ↑ indicates higher is better, ↓ indicates lower is better. All metrics are averaged over 5 random task sequences (see Equation 2.6). Overall, we observe that models pre-trained on diverse and larger corpora (RoBERTa-base) undergo less forgetting.

Given the generic nature of the pre-trained initialization, we ask—what happens when we train the model sequentially on diverse tasks? To answer this question, we conduct experiments on 5-dataset-CV and 5-dataset-NLP. From Figure 3.3b, **we empirically observe that pre-trained models are more susceptible to forgetting when exposed to diverse tasks in comparison to homogeneous tasks**. Particularly, DistilBERT-PT/ResNet-18-PT undergoes a $16.73/38.28\%$ absolute points drop in accuracy when trained on 5-dataset-NLP/5-dataset-CV. Figures 3.2a and 3.3a report average accuracy after training on the last task. We report task-specific results for 5-dataset-NLP/5-dataset-CV in Appendix A.2.

3.3.3 How do different pre-trained initializations affect forgetting?

To examine the impact of varying pre-trained initialization on forgetting, we evaluate different pre-trained Transformer models, DistilBERT (Sanh et al., 2019), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), on text classification tasks. From the previous subsection, we observe that pre-trained models are relatively more susceptible to forgetting on LL of diverse tasks. In response, we conduct a thorough investigation on the 5-dataset-NLP. From Figure 3.4, we observe that when keeping the pre-training corpora the same and increasing the capacity of the model – DistilBERT (66M), BERT-base (110M), and BERT-large (336M) – we observe that larger models undergo less forgetting on sequential finetuning of diverse tasks. Further, to understand the impact of the diversity of the pre-training corpora, we compare the BERT-base (110M) with the RoBERTa-base (125M). We observe that the RoBERTa-base model performs far superior to the BERT-base, thus hinting at the essential role of size and diversity of pre-training corpora in implicitly alleviating forgetting. We acknowledge other significant distinctions between BERT-base and RoBERTa-base, including training objectives, dynamic masking strategy, and the use of large mini-batches, Liu et al. (2019b) confirm the importance of data size and diversity in pre-training. A more thorough examination of the role of these factors in forgetting is left for

future investigations. To stress-test these models, we experiment with the 15-dataset-NLP. We observe that by increasing the number of tasks in the sequence, pre-trained models undergo severe forgetting. Surprisingly, the RoBERTa-base model outperforms BERT-Large despite having many fewer parameters. Empirically, we infer that **data size and diversity of pre-training corpora plays a vital role in easing forgetting during lifelong learning of diverse tasks.**

3.4 Exploring the Loss Landscape

To better understand how pre-training reduces forgetting, we perform experiments analyzing where models are situated in the loss landscape after training on each task. We denote model parameters after training on task k as w_k. If we define forgetting as the increase in loss for a given task during training (instead of a decrease in accuracy), Mirzadeh et al. (2020b) shows that the forgetting can actually be bounded by:

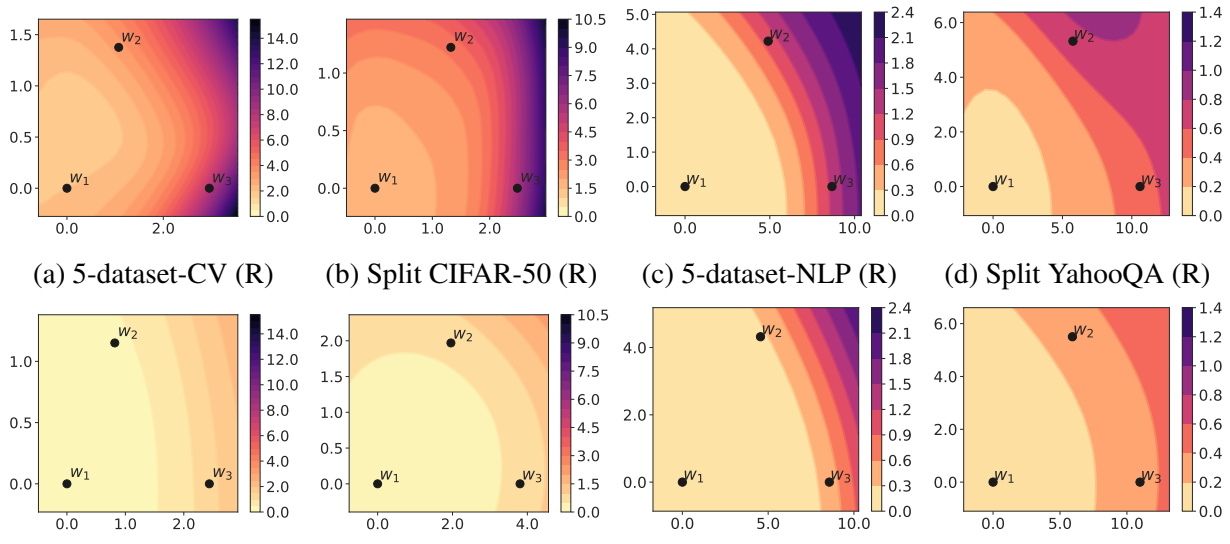
$$L_1(w_2) - L_1(w_1) \approx \frac{1}{2}\Delta w^\top \nabla^2 L_1(w_1)\Delta w \leq \frac{1}{2}\lambda_1^{max}\|\Delta w\|^2 \tag{3.1}$$

where $L_1(w)$ represents the loss on task 1 with parameters w, $\Delta w = w_2 - w_1$, and λ_1^{max} is the largest eigenvalue of $\nabla^2 L_1(w_1)$. The magnitude of the eigenvalues of $\nabla^2 L_1(w)$ can be used to characterize the curvature of the loss function (Keskar et al., 2017), and thus λ_1^{max} can be thought of as a proxy for the flatness of the loss function (lower is flatter). From Equation 3.1, we can see that the flatter the minima, the less forgetting occurs in the model.

We hypothesize that the one explanation of improvements from pre-training shown in the previous section might be because pre-training leads to a more favorable loss landscape. Specifically, pre-training results in wider/flatter minima for each task. The effect of these wider minima is that the change in weights from learning on future tasks results in a gradual change of the current task loss, thereby reducing forgetting. We verify this idea in two parts. First, we use loss contours and then linearly interpolate between sequential minima to show that the flat loss basins lead to smaller changes in the loss. Next, we compute a sharpness metric to show that pre-training indeed leads to flat loss basins. All models analyzed in this section are sequentially trained using the finetune method.

3.4.1 Loss Contour

To better understand the changes in task loss during continual training on a sequence of tasks, we utilize loss contours, which involve linearly interpolating between the continual learning minima. In order to construct a 2D loss contour, we require three points to define two basis vectors. Specifically, we train on tasks 1, 2, and 3 sequentially, resulting in minima represented by w_1, w_2, and w_3, respectively. We designate w_1 as our reference point (0, 0) and calculate $\vec{u}$=$\mathrm{w}_3-\mathrm{w}_1$ as one basis vector (representing the x-axis in our plots). Additionally, we compute an orthogonal projection $\vec{v}$ of $\mathrm{w}_2-\mathrm{w}_1$ onto $\vec{u}$, which serves as the second basis vector (representing the y-axis in our plots). Consequently, for any coordinate (x, y) within the 2D loss contour, we derive the corresponding model parameters as $\mathrm{w}(x, y) = \mathrm{w}_1 + x.\mathrm{u} + y.\mathrm{v}$ and compute the validation



(a) 5-dataset-CV (R) (b) Split CIFAR-50 (R) (c) 5-dataset-NLP (R) (d) Split YahooQA (R)

(e) 5-dataset-CV (PT) (f) Split CIFAR-50 (PT) (g) 5-dataset-NLP (PT) (h) Split YahooQA (PT)

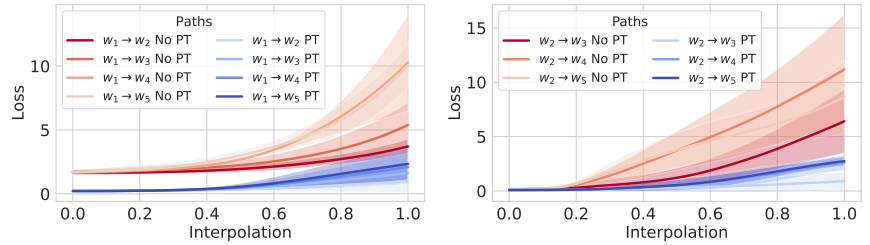
Figure 3.5: **Loss contours** for task 1 where $\mathbf{w}_1$**,** $\mathbf{w}_2$**,** $\mathbf{w}_3$ are minima obtained after sequential training on tasks 1, 2, and 3, respectively. The top row visualizes loss contours for randomly initialized models (R), and the bottom row visualizes loss contours for pre-trained models (PT).

loss for the task under consideration. In this setup, the distance between any two points on the loss contour reflects the Euclidean distance between the corresponding model parameters.

In Figure 3.5, we visualize loss contours for the first task across different data set-specific task sequences. For every contour, we plot minima (w_1, w_2, w_3) of the model after continual training on three tasks (T_1, T_2, T_3). As the model is trained continuously on a sequence of tasks, the pre-training initialized model remains largely at the same loss level (for task 1) as compared to the randomly initialized model, despite drifting a comparable distance away from the original model (w_1). For example, in the loss contour for the pre-trained model on 5-dataset-CV (Figure 3.5e), we observe that the model after training on task 2 (w_2) remains at the same loss level as after training on task 1 (w_1) and slightly higher loss level after training on task 3 (w_3). For the randomly initialized model (Figure 3.5a), the Euclidean distance between the model parameter vectors is approximately the same as for the pre-trained model, but the differences in task 1 loss levels are significantly higher. We visualize more instances of loss contours over different task sequences in Appendix A.3. In summary, **we observe that pre-trained models consistently lead to wider loss basins across different data sets (NLP and CV domains), model architectures (ResNet and Transformer), and task sequences (5 random orderings)**.

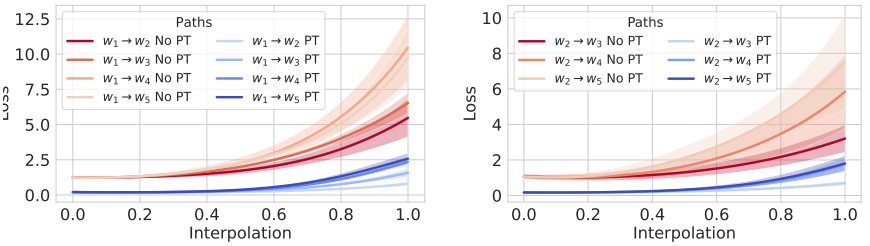
3.4.2 Linear Model Interpolation

Ideally, to ease forgetting during sequential training of tasks, loss on previous tasks should undergo minimal change. This desideratum would be satisfied if a previous task minimum lands in a flat loss region and subsequent task minima also remain in that flat loss region. To probe this behavior,



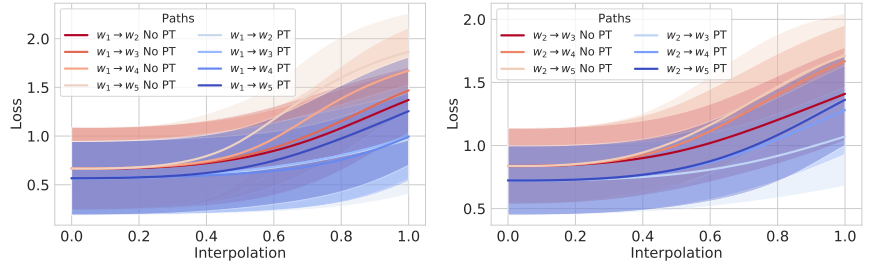
(a) 5-dataset-CV (task 1)

(b) 5-dataset-CV (task 2)



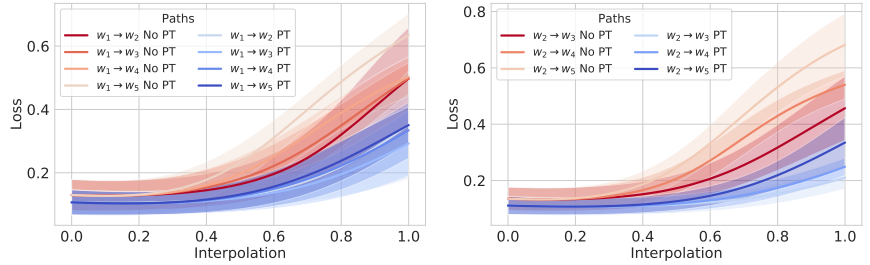
(c) Split CIFAR-50 (task 1)

(d) Split CIFAR-50 (task 2)



(e) 5-dataset-NLP (task 1)

(f) 5-dataset-NLP (task 2)



(g) Split YahooQA (task 1)

(h) Split YahooQA (task 2)

Figure 3.6: **Linear model interpolation** plots for different datasets. The plots for pre-training initialized (PT) models are shown in hues of blue, and the randomly initialized (No PT) models are shown in hues of red. We linearly interpolate between the **task 1/ task 2** minimum ($\mathbf{w}_1$/$\mathbf{w}_2$) to the subsequent task mimimum ($\mathbf{w}_i \to \mathbf{w}_j$, $j > i$), tracking the loss in the process. The loss landscape is generally flatter along these paths for pre-trained initialized models compared to randomly initialized models.

	ResNet-18-R	ResNet-18-PT	ResNet-18-R	ResNet-18-PT
	$\epsilon = 5 \times 10^{-4}$		$\epsilon = 10^{-3}$	
5-dataset-CV	$2.1_{0.6}$	$0.1_{0.0}$	$5.7_{1.6}$	$0.2_{0.0}$
Split CIFAR-50	$2.3_{0.7}$	$0.2_{0.1}$	$6.1_{1.5}$	$0.4_{0.1}$
Split CIFAR-100	$2.3_{0.6}$	$0.1_{0.0}$	$5.9_{1.3}$	$0.2_{0.1}$
	DistilBERT-R	DistilBERT-PT	DistilBERT-R	DistilBERT-PT
	$\epsilon = 5 \times 10^{-5}$		$\epsilon = 10^{-4}$	
5-dataset-NLP	$32.7_{1.2}$	$28.3_{1.2}$	$213.6_{11.5}$	$129.0_{10.5}$
Split YahooQA	$10.4_{0.4}$	$8.8_{0.4}$	$53.2_{7.0}$	$43.0_{4.2}$

Table 3.1: **Average sharpness value** (lower value corresponds to flat loss basin) of task minima. ResNet-18-PT/DistilBERT-PT has lower average sharpness than ResNet-18-R/DistilBERT-R. Pre-training reduces the sharpness of minima for each task in training by order of magnitude.

we linearly interpolate between w_1 (w_2) and subsequent task minima, tracking the (validation) loss on task 1 (task 2). This probe can be interpreted as viewing a slice of the loss contours in Figure 3.5 along the linear trajectory connecting w_1 (w_2) to a subsequent minimum. In Figure 3.6, we visualize the results from linear interpolation with the first row for task 1 and the second row for task 2. The plots for pre-training initialized models are shown in hues of blue, and the randomly initialized models are shown in hues of red. These plots illustrate that the pre-training initialized models experience a much more gradual increase in loss compared to the randomly initialized models, even when interpolating to minima after training on several tasks. Moreover, these results hold for task 2 as well, thereby reinforcing that **pre-training initialized models lead to flat minima even for subsequent tasks in a sequence**.

3.4.3 Sharpness Metric

As discussed earlier, the flatness of the minima can be estimated based upon the magnitude of eigenvalues of $\nabla^2 L(w)$. However, computing these eigenvalues is computationally expensive. Therefore, Keskar et al. (2017) introduces a sensitivity measure, termed *sharpness metric*, as a computationally feasible alternative to computing eigenvalues. The sharpness metric estimates the flatness by computing the maximum value of the loss function L in a constrained neighborhood around the minima. Given that the maximization process can be inaccurate, Keskar et al. (2017) suggests performing maximization in a random subspace $\mathbb{R}^p$ of the entire parameter space $\mathbb{R}^n$, specified by a projection matrix $A \in \mathbb{R}^{n \times p}$. For our experiments, we randomly sample our matrix A and set $p = 100$ as in Keskar et al. (2017). The neighborhood maximization region (C_ϵ) is defined as

$$C_\epsilon = \{z \in \mathbb{R}^p : -\epsilon(|(A^+ w)_i| + 1) \leq z_i \leq \epsilon(|(A^+ w)_i| + 1); \forall i \in \{1 \dots p\}\}, \tag{3.2}$$

where A^+ is the pseudo inverse of A, w is the parameter vector and ϵ is a hyperparameter controlling the size of the neighborhood. Formally, the sharpness metric is defined as follows

$$\phi_{w,L} := \frac{(\max_{z \in C_\epsilon} L(w + Az)) - L(w)}{1 + L(w)} \times 100, \tag{3.3}$$

where $L(w)$ denotes the loss function with parameters w. According to Keskar et al. (2017), the sharpness metric is closely related to the spectrum $\nabla^2 L(w)$, therefore acts as a proxy measure for λ_1^{max} in Equation 3.1. After training on each task, we evaluate the minimum reached by the model for its sharpness (alternatively flatness) using the test dataset. We average the sharpness values across all tasks in a given task sequence and report the mean and standard deviation across five random task orderings. In Appendix A.1, we provide implementation details about the sharpness metric.

In Table 3.1, we report sharpness values for ResNet-18 on 5-dataset-CV, Split CIFAR-50/CIFAR-100 for $\epsilon \in \{5e^{-4}, 1e^{-3}\}$ and DistilBERT on 5-dataset-NLP and Split YahooQA datasets for $\epsilon \in \{5e^{-5}, 1e^{-4}\}$. We see that for all datasets, **the average sharpness value for the pre-trained initialized models is significantly lower than for the randomly initialized models, validating the relative flatness of the task minima in the case of pre-trained models**.

3.5 Related Work

In this section, we establish the connections to significant related works in the field.

Transfer learning from generic pre-trained models has sparked significant advancements in machine learning (Zhuang et al., 2021). Initially emerging in CV with the introduction of the ImageNet data set (Deng et al., 2009), the practice of transfer learning has also undergone its own "ImageNet revolution" in NLP. Notably, large models pre-trained on self-supervised tasks have exhibited remarkable performance across various language-related tasks (Peters et al., 2018; Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019b; Raffel et al., 2020). The NIPS-95 workshop on "Learning to Learn" (Pan and Yang, 2009) initially motivated transfer learning as a means to facilitate lifelong learning. Our work revisits it in light of recent progress in transfer learning.

Lifelong learning approaches focus on the idea of mitigating the forgetting phenomenon and can be grouped into four categories: (1) *regularization-based* approaches either augment the loss function with extra penalty terms preventing important parameters learned on previous tasks from significantly deviating while training on the new task (Kirkpatrick et al., 2017; Zenke et al., 2017; Chaudhry et al., 2018a; Aljundi et al., 2018) and/ or enforce distillation-based penalty (Li and Hoiem, 2017; Dhar et al., 2019); (2) *memory-based* approaches that augment the model with episodic memory for sparse experience replay of previous task examples, either during training (Lopez-Paz et al., 2017; Chaudhry et al., 2018b, 2019; Guo et al., 2020) and/ or during inference (Rebuffi et al., 2017; de Masson D'Autume et al., 2019; Mehta et al., 2020); some approaches learn generative model to simulate replay buffers (Shin et al., 2017; Sun et al., 2020); (3) *optimization-based* approaches that either maintain a space of gradient directions for previous tasks and projects the gradients of a new task in a direction orthogonal to that space, ensuring less disruption of older tasks (Farajtabar et al., 2020) or modify training regimes by specific hyperparameter configurations yielding wider minima and reducing the forgetting (Mirzadeh et al., 2020b); and (4) *architecture-based* approaches that dynamically expand the network based upon new tasks (Rusu et al., 2016; Aljundi et al., 2017; Yoon et al., 2018; Sodhani et al., 2020), or

iteratively learn mask (Mallya et al., 2018; Serra et al., 2018) or prunes networks (Mallya and Lazebnik, 2018) for new tasks. By formulation, these approaches do not undergo forgetting and hence we consider regularization and episodic memory-based approaches for our analysis.

Meta-learning involves the development of models capable of learning over time and has been employed in various studies focusing on lifelong learning (Riemer et al., 2019; Finn et al., 2019; Javed and White, 2019; Mehta et al., 2020; Gupta et al., 2020). Notably, Caccia et al. (2020) propose a two-phase continual learning scenario, where the initial phase entails pre-training utilizing MAML (Finn et al., 2017), followed by continual deployment with task revisiting. They emphasize that deploying agents without any pre-training in lifelong learning scenarios would be impractical, a sentiment shared by other studies (Lomonaco et al., 2020). While some of these works employ pre-trained initializations, the comprehensive examination of the impact of pre-training in lifelong learning remains largely unexplored and is the focus of our work.

Optimization and loss landscape works have explored the relationship between pre-training and wider optima in single-task generalization (Hao et al., 2019; Neyshabur et al., 2020), as well as the effects of larger batch sizes on sharper minima and poorer generalization in single-task learning (Keskar et al., 2017). Additionally, Mirzadeh et al. (2021) compare minima resulting from multitask learning and continual learning, establishing that minima from continual learning are linearly connected to optimal sequential multitask minima but not to each other, leading to forgetting. While these studies examine the connection between pre-training and flat minima in single-task scenarios or the relationship between flat minima and model generalization, we extend this research by investigating whether the benefits of pre-training persist during sequential training on multiple tasks. We explore the effects of pre-training on loss landscapes throughout lifelong learning and validate a hypothesis that elucidates the role of pre-training in lifelong learning.

3.6 Discussion

In this chapter, we study the effect of pre-training on lifelong learning across various datasets and modalities. We find that compared to models with random initialization, models with pre-trained initialization undergo significantly less forgetting. Specifically, despite task accuracy starting at a higher base for pre-trained models, the absolute forgetting value is still lower for pre-trained models. This effect even holds when comparing a sequentially finetuned pre-trained model (with no additional regularization to improve performance or reduce forgetting) to a randomly initialized model trained with state-of-the-art lifelong learning methods. One takeaway from our study is that lifelong learning methods should focus on learning generic initialization instead of simply alleviating catastrophic forgetting, as generic initialization appears to undergo minimal forgetting. We also explore the effect of different pre-trained models on performance in an NLP setting and find that while increased model capacity helps up to a certain point when considering shorter task sequences when considering longer and more diverse task sequences, the quality and the size of the pre-trained corpora matter more than model capacity.

To explain this effect, we perform several analyses of the loss landscapes produced during training for both random and pre-train initialized models. We find that the minima found by the

pre-trained models at the end of training on each task are significantly flatter than those created by the randomly initialized models. This means that even when pre-trained models drift away from the original flat task minima, the task loss does not increase significantly, which results in less forgetting.

Based on our findings and results, explicitly seeking flat loss basins may further reduce forgetting compared to existing methods. Another potential line of work could explore where the multitask minima are related to the pre-trained initialization, as Mirzadeh et al. (2021) shows that the sequential multitask minima are linear mode connected to minima after each task in lifelong learning. The flatness of the minima for every model starting from a pre-trained initialization could suggest a way to regularize the sequential training process with the pre-trained initialization such that the model ends up at the multitask minima.

In this chapter, we presented a novel set of 15 diverse text classification tasks (15-dataset-NLP) to investigate lifelong learning with a larger number of tasks. The introduced suite proves to be more challenging than the previous benchmark consisting of 5 datasets (see Figure 3.4). Future research could explore using ExMix (Aribandi et al., 2021), a comprehensive collection of 107 supervised tasks across diverse domains, as a potentially more challenging and realistic benchmark for lifelong language learning.

Chapter 4

Architecture: Dynamic Parameter-Efficient Experts for Lifelong Learning

4.1 Overview

We considered fixed-capacity models for instantiating lifelong learning systems in Chapter 3. However, there is a tug-of-war dynamic (due to the stability-plasticity dilemma) between learning new knowledge (plasticity) and preventing forgetting old knowledge (stability) given fixed capacity (Sodhani et al., 2022). Motivated by this and compartmentalization of complex systems (Simon, 1962; Meunier et al., 2009), recent works in task-incremental learning scenario (Rusu et al., 2016; Aljundi et al., 2017) learn a separate expert (Jacobs et al., 1991; Shazeer et al., 2017) for each task and devise a gating mechanism to forward the data through a relevant expert. So for every new task, a separate expert is added to the system, thereby growing the network's capacity while addressing the above dilemma. However, for these approaches, the number of experts (and thereby the network's capacity) grows linearly with tasks and hence cannot scale to a scenario where we have many tasks. Moreover, this approach only applies to scenarios with explicit task boundaries, as different experts correspond to different tasks. However, there are application scenarios where we have many tasks without explicit task delimitation. For instance, consider a question-answering system that answers the questions given candidate passages. This system is a well-suited candidate for lifelong learning; the system may encounter passages from evolving (or unseen) domains requiring continuous learning (Liska et al., 2022). Moreover, at inference time the system can see examples from any domain, necessitating knowledge-sharing between the underlying experts and learning them without any domain identifier.

Recent advances in parameter-efficient fine-tuning techniques (Mangrulkar et al., 2022) have enabled the training of domain-specific experts, such as prompts (Wang et al., 2022a), without significant additional storage requirements. However, these methods necessitate an inference-time routing mechanism for expert selection and often rely on the implicit domain discriminative capabilities of pre-trained models (Aharoni and Goldberg, 2020). However, this straightforward method might not consistently excel in distinguishing domains across various datasets (see Section 4.4.3). On the other hand, prior works have also explored generative methods to create synthetic examples for experience replay (Shin et al., 2017; Sun et al., 2020; Qin and Joty, 2022).

However, these methods have generally fallen short compared to their counterparts that retain actual data for experience replay (refer to Sun et al. (2020) and results in Section 4.4.2), possibly due in part to the inclusion of noise in the form of low-quality synthetic examples. An intriguing question arises—Rather than employing noisy synthetic examples for generative replay, can they serve a more effective purpose in domain discrimination? More specifically, **can we leverage synthetically generated samples to develop an inference-time routing mechanism**?

In this chapter, we propose Generate to Discriminate for Expert Routing (G2D*f*ER; Byun et al., 2023)[1], a continual learning method that leverages modern generative language models (Raffel et al., 2020) to generate per-domain synthetic examples for purposes of domain discrimination (rather than generative replay). Concretely, for each new domain t, our approach involves: (i) training a domain-specific expert model, (ii) training a domain-specific generative model to generate synthetic examples, and (iii) training the domain discriminator using generated samples from both the previous and current domains to enable domain prediction. At inference time, we pass samples through our domain discriminator, which routes each sample to a determined domain-specific expert. In summary, we contribute the following

- Generate to Discriminate for Expert Routing (G2D*f*ER), an expert routing method for domain-incremental learning using generative models for domain discrimination.
- Analysis demonstrating that training a domain identifier substantially outperforms augmenting training data for downstream tasks with the same synthetic samples (i.e., the generative replay approach).
- Experiments demonstrating that our approach is competitive with previous test-time adaptation-based state-of-the-art approaches and outperforms by 6.2 F_1 in scenarios characterized by stringent constraints on data sharing.

4.2 Experimental Setup

4.2.1 Problem Formulation

We consider a domain-incremental learning from a sequence of $\mathcal{T}$ domains, $\mathcal{D}_1 \rightarrow \cdots \rightarrow \mathcal{D}_{\mathcal{T}}$, where $\mathcal{D}^t_{train} = \{x^t_i, y^t_i\}^{n_t}_{i=0}$ represents a dataset corresponding to domain t, sampled from an underlying distribution $P_t(\mathcal{X}, \mathcal{Y})$. $x^t_i \in \mathcal{X}$ is the i-th text passage, $y^t_i \in \mathcal{Y}$ is its label and n_t is the total number of training samples for domain t. As discussed in Section 2.1.2, in the domain-incremental scenario, the marginal or conditional distributions over $\mathcal{X}$ and $\mathcal{Y}$ can change across domains, i.e., $P_i(\mathcal{X}) \neq P_j(\mathcal{X})$ and $P_i(\mathcal{Y}|\mathcal{X}) \neq P_j(\mathcal{Y}|\mathcal{X})$, $\forall i, j \in \{1, \cdots, \mathcal{T}\}$, while the label space $\mathcal{Y}$ remains fixed across all domains. Additionally, following de Masson D'Autume et al. (2019), we assume that we are continuously learning from an online stream of data. During training, the domain identifier is assumed to be available for each example but it is unknown during inference time. The goal is to learn a predictor $f_w : \mathcal{X} \rightarrow \mathcal{Y}$, parameterized by $w \in \mathbb{R}^P$, to minimize the average expected risk across all $\mathcal{T}$ domains (see Equation 2.5). To demonstrate the model's learning behavior over the sequence of domains and analyze catastrophic forgetting of

[1] In our paper (Byun et al., 2023), we term our approach Generate to Discriminate (G2D), while in the context of this thesis, we specifically denote it as Generate to Discriminator for Expert Routing (G2D*f*ER).

the previously seen domains, we evaluate the model after training on a specific domain t using the test dataset of that domain, $\mathcal{D}^t_{test} \sim P_t(\mathcal{X}, \mathcal{Y})$, and from past domains, $\mathcal{D}^\tau_{test} \sim P_\tau(\mathcal{X}, \mathcal{Y}), \forall \tau \in [1, \ldots, t-1]$. For overall performance evaluation, we compute the average accuracy (A_t) metric as defined in Section 2.2.

4.2.2 Benchmark and Domain sequences

We evaluate our method on the standard domain-incremental question-answering benchmark consisting of four datasets: SQuAD v1.1, TriviaQA (Web), TriviaQA (Wikipedia), and QuAC (refer to Section 2.3.2 for more details). We compute F_1 score for question answering task and evaluate the model at the end of all domains, i.e., we compute A_4. Next, we run all our experiments with four random domain sequences and report average performance. We consider the following domain orderings for our experimentation (consistent with de Masson D'Autume et al., 2019)

Seq1 QuAC→TrWeb→TrWik→SQuAD

Seq2 SQuAD→TrWik→QuAC→TrWeb

Seq3 TrWeb→TrWik→SQuAD→QuAC

Seq4 TrWik→QuAC→TrWeb→SQuAD

4.2.3 Baselines

We compare our method with state-of-the-art continual learning methods that address the domain incremental scenario. Specifically, we compare with Online EWC from the parameter-based regularization approaches (Section 2.4.1), ER and A-GEM with their online variants from the episodic memory-based approaches (Section 2.4.2), and MbPA++ from test-time adaptation-based approaches (Section 2.4.3). Excluded from the comparison are optimization-based approaches (Section 2.4.4), as these have been introduced for task-incremental learning scenarios. We provide chapter-specific details of the aforementioned baselines below, with a detailed description available in Section 2.4.

- **Finetune** (FT): The model is sequentially fine-tuned on each domain without additional learning constraints.
- **Online Elastic Weight Consolidation** (EWC; Schwarz et al., 2018): An online parameter-based regularization approach that tries to mitigate forgetting by restricting learning to parameters important to previously learned domains.
- **Episodic Replay** (ER; Chaudhry et al., 2019): This approach involves the use of a replay buffer to store and replay past experiences during training. This enables the model to revisit and learn from previously seen examples, mitigating catastrophic forgetting and enhancing its ability to retain knowledge across multiple domains. Following de Masson D'Autume et al. (2019), we retain 1% of the total examples in the replay buffer and perform experience replay by sampling 100 examples from the memory and performing a gradient update after every 10,000 training steps, which gives us a 1% replay rate.
- **Improved Memory-based Parameter Adaptation** (MbPA++; de Masson D'Autume et al., 2019): This approach uses a replay buffer to perform sparse experience replay during

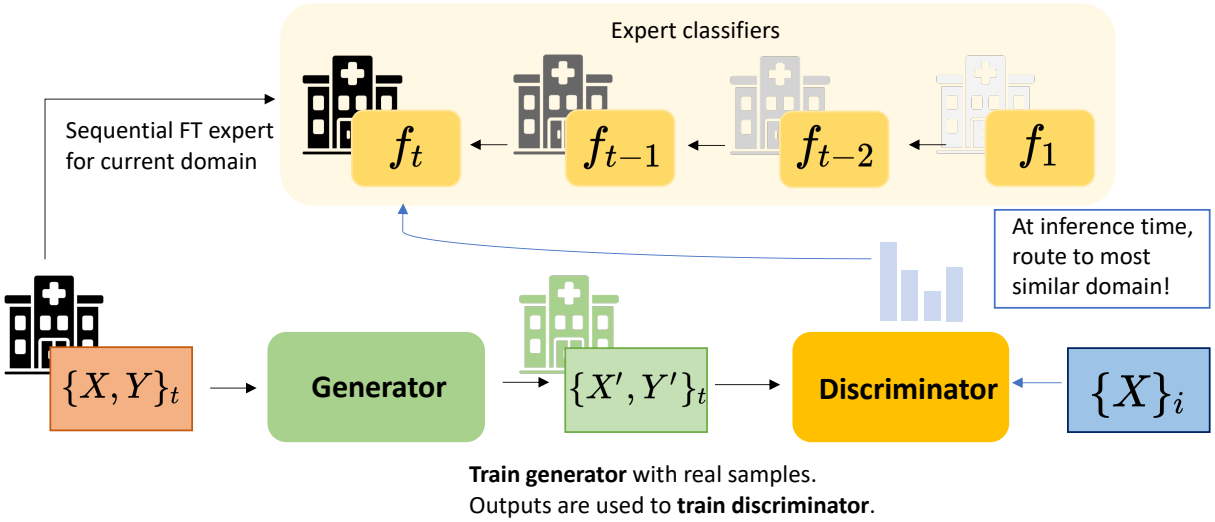


Figure 4.1: Generate to Discriminate for Expert Routing (G2D*f*ER); At train time, we i) fine-tune the generator and expert model and ii) train a domain discriminator on synthetic samples produced by our generator. At inference time, based on our discriminator's prediction, we route test samples to the corresponding expert.

training and domain-specific test-time adaptation during inference. Similar to ER, 1% samples are retained in the buffer and the replay rate is set to 1%.

- **Meta-MbPA** (Mehta et al., 2020) On top of MbPA++, this approach trains the model to attain a more suitable initialization for test-time adaptation and represents the prior state-of-the-art performance on the considered question-answering benchmark. Note that this approach is presented in this thesis and a detailed description of it can be found in Chapter 6.
- **Generative replay** (GR): In this approach, a language model is used to generate synthetic samples for the replay buffer (more details in the following section).
- **Oracle Routing**: In this theoretical method, we assume access to the domain identifier during inference and direct the example to the corresponding domain-specific expert. We include this approach for comparison with our learned routing against ground-truth routing.
- **MTL**: In this approach, access to real data from all domains is allowed at every domain step. This is equivalent to training on the union of all existing data and can be viewed as an upper bound on performance when there is no significant negative transfer between domains.

4.3 Generate to Discriminate for Expert Routing (G2D*f*ER)

In this section, we discuss Generate to Discriminate for Expert Routing (G2D*f*ER), our proposed continual learning method that leverages modern generative models (language models) — to generate per-domain synthetic examples for purposes of domain discrimination (rather than generative replay). We then leverage this discriminator at inference time to route examples to

relevant experts. Figure 4.1 schematically depicts the workflow of our proposed approach.

4.3.1 Generation of synthetic samples for domain discriminator

At each new domain t, we fine-tune a generative language model G with training samples from the current domain, $\mathcal{D}^t_{train} = \{(x^t_i, y^t_i)\}^{n_t}_{i=0}$. Then, we generate synthetic samples $\mathcal{M}_t$ from G. Given synthetically generated data from all seen domains, i.e., $1 \leq \tau \leq t$, we train a domain discriminator D_{w^t} on the union of the synthetically generated samples $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_t$, for domain identity prediction (i.e., t-way classification). More formally, we construct a dataset of $\bigcup^t_{i=1}\{(x, i)|x \in \mathcal{M}_i\}$. In essence, our domain discriminator learns to predict domain membership, or route samples to their corresponding or most similar domains.

Implementation details. For our generator, we use the prompt tuning (Lester et al., 2021) to learn the parameter-efficient models. We use the pre-trained T5-Large v1.1 checkpoint adapted for prompt tuning as the backbone (Raffel et al., 2020) and the prompt embeddings are initialized randomly. We set the prompt length to 400 tokens which accounts for 819K trainable parameters, i.e., around 0.1% in comparison to 784M frozen T5-Large parameters. We input a special token into the model and conditionally generate a document content, question, and answer, all separated by the special tokens. We employ the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 1.0, a warmup ratio of 0.01, and linearly decay the learning rate over 5 epochs, use a batch size of 8 and set weight decay to $1e^{-5}$. Our maximum sequence length is set to 512, and we truncate the document content after tokenizing the question-answer pair. During the generation process, we provide multiple text prompts. We use the following text prompts to conditional generate synthetic samples — "Generate article, question and answer.", "Generate context, question and answer.", "Generate answers by copying from the generated article.", "Generate factual questions from the generated article." During generation, we use ancestral sampling, which selects the next token randomly based on the model's probability distribution over the entire vocabulary, thereby reducing the risk of repetition. We generate samples with a minimum length of 50 tokens and a maximum of 1,000 tokens, retaining only those samples that contain exactly one question-answer pair with the answer included in the generated document content.

For training our domain discriminator, we use low-rank adaptation (LoRA; Hu et al., 2022) and freeze a pre-trained BERT-base (Devlin et al., 2019) backbone. BERT-base has 12 Transformer layers, 12 self-attention heads, and 768 hidden dimensions (110M parameters). We train our discriminator for 5 epochs using the Adam optimizer and the learning rate is set to 5e-4. For LoRA, we set the dimension of the low-rank matrices (r) to 32 which gives us around 1.2M trainable parameters (1.07% of full BERT-base 110M parameters).

4.3.2 Expert models

At each new domain t, given $\mathcal{D}^t_{train}$, we sequentially fine-tune our model f_{w^t} as the expert on domain t, and add f_{w^t} to our existing list of experts $\{f_{w^1}, ..., f_{w^{t-1}}\}$. At inference time, we use our domain discriminator to predict the most likely domain, and the test sample is routed to the corresponding expert classifier for our class prediction.

Method	Average $F_1(\uparrow)$
ER	61.2 ± 1.8
MbPA++†	61.9 ± 0.2
Meta-MbPA†	64.9 ± 0.3
FT	56.6 ± 5.7
EWC	55.9 ± 3.7
Generative Replay	58.5 ± 3.7
G2D*f*ER (PEFT)	64.7 ± 0.2
G2D*f*ER (Full FT)	66.6 ± 0.7
Oracle Routing	67.6 ± 0.1
Upper Bound (MTL)	68.6 ± 0.0

Table 4.1: Results on Question Answering benchmark. Comparing performance in terms of average F_1 across methods after training on the last domain (averaged over four random domain sequences). ↑ indicates higher is better, † denotes results obtained from Mehta et al., 2020. ER, MbPA++ and Meta-MbPA use a buffer size of 1% actual samples. PEFT denotes parameter-efficient fine-tuning and Full FT denotes full fine-tuning. Our G2D*f*ER (Full FT) approach outperforms all baselines and the G2D*f*ER (PEFT) approach demonstrates competitive performance, even in the absence of retaining the actual samples, when compared to state-of-the-art methods.

Implementation details. We apply the same LoRA technique and hyperparameters as those used for the domain discriminator to train our domain expert. We train our expert for 3 epochs using Adam optimizer and the learning rate is set to 5e-4. In the case of a full fine-tuning scenario, for training our expert, we mainly set hyper-parameters as mentioned in de Masson D'Autume et al. (2019). We use Adam as our optimizer, set dropout to 0.1, and the base learning rate to 3e-5. We use a training batch of size 8, set the maximum total input sequence length after tokenization to 384 and to deal with longer documents we set document stride to 128. We also set the maximum question length to 64. The hyper-parameters for baseline methods are set as described in Mehta et al. (2020). For ER (or Generative Replay) we retain (or sample) 1% examples which account for around 6,000 examples across all four considered domains.

4.4 Experiments

4.4.1 How much does G2D*f*ER help in alleviating forgetting?

In Table 4.1, we report our results on the question-answering benchmark. Our approach, with full fine-tuning, G2D*f*ER Full FT (66.6), outperforms both ER (61.2) and test-time adaptation techniques such as MbPA++ (61.9) and Meta-MbPA (64.9) and G2D*f*ER with parameter-efficient LoRA (64.7) is competitive with Meta-MbPA. It is important to note that all of these baseline methods retain actual samples in their buffers (1% of total samples). These findings underscore

Dataset	Fields	Generated sample
		Correct
SQuAD	*Context:*	During the late 19th and early 20th centuries, the city's trade sector expanded greatly, and through the 20th century, more than half of its residents worked toward higher status in the military. Other industries included industry, commerce, public administration, and medicine. Its largest sector was public services - police, fire services, and healthcare and was the nation's third largest.
	Question:	What is one of the industry sectors that were the biggest?
	Answer:	public services
		Incorrect
	Context:	In the United Kingdom there is a general agreement between the government and the private sector in principle that both private and publicly funded institutions of higher education constitute university colleges. Further, there is a mutual agreement between the independent college and the university to promote higher education. However, in both cases all the institutions of higher education are either controlled by private individuals or by a national agency, in such a way as to protect freedom of expression.
	Question:	What are some of the institutions of higher education that are controlled by private individuals?
	Answer:	private individuals

Table 4.2: Generated samples (context, question-answer pair) for the SQuAD domain. For the incorrectly generated samples, we underline one possible correct answer.

the ability of our G2D*f*ER method to enhance performance even in scenarios characterized by stringent constraints on data sharing, i.e., data from prior domains cannot be retained in the episodic memory.

4.4.2 How to most effectively use synthetic data for continual learning?

We empirically evaluate how to more effectively use synthetic data for continual learning, by comparing our method G2D*f*ER to a generative replay alternative. Our parameter-efficient method, G2D*f*ER (PEFT), improves over generative replay by 6.2 F_1 points. In Table 4.2, we visualize generated samples for their quality. Intuitively, these gains can be attributed to the fact that marginal distribution over a given domain $p(x)$ can be modeled more easily than the conditional $p(y|x)$ using synthetic samples. Therefore, we observe that **synthetic samples used for domain discrimination bring greater performance improvement than those used for generative replay**.

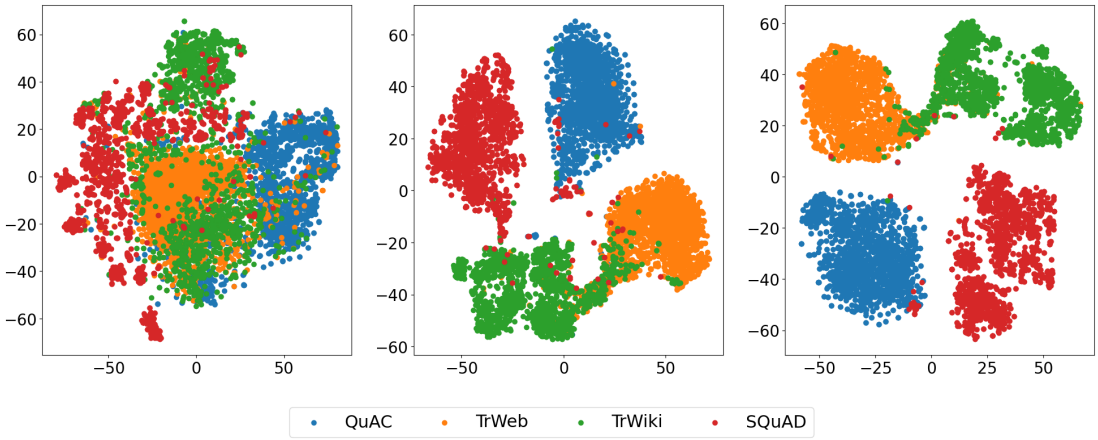


Figure 4.2: Domain discrimination visualization. t-SNE visualizations of domain clusterings for question-answering benchmark (4 domains in total). The left plot highlights the implicit domain discriminative nature of pre-trained BERT-base language model representations (Devlin et al., 2019). Notably, there is confusion between the TrWeb (orange) and TrWiki (green) domains, both derived from the same TriviaQA dataset. Similarly, the TrWiki (green) and SQuAD (red) domains, originating from the same Wikipedia source, necessitate explicit discriminator training. In the middle plot, we visualize the clustering of representations from the discriminator trained using generated samples, achieving a domain discrimination accuracy of 94.5%. On the right plot, we present the clustering from the discriminator trained using real samples, with an accuracy of 97.1%. Remarkably, the discriminator trained with synthetic samples closely mirrors the performance and clustering patterns of the discriminator trained using real data.

4.4.3 Domain discrimination analysis

Figure 4.2 displays t-SNE plots for the text domain, visualizing the domain discriminative capability (Aharoni and Goldberg, 2020) of a pre-trained language model with a discriminator trained on synthetic samples. Notably, there is confusion between the TrWeb (orange) and TrWiki (green) domains, both derived from the same TriviaQA dataset (Joshi et al., 2017). Similarly, the TrWiki (green) and SQuAD (red) domains, originating from the same Wikipedia source, necessitate explicit discriminator training. It is evident that training an explicit discriminator results in superior clustering. To assess domain identification performance using generated samples (achieving 94.5% accuracy), we compare it with a theoretical upper bound, namely a discriminator trained using real samples (achieving 97.1% accuracy). The results show very close performance, with similar clustering patterns. In summary, it is evident that **our G2D*f*ER method significantly improves domain identifiability.**

4.4.4 Parameter efficient fine-tuning analysis

In our work, we employ parameter-efficient fine-tuning (PEFT; Mangrulkar et al., 2022) techniques to address potential efficiency concerns in terms of storage. We use prompt tuning (Lester et al., 2021) to learn parameter-efficient generative models. Despite the great reduction in trainable parameters, we empirically observe that the performance of our domain discriminator trained on

these generated samples is sufficient for domain identification and outperforms previous existing methods. For our domain-specific experts, we fine-tune only 1.07% of trainable parameters, again, resulting in a great reduction in both computational and storage efficiency. Despite this reduction in parameters, we can reach state-of-the-art performance on the considered benchmark. We further analyze a full fine-tuning variant of our method, i.e., G2D*f*ER (Full FT), to study the potential performance drop we are experiencing from the sizeable reduction in trainable parameters.

4.5 Related Work

We focus on the domain-incremental setting of continual learning (Van de Ven and Tolias, 2019). Most continual learning methods fall into the categories of (i) parameter-based regularization, (ii) rehearsal-based, and (iii) rehearsal-free techniques (Sodhani et al., 2022).

Parameter-based regularization approaches — Two notable methods within this category are Elastic Weight Consolidation (EWC; Kirkpatrick et al., 2017) and Synaptic Intelligence (SI; Zenke et al., 2017). Both EWC and SI assess the importance of parameters related to previous domains and utilize a penalty term to safeguard the knowledge stored in those parameters while updating them for new domains.

Rehearsal-based approaches retain a subset of data from previous domains as an episodic memory, which is sparsely replayed during the learning of new domains. Several replay-based methods have been proposed, each differing in whether the episodic memory is utilized during training, such as Gradient Episodic Memory (GEM; Lopez-Paz et al., 2017), Averaged Gradient Episodic Memory (A-GEM; Chaudhry et al., 2018b), Episodic Replay (ER; Chaudhry et al., 2019), Mixed Stochastic Gradient (MEGA; Guo et al., 2020), or during inference, like Memory-based Parameter Adaptation (MbPA++; de Masson D'Autume et al., 2019, Meta-MbPTA; Mehta et al., 2020). These methods assume that the true data can be retained for replay. However, in settings where data sharing is restricted, one may not be able to retain actual samples from prior domains in episodic memory. To address this limitation, generative replay-based methods have been introduced (DGR; Shin et al., 2017, LAMOL; Sun et al., 2020, LFPT5; Qin and Joty, 2022). The main idea behind these methods is to learn a generative model of the data and use it to generate samples for experience replay. However, Sun et al. (2020) demonstrates that for a given sample complexity, ER tends to be an upper bound in terms of the performance for generative replay.

Rehearsal-free approaches have emerged in the field of continual learning in response to the increasing popularity of pre-trained models. Mehta et al. (2023b) demonstrate that pre-trained initializations implicitly mitigate the issue of forgetting when sequentially fine-tuning models. Another subcategory of approaches, known as prompt-based continual learning, exemplified by L2P (Wang et al., 2022c), DualPrompt (Wang et al., 2022b), S-Prompt (Wang et al., 2022a), and CODA-Prompt (Smith et al., 2023), involves learning a small number of parameters per domain in the form of continuous token embeddings or prompts while keeping the remaining pre-trained model fixed. The appropriate prompt is then selected based on the input data. Although these

methods allow for continual learning without rehearsal, they depend on access to pre-trained models that provide a high-quality backbone across all domains, which may not be available in sensitive environments in real-world deployment (e.g., healthcare). Another classical approach involves incorporating task-specific experts for each new task in a sequence and subsequently using an expert gate to direct examples to the appropriate expert (Aljundi et al., 2017). Like prompt-based methods, this approach relies on the inherent domain discriminative abilities of AlexNet (pre-trained with ImageNet) and is susceptible to the same limitations mentioned earlier in the context of prompt-based methods.

4.6 Discussion

In this work, we propose Generate to Discriminate for Expert Routing (G2D*f*ER), an expert routing method for domain-incremental learning using generative models for domain discrimination. Experiments demonstrate that our approach is competitive with previous test-time adaptation-based state-of-the-art approaches and outperforms by 6.2 F_1 in scenarios characterized by stringent constraints on data sharing. Further, we analyze how to most effectively leverage the capabilities of generative models and synthetic data for continual learning, by comparing our method to a generative replay alternative. Surprisingly, we find that training a domain identifier is significantly more effective than using synthetic samples to augment training data for experience replay. We further analyze our domain discriminator, by comparing it with previous domain discrimination approaches (unsupervised clustering methods) and a discriminator trained using real samples, where we find that our method significantly improves domain identifiability. Our method also employs parameter-efficient fine-tuning methods in the generator, discriminator, and downstream experts, to address potential efficiency concerns, enabling progress on dynamic parameter-efficient experts for lifelong learning.

Part II

Training: Objective & Optimization

Chapter 5

Optimization: Lifelong Learning with Sharpness Aware Minimization

5.1 Overview

In Chapter 3, we looked at the role of initialization in alleviating forgetting. Specifically, pre-trained initializations favor flat loss basins and implicitly mitigate forgetting to some extent. On the other hand, Mirzadeh et al. (2020b) suggests modifying the training regime by varying learning rate decay, batch size, and dropout regularization such that inherent noise in the stochastic gradients leads to flat basins in the loss landscape. However, the procedure for tuning these hyperparameters is ill-defined for lifelong learning, rendering their strategy less helpful. Furthermore, the suggested hyperparameter sweep is expensive (e.g., 48 separate runs just for one dataset) and does not transfer across different architectures and datasets. Motivated by these shortcomings, we pose a question—**What if we modify the optimization dynamics by explicitly seeking flat loss basins during lifelong learning of the model**?

To answer the above question, we employ the Sharpness-Aware Minimization (SAM; Foret et al., 2021) procedure that seeks parameters in neighborhoods with uniformly low loss regions by jointly minimizing loss value and loss sharpness. We show that this optimization approach outperforms several state-of-the-art task-sequential continual learning algorithms across multiple settings, occasionally even without retaining a memory that scales in size with the number of tasks. Our proposed approach is appealing in terms of not requiring explicit memory and it is suitable for lifelong learning applications where it may not be possible to retain data from previous tasks due to privacy constraints (Farquhar and Gal, 2018). Furthermore, our analysis of different initializations, namely task-agnostic meta-learned and supervised pre-trained models explicitly guided towards flat loss regions, highlights the synergistic benefits when combined with the explicit optimization for flatness during sequential fine-tuning.

5.2 Background: Sharpness-Aware Minimization (SAM)

SAM defines the sharpness of the loss function L at parameters w as

$$\max_{||\epsilon||_2 \leq \rho} L(w+\epsilon) - L(w), \tag{5.1}$$

where the maximization region is an ℓ^p ball with radius ρ for $p = 2$ in Equation 5.1. The SAM problem can be defined in terms of the following minimax optimization

$$\min_{w} \max_{||\epsilon||_2 \leq \rho} L(w+\epsilon) + \lambda ||w||_2^2. \tag{5.2}$$

The gradient of the result of the maximization problem can be approximated as

$$\nabla_w \max_{||\epsilon||_2 \leq \rho} L(w+\epsilon) \approx \nabla_w L(w)\Big|_{w+\hat{\epsilon}(w)} + \frac{d\hat{\epsilon}(w)}{dw} \nabla_w L(w)\Big|_{w+\hat{\epsilon}(w)}, \tag{5.3}$$

where

$$\hat{\epsilon}(w) = \rho \, \text{sign}\left(\nabla_w L(w)\right) \left(\frac{\left|\nabla_w L(w)\right|}{\|\nabla_w L(w)\|_2} \right). \tag{5.4}$$

To make the optimization simpler, the second-order term in the gradient is dropped, leaving us with

$$\nabla_w \max_{||\epsilon||_2 \leq \rho} L(w+\epsilon) \approx \nabla_w L(w)\Big|_{w+\hat{\epsilon}(w)}. \tag{5.5}$$

For the complete derivation of this gradient, we defer readers to Foret et al. (2021).

5.3 Does SAM alleviate forgetting during lifelong learning?

To investigate the efficacy of the SAM optimization procedure, we consider the experimental setup as discussed in Section 3.2. Specifically, we consider task-incremental learning scenarios with offline training for various NLP and CV benchmarks. We report the results with the discussed SAM procedure in Tables 5.1, 5.2 and 5.3. We see that SAM results in a consistent improvement in performance over non-SAM counterparts. Simply augmenting SAM with the finetune method (FT w/ SAM) results in a competitive baseline, sometimes outperforming state-of-the-art baselines like ER (Chaudhry et al., 2019), Stable SGD (Mirzadeh et al., 2020b) and MC-SGD (Mirzadeh et al., 2021) (see Table 5.1, Split CIFAR-50 w/PT ResNet-18-PT). Note SAM requires minimal hyper-parameter tuning (we set $\rho = 0.05$ to the default value for all our CV experiments). Since SAM is just a modification to the optimization procedure, we propose augmenting it with existing state-of-the-art continual learning methods like ER (Chaudhry et al., 2019) and MC-SGD (Mirzadeh et al., 2021).

Table 5.1 demonstrates that MC-SGD with SAM achieves superior performance in terms of overall accuracy and forgetting compared to all existing baseline methods (FT, Stable SGD, EWC, A-GEM, ER, MC-SGD, as described in Section 3.2.3) when evaluated across various CV benchmarks, including Split CIFAR-50, Split CIFAR-100, and 5-dataset-CV, under both random

	w/o PT (ResNet-18-R)			w/ PT (ResNet-18-PT)		
	Accuracy(↑)	Forgetting(↓)	LA(↑)	Accuracy(↑)	Forgetting(↓)	LA(↑)
Split CIFAR-50						
FT	$42.8_{3.1}$	$23.7_{1.1}$	$66.4_{2.1}$	$86.3_{1.2}$	$7.1_{0.9}$	$93.4_{0.5}$
FT w/ SAM	$50.3_{2.2}$	$15.0_{2.1}$	$65.3_{1.2}$	$\mathbf{90.5_{1.1}}$	$\mathbf{4.2_{1.0}}$	$94.7_{0.4}$
Stable SGD	$46.0_{2.3}$	$12.1_{0.4}$	$58.1_{2.5}$	$84.1_{1.9}$	$5.2_{1.6}$	$89.2_{0.7}$
EWC	$45.3_{2.5}$	$20.7_{1.5}$	$65.9_{1.3}$	$86.2_{0.9}$	$7.4_{0.9}$	$93.5_{0.7}$
A-GEM	$47.3_{2.7}$	$21.1_{2.0}$	$68.4_{0.7}$	$87.3_{1.0}$	$6.2_{0.6}$	$93.4_{0.4}$
ER	$45.8_{1.8}$	$20.6_{1.4}$	$66.4_{2.7}$	$86.2_{1.1}$	$7.1_{0.8}$	$93.3_{0.5}$
ER w/ SAM	$50.8_{0.5}$	$16.9_{0.9}$	$67.6_{0.7}$	$88.4_{1.3}$	$6.0_{1.1}$	$94.4_{0.3}$
MC-SGD	$59.0_{2.3}$	$5.4_{1.3}$	$63.7_{1.8}$	$86.5_{0.9}$	$4.1_{0.5}$	$90.4_{0.6}$
MC-SGD w/ SAM	$\mathbf{59.1_{2.9}}$	$\mathbf{5.2_{1.7}}$	$63.9_{2.1}$	$87.9_{0.7}$	$3.8_{0.2}$	$91.7_{0.7}$
5-dataset-CV						
FT	$33.7_{2.5}$	$51.5_{2.6}$	$85.2_{2.0}$	$57.2_{5.1}$	$38.3_{5.0}$	$95.5_{0.2}$
FT w/ SAM	$47.6_{3.8}$	$40.6_{4.0}$	$88.2_{1.3}$	$70.4_{4.4}$	$25.6_{4.4}$	$96.0_{0.1}$
Stable SGD	$50.2_{7.0}$	$40.3_{7.8}$	$90.5_{1.0}$	$71.3_{2.7}$	$20.5_{2.5}$	$91.9_{0.8}$
EWC	$35.0_{4.9}$	$50.1_{6.5}$	$85.1_{1.9}$	$56.7_{3.8}$	$38.8_{3.8}$	$95.4_{0.2}$
A-GEM	$46.1_{6.8}$	$39.5_{7.1}$	$85.2_{2.5}$	$72.0_{2.3}$	$23.0_{2.3}$	$95.0_{0.2}$
ER	$50.6_{4.5}$	$35.0_{5.4}$	$85.6_{1.3}$	$70.7_{1.5}$	$24.2_{1.4}$	$94.9_{0.2}$
ER w/ SAM	$60.3_{3.9}$	$27.3_{4.1}$	$87.6_{1.3}$	$77.4_{3.9}$	$18.2_{3.9}$	$95.6_{0.2}$
MC-SGD	$71.3_{5.9}$	$21.3_{5.7}$	$92.6_{0.3}$	$81.9_{2.6}$	$13.3_{2.6}$	$95.2_{0.3}$
MC-SGD w/ SAM	$\mathbf{72.7_{7.8}}$	$\mathbf{19.9_{7.6}}$	$92.6_{0.4}$	$\mathbf{87.1_{1.6}}$	$\mathbf{8.5_{1.7}}$	$95.5_{0.2}$
Split CIFAR-100						
FT	$38.9_{2.2}$	$39.1_{2.0}$	$78.0_{1.0}$	$82.0_{3.0}$	$13.8_{2.6}$	$95.8_{0.5}$
FT w/ SAM	$48.9_{4.8}$	$28.5_{5.0}$	$77.5_{0.9}$	$88.3_{1.7}$	$8.6_{1.3}$	$96.9_{0.6}$
Stable SGD	$52.9_{1.7}$	$21.0_{2.0}$	$73.8_{1.5}$	$86.6_{2.2}$	$5.5_{1.5}$	$91.8_{0.7}$
EWC	$37.4_{1.5}$	$40.1_{1.8}$	$77.5_{1.5}$	$81.3_{2.5}$	$14.5_{2.1}$	$95.8_{0.6}$
A-GEM	$46.8_{3.5}$	$32.0_{3.8}$	$78.8_{1.0}$	$84.0_{1.6}$	$11.7_{1.0}$	$95.7_{0.7}$
ER	$48.6_{1.9}$	$29.8_{1.3}$	$78.1_{0.7}$	$84.4_{2.2}$	$11.4_{1.7}$	$95.8_{0.5}$
ER w/ SAM	$60.5_{0.5}$	$20.9_{0.7}$	$81.4_{0.7}$	$88.4_{0.7}$	$8.6_{0.2}$	$96.7_{0.5}$
MC-SGD	$62.2_{2.4}$	$13.3_{1.8}$	$75.2_{0.7}$	$84.7_{2.4}$	$8.5_{1.8}$	$93.0_{0.7}$
MC-SGD w/ SAM	$\mathbf{65.1_{1.1}}$	$\mathbf{10.4_{1.1}}$	$75.4_{1.0}$	$\mathbf{89.0_{1.7}}$	$\mathbf{5.3_{1.1}}$	$94.2_{0.6}$

Table 5.1: Comparing performance in terms of average accuracy(%), forgetting(%), and learning accuracy(%) (see Equation 2.6) across methods after training on the last task of CV benchmarks (all metrics are averaged over five random task sequences). ↑ indicates higher is better, ↓ indicates lower is better. Augmenting the FT baseline with SAM results in performance competitive with state-of-the-art methods, and augmenting the ER or MC-SGD method with SAM often outperforms state-of-the-art methods demonstrating SAM as a valuable addition to current lifelong learning methods.

	w/o PT (DistilBERT-R)			w/ PT (DistilBERT-PT)		
	Accuracy(↑)	Forgetting(↓)	LA(↑)	Accuracy(↑)	Forgetting(↓)	LA(↑)
Split YahooQA						
FT	$73.1_{4.7}$	$26.4_{5.9}$	$94.2_{0.0}$	$87.7_{3.7}$	$9.5_{4.7}$	$95.2_{0.0}$
FT w/ SAM	$73.5_{4.0}$	$25.9_{5.0}$	$94.2_{0.0}$	$88.5_{2.8}$	$8.4_{3.5}$	$95.2_{0.0}$
EWC	$76.1_{3.1}$	$22.7_{3.9}$	$94.2_{0.0}$	$89.5_{3.4}$	$7.1_{4.2}$	$95.2_{0.0}$
ER	$77.2_{3.3}$	$21.3_{4.2}$	$94.2_{0.0}$	$\mathbf{89.4_{0.7}}$	$\mathbf{7.3_{0.9}}$	$95.2_{0.0}$
ER w/ SAM	$\mathbf{77.5_{1.4}}$	$\mathbf{20.9_{1.8}}$	$94.2_{0.0}$	$89.0_{0.7}$	$7.8_{0.9}$	$95.2_{0.0}$
5-dataset-NLP						
FT	$44.3_{5.0}$	$36.7_{6.3}$	$73.6_{0.1}$	$64.3_{4.5}$	$16.7_{5.7}$	$77.7_{0.1}$
FT w/ SAM	$46.0_{5.0}$	$34.3_{6.3}$	$73.4_{0.1}$	$66.4_{2.8}$	$13.9_{3.5}$	$77.6_{0.1}$
EWC	$48.7_{4.9}$	$31.1_{6.2}$	$73.6_{0.0}$	$66.8_{3.3}$	$13.6_{4.2}$	$77.6_{0.1}$
ER	$56.3_{3.1}$	$21.6_{3.9}$	$73.6_{0.1}$	$70.2_{1.6}$	$9.4_{2.0}$	$77.7_{0.1}$
ER w/ SAM	$\mathbf{56.3_{3.9}}$	$\mathbf{21.5_{5.0}}$	$73.4_{0.1}$	$\mathbf{71.1_{1.2}}$	$\mathbf{8.1_{1.5}}$	$77.5_{0.0}$

Table 5.2: Comparing performance in terms of average accuracy(%), forgetting(%), and learning accuracy(%) across methods after training on the last task of NLP benchmarks (all metrics are averaged over five random task sequences). ↑ indicates higher is better, ↓ indicates lower is better. Augmenting the ER method with SAM often outperforms state-of-the-art methods demonstrating SAM as a valuable addition to current lifelong learning methods.

and pre-trained initialized models (ResNet-18-R, ResNet-18-PT). Similarly, Tables 5.2 and 5.3 demonstrate ER with SAM results in a method that outperforms all existing baselines in terms of overall accuracy and forgetting across considered NLP benchmarks. Furthermore, in Table 5.3, we present the results obtained with T5-Small (v1.1) (Raffel et al., 2020) on the 5-dataset-NLP and 15-dataset-NLP benchmarks. Consistent with encoder-only models such as DistilBERT, BERT, and RoBERTa, we observe that FT w/ SAM and ER w/ SAM outperform their non-SAM counterparts on the encoder-decoder architecture. This finding highlights the broad applicability of SAM across different model architectures, including both encoder-only and encoder-decoder setups. To summarize, **SAM serves as a valuable addition to current continual learning methods and can be seamlessly incorporated to enhance overall performance.**

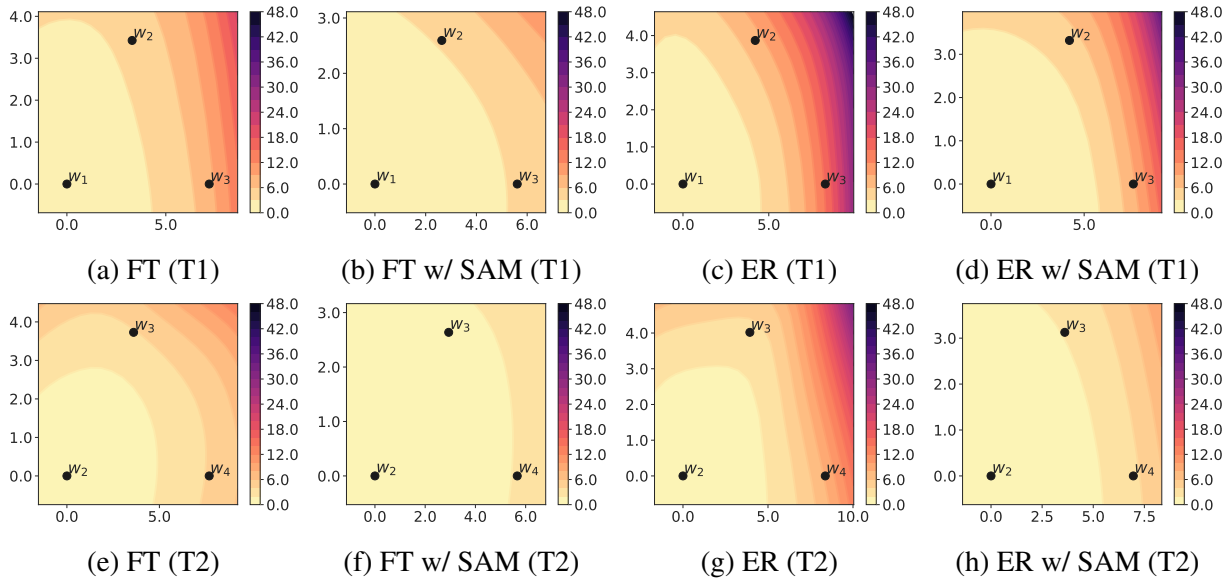
5.3.1 Loss Contours and Sharpness with SAM

To understand the effectiveness of the SAM, we visualize the loss contours (see Section 3.4.1 for details about how we plot loss contours) and compute the sharpness metric (see Section 3.4.3 for more details). We plot loss contours for task 1/ task 2 of Split CIFAR-50 (Figure 5.1) and 5-dataset-CV (Figure 5.2), under continual training from randomly initialized weights, and compare them across four different methods: FT, FT w/ SAM, ER, and ER w/ SAM. We show that SAM (FT w/ SAM and ER w/ SAM) leads to wide task minima (task 1/ task 2) across both datasets as compared to FT and ER methods. Moreover, from Table 5.1 for ResNet-18-R (w/o PT) initialization, we see that for Split CIFAR-50, FT w/ SAM (15.0), ER w/ SAM (16.9), and MC-SGD w/ SAM (5.2) undergoes lesser forgetting than FT (23.7), ER (20.6), and MC-SGD (5.4) methods, respectively.

	5-dataset-NLP			15-dataset-NLP		
	Accuracy(↑)	Forgetting(↓)	LA(↑)	Accuracy(↑)	Forgetting(↓)	LA(↑)
DistilBERT						
FT	$64.3_{4.5}$	$16.7_{5.7}$	$77.7_{0.1}$	$47.0_{3.5}$	$18.8_{4.0}$	$64.4_{1.2}$
FT w/ SAM	$66.4_{2.8}$	$13.9_{3.5}$	$77.6_{0.1}$	$47.5_{3.1}$	$16.5_{3.8}$	$62.5_{0.8}$
ER	$70.2_{1.6}$	$9.4_{2.0}$	$77.7_{0.1}$	$53.2_{3.4}$	$13.1_{4.1}$	$65.3_{0.6}$
ER w/ SAM	$\mathbf{71.1_{1.2}}$	$\mathbf{8.1_{1.5}}$	$77.5_{0.0}$	$\mathbf{53.5_{2.0}}$	$\mathbf{11.0_{3.1}}$	$63.1_{1.0}$
T5-Small						
FT	$65.9_{2.8}$	$12.6_{3.7}$	$76.0_{0.2}$	$44.9_{3.2}$	$21.9_{3.0}$	$65.2_{0.6}$
FT w/ SAM	$66.4_{2.3}$	$11.9_{3.0}$	$75.9_{0.3}$	$46.6_{3.0}$	$19.3_{3.0}$	$64.4_{0.7}$
ER	$69.4_{1.0}$	$8.2_{1.2}$	$75.9_{0.2}$	$48.1_{1.3}$	$18.9_{1.0}$	$65.5_{0.8}$
ER w/ SAM	$\mathbf{69.9_{0.2}}$	$\mathbf{7.5_{0.1}}$	$75.9_{0.2}$	$\mathbf{48.6_{1.6}}$	$\mathbf{17.4_{1.2}}$	$64.5_{0.7}$
BERT-base						
FT	$67.6_{2.8}$	$13.6_{3.4}$	$78.4_{0.1}$	$52.9_{2.8}$	$19.2_{3.0}$	$70.8_{0.3}$
FT w/ SAM	$70.8_{2.1}$	$9.5_{2.5}$	$78.4_{0.1}$	$55.1_{2.6}$	$16.4_{3.2}$	$70.4_{0.7}$
ER	$70.6_{2.1}$	$9.7_{2.6}$	$78.4_{0.1}$	$56.4_{2.9}$	$15.7_{3.0}$	$71.0_{0.4}$
ER w/ SAM	$\mathbf{73.0_{1.5}}$	$\mathbf{6.9_{1.9}}$	$78.5_{0.1}$	$\mathbf{57.8_{1.9}}$	$\mathbf{13.7_{1.9}}$	$70.5_{0.5}$
RoBERTa-base						
FT	$71.4_{1.7}$	$9.5_{2.0}$	$79.0_{0.1}$	$55.5_{3.1}$	$21.0_{3.0}$	$75.1_{0.5}$
FT w/ SAM	$72.6_{1.2}$	$7.8_{1.5}$	$78.8_{0.0}$	$57.8_{1.7}$	$15.4_{1.9}$	$72.1_{1.4}$
ER	$73.7_{0.8}$	$6.7_{0.9}$	$79.1_{0.1}$	$60.9_{1.4}$	$15.3_{1.6}$	$75.2_{0.1}$
ER w/ SAM	$\mathbf{74.3_{0.6}}$	$\mathbf{5.7_{0.8}}$	$78.9_{0.0}$	$\mathbf{62.1_{1.5}}$	$\mathbf{12.2_{2.1}}$	$73.3_{0.8}$
BERT-Large						
FT	$71.0_{2.0}$	$10.2_{2.5}$	$79.2_{0.0}$	$53.8_{1.2}$	$23.4_{1.4}$	$75.7_{0.4}$
FT w/ SAM	$73.7_{1.3}$	$6.9_{1.6}$	$79.2_{0.0}$	$58.7_{3.4}$	$17.1_{4.3}$	$74.6_{2.2}$
ER	$73.5_{1.2}$	$7.2_{1.4}$	$79.2_{0.1}$	$61.1_{2.6}$	$15.1_{2.9}$	$75.1_{1.2}$
ER w/ SAM	$\mathbf{74.6_{0.6}}$	$\mathbf{5.7_{0.8}}$	$79.2_{0.1}$	$\mathbf{61.7_{1.3}}$	$\mathbf{13.7_{2.8}}$	$74.5_{1.5}$

Table 5.3: Comparing performance in terms of average accuracy (%), forgetting (%), and learning accuracy (%) across pre-trained Transformers after continual learning the last task. ↑ indicates higher is better, ↓ indicates lower is better. All metrics are averaged over five random task sequences. Overall, we observe that larger models and/ or pre-trained on diverse and larger corpora (RoBERTa-base) undergo less forgetting on both 5 and 15 diverse tasks. Furthermore, augmenting the FT and ER methods with SAM often outperforms state-of-the-art methods.

These results convincingly demonstrate the effectiveness of SAM when used with vanilla FT, ER, and/ or MC-SGD methods. Similarly, we see that for 5-dataset-CV, MC-SGD w/ SAM (19.9) undergoes lesser forgetting than ER w/ SAM (27.3) and FT w/ SAM (40.6), which in turn significantly improves over FT (51.5). Next, we compare the loss contours between FT and ER methods and do not notice any stark difference in terms of flatness. However, in the presence of SAM, qualitatively we see that ER w/ SAM (Figures 5.2d, 5.2h) leads to a flat loss basin in comparison to FT w/ SAM (Figures 5.2b, 5.2f).



(a) FT (T1) (b) FT w/ SAM (T1) (c) ER (T1) (d) ER w/ SAM (T1)

(e) FT (T2) (f) FT w/ SAM (T2) (g) ER (T2) (h) ER w/ SAM (T2)

Figure 5.1: Loss contours for **task 1** (T1) and **task 2** (T2) of **Split CIFAR-50**. The top row visualizes loss contours for task 1 where $\mathbf{w}_1$**,** $\mathbf{w}_2$**,** $\mathbf{w}_3$ are minima obtained after sequential training on tasks 1, 2, and 3, respectively. Similarly, the bottom row visualizes loss contours for task 2 after sequential training on tasks 2, 3, and 4. All of the above models start with random weights. SAM (FT w/ SAM, ER w/ SAM) leads to wide task minima compared to finetune (FT) and ER methods.

Dataset	Method	$\epsilon = 5 \times 10^{-4}$		$\epsilon = 1 \times 10^{-3}$	
		ResNet-18-R	ResNet-18-PT	ResNet-18-R	ResNet-18-PT
5-dataset-CV	FT	$2.1_{0.6}$	$0.1_{0.0}$	$5.7_{1.6}$	$0.2_{0.0}$
	FT w/ SAM	$0.7_{0.2}$	$0.1_{0.0}$	$1.8_{0.4}$	$0.3_{0.0}$
Split CIFAR-50	FT	$2.3_{0.7}$	$0.2_{0.1}$	$6.1_{1.5}$	$0.4_{0.1}$
	FT w/ SAM	$0.7_{0.1}$	$0.2_{0.0}$	$2.0_{0.3}$	$0.6_{0.0}$

Table 5.4: Average sharpness value (lower value corresponds to flat loss basin) of task minima in a 100-dimensional random subspace. SAM significantly lowers the sharpness metric in comparison to the Finetune (FT) method in the case of randomly initialized models (ResNet-18-R).

We compute the sharpness metric for FT and FT w/ SAM methods. In Table 5.4 we report sharpness metrics for 5-dataset-CV and Split CIFAR-50. We see that the SAM significantly reduces the sharpness in the case of randomly initialized models. Concretely, on the 5-dataset-CV, we see that the sharpness value (for $\epsilon = 5\text{x}10^{-4}$) decreases from 2.1 (FT) to 0.7 (FT w/ SAM). Similarly, on the Split CIFAR-50, we see a drop from 2.3 (FT) to 0.7 (FT w/ SAM). These results validate that **SAM indeed leads to flat minima, therefore, explaining the superior performance (in terms of average accuracy and forgetting) of the SAM optimization procedure over baseline**.

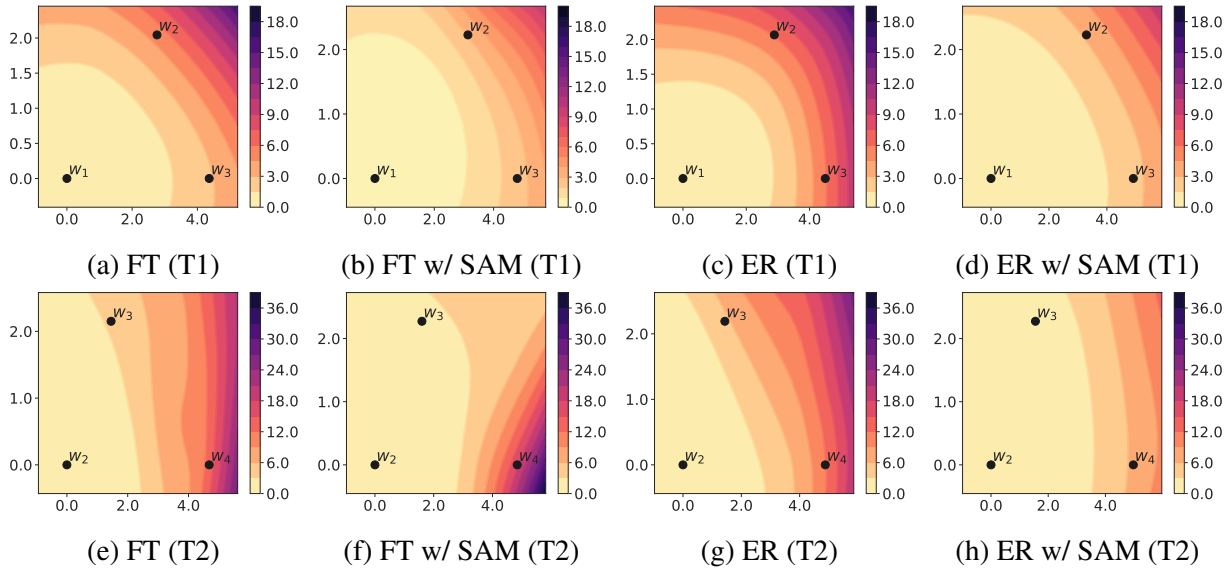


Figure 5.2: Loss contours for **SVHN** (T1) and **MNIST** (T2) of **5-dataset-CV**. The top row visualizes loss contours for SVHN where $\mathbf{w}_1$**,** $\mathbf{w}_2$**,** $\mathbf{w}_3$ are minima obtained after sequential training on SVHN, MNIST, and nonMNIST, respectively. Similarly, the bottom row visualizes loss contours for MNIST where $\mathbf{w}_2$**,** $\mathbf{w}_3$**,** $\mathbf{w}_4$ are minima obtained after sequential training on tasks MNIST, nonMNIST, and Fashion-MNIST. All of the above models start with random weights. SAM (FT w/ SAM, ER w/ SAM) leads to wide task minima compared to finetune (FT) and ER methods.

5.4 Analyzing the influence of pre-training task minima curvature on forgetting

In Chapter 3, we demonstrate that pre-trained initializations alleviate forgetting in lifelong learning scenarios by guiding optimization towards flat minima in the loss basin for sequentially trained tasks. Furthermore, in Section 5.3, we show that explicitly optimizing for the flatness of the loss basin using the SAM procedure leads to an additional reduction in forgetting. It is important to note that our discussion so far has mainly focused on the flatness of the loss basin near the fine-tuned task minima. However, we now inquire about the impact of the loss basin's flatness (or sharpness) near the pre-training task minima on forgetting during lifelong learning. Specifically, we actively push a pre-trained model toward a region with low (or high) curvature. While such a model would benefit from learning structure from pre-training data, it would reside in flat (or sharp) regions of the loss basin with respect to the pre-training task. The question we pose is — **What role does the curvature of the pre-training task minima play in lifelong learning, particularly concerning forgetting in the fine-tuned task?**

Experimental design. To answer the above question, we conduct a controlled experiment with SVHN as our pre-training task and analyze the forgetting in MNIST (and its subsets) when

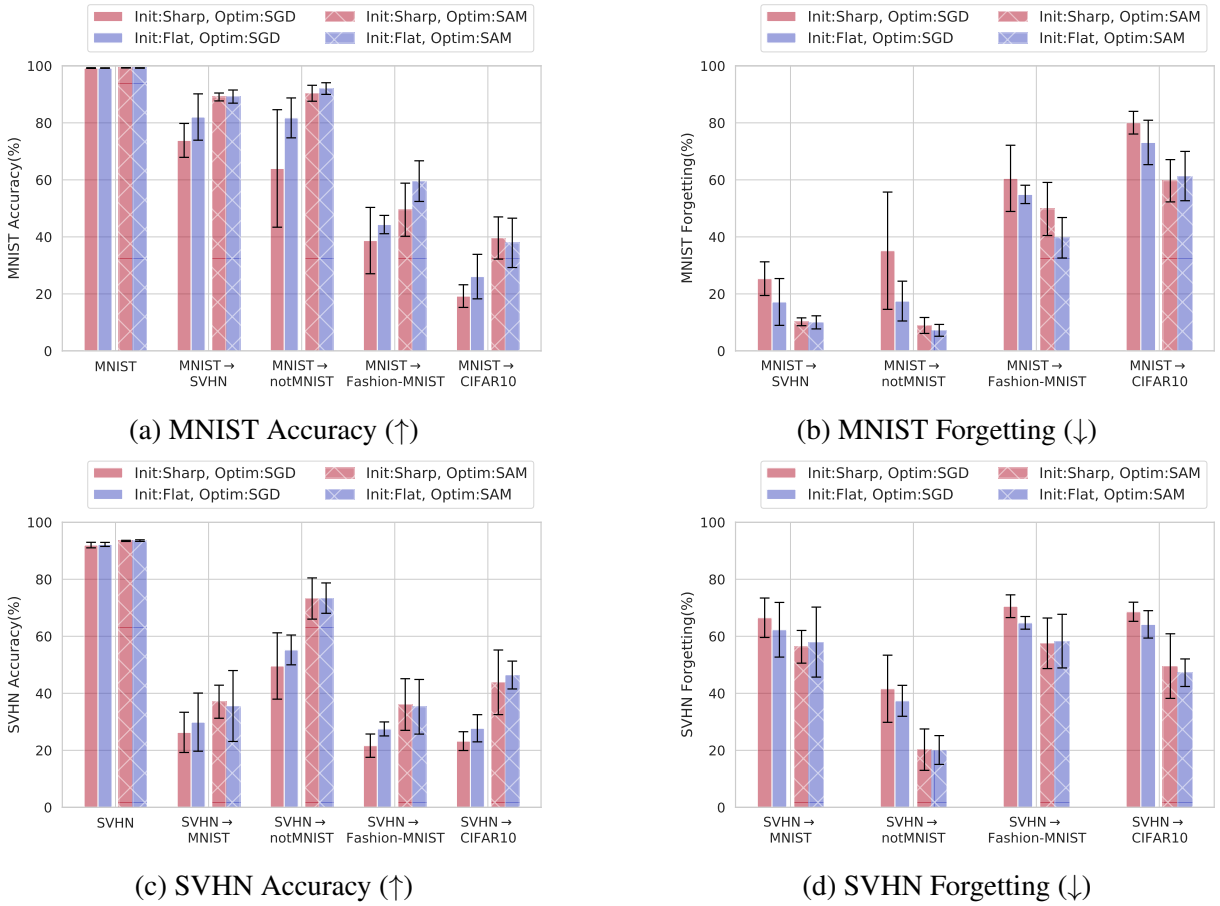
learning homogeneous as well as diverse tasks in a sequence. We pre-train two separate models on SVHN, ensuring that one model converges to a flat minimum using the SAM procedure, while the other model converges to a sharper minimum using the Nudged-SGD procedure (NSGD; Jastrzębski et al., 2019) as discussed in the following section. It is important to ensure that both models exhibit comparable generalization performance on SVHN. Subsequently, we initialize these models and perform sequential training on four diverse task sequences, starting with MNIST as our fine-tuning task. The task sequences are as follows: MNIST→SVHN, MNIST→notMNIST, MNIST→Fashion-MNIST, and MNIST→CIFAR10.

To create homogeneous tasks in the Split MNIST data set, we randomly select two digits for each task. Considering that MNIST comprises 10 digits, we then create a random sequence of five tasks. By utilizing different random seeds, we generate different tasks and consequently obtain a variety of task orderings. In total, we generated 25 distinct task sequences for the Split MNIST data set. By examining the degree of forgetting observed in the case of (Split) MNIST, we can gain insights into the influence of the flatness (or sharpness) of the pre-training task minimum on the subsequent forgetting phenomenon. To ensure the broad applicability of our findings, we further conduct experiments using MNIST as the pre-training task and SVHN as the initial task for fine-tuning over four diverse task sequences. It is important to highlight that we utilize NSGD exclusively for the experiments conducted in this section.

5.4.1 Nudged-SGD (NSGD)

Jastrzębski et al. (2019) investigate the SGD dynamics in relation to the sharpest directions of the loss basin and demonstrate that although SGD updates align closely with these directions, they generally fail to minimize the loss when solely constrained to these directions. To enhance both the convergence speed and the generalization of the resulting model, Jastrzębski et al. (2019) introduces a variant of SGD, called Nudged-SGD (NSGD). NSGD aims to reduce the alignment between the SGD update direction and the sharpest directions. Specifically, NSGD is implemented as follows: instead of the standard SGD update $\Delta w(t) = -\eta \mathbf{g}(t)$, NSGD employs a reduced learning rate, $\eta' = \gamma\eta$, along the top K eigenvectors. Meanwhile, NSGD continues to follow the standard SGD update in other directions. By setting $\gamma < 1.0$, NSGD diminishes the updates along the top K eigenvectors. As a result, training speed improves while converging to sharper and better generalizing minima in comparison to vanilla SGD. In our experimental setup, we utilize a randomly initialized ResNet-18 model. Following Jastrzębski et al. (2019), we set the parameters γ to 0.01 and K to 20. Additionally, we employ 100 training examples to compute the top K eigenvectors using pytorch-hessian-eigenthings (Golmant et al., 2018) for every 100 training updates. Lastly, we set tolerance (relative accuracy for eigenvalues) to $1e^{-5}$ for determining the convergence of the Lanczos algorithm.

To begin, we conduct supervised pre-training of ResNet18 models using two different optimization procedures: SAM and NSGD, with the SVHN data set. Utilizing SAM, we achieve a validation accuracy of $92.0(\pm 0.4)$ and a maximum eigenvalue λ_1^{max} of $338.5(\pm 91.5)$ (averaged over five runs). On the other hand, employing NSGD yields a validation accuracy of $92.4(\pm 0.3)$ and a maximum eigenvalue λ_1^{max} of $1416.8(\pm 411.2)$ (also averaged over five runs). It is evident from these results that the NSGD leads to convergence towards sharper minima (as indicated by higher λ_1^{max} values) and better generalization (as reflected in higher accuracy) compared to



(a) MNIST Accuracy (↑)

(b) MNIST Forgetting (↓)

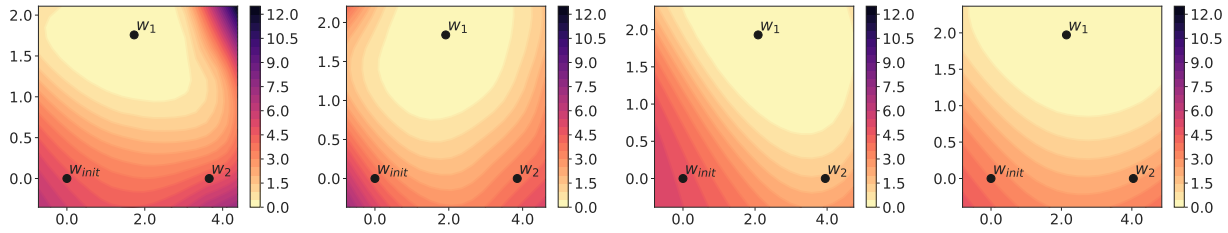
(c) SVHN Accuracy (↑)

(d) SVHN Forgetting (↓)

Figure 5.3: Comparing performance of the first task (MNIST in the top row, SVHN in the bottom row) after sequential training on the second task across different supervised pre-training initializations (Init:Sharp, Init:Flat) and optimization procedures (Optim:SGD, Optim:SAM). ↑ indicates higher performance, ↓ indicates lower performance. All metrics are averaged over five runs (see Equation 2.6). Pre-trained models converged to flat minima with respect to the pre-training task (Init:Flat, Optim:SGD) exhibit reduced forgetting with SGD in comparison to sharp minima(Init:Sharp, Optim:SGD). Notably, explicitly promoting flatness (Optim:SAM) for the fine-tuning task yields an even greater reduction in forgetting.

SAM and vanilla SGD. With vanilla SGD, we obtain a validation accuracy of $91.5(\pm 0.6)$ and a maximum eigenvalue λ_1^{max} of $1241.3(\pm 236.5)$. Consequently, we have successfully obtained pre-trained models with varying levels of flatness or sharpness concerning the pre-training SVHN task, achieved through the SAM (*Init:Flat*) and NSGD (*Init:Sharp*) procedures.

Discussion. To examine the influence of pre-training task minima curvature on forgetting during fine-tuning, we proceed to sequentially fine-tune the aforementioned pre-trained models (Init:Sharp or Init:Flat) on MNIST, followed by one of four tasks: SVHN, notMNIST, Fashion-MNIST, and CIFAR10. Additionally, to investigate the interplay between pre-trained minima



(a) Init:Sharp, Optim:SGD (b) Init:Flat, Optim:SGD (c) Init:Sharp, Optim:SAM (d) Init:Flat, Optim:SAM

Figure 5.4: Loss contours are shown for MNIST, with w_{init}, w_1, and w_2 representing the minima obtained after supervised pre-training on SVHN, followed by sequential training on MNIST and nonMNIST, respectively. The models are initialized either with a sharp pre-trained model (Init:Sharp) or a flat pre-trained model (Init:Flat). (a), (b) starting with a flat pre-trained model results in a flatter loss basin for MNIST during sequential fine-tuning. (c), (d) explicitly optimizing for flat MNIST minima using SAM (Optim:SAM) leads to even wider task minima compared to vanilla SGD.

curvature and optimization dynamics, we conduct sequential fine-tuning using either the vanilla SGD optimizer (*Optim:SGD*) or the SAM procedure (*Optim:SAM*). Figures 5.3a and 5.3b illustrate the accuracy and forgetting values for the MNIST task, respectively. From Figure 5.3b, it is evident that when fine-tuning with vanilla SGD (Optim:SGD), pre-trained models with flat minima (Init:Flat; shown in blue) consistently exhibit lower levels of forgetting compared to models with sharp minima (Init:Sharp; shown in red) across various task sequences. However, the advantage provided by the flat pre-trained models appears to diminish when employing the SAM optimization procedure (see Init:Sharp, Optim:SAM and Init:Flat, Optim:SAM). This finding highlights that the flatness of the MNIST task minima plays a more significant role in reducing forgetting for MNIST compared to the initialization flatness with respect to the pre-training task (SVHN in this case) minima. Similar observations are reported in Figures 5.3c and 5.3d when MNIST is used as the pre-training task, and forgetting is analyzed for the SVHN task during continual learning.

In Figure 5.4, we provide a comparison of the loss contours for MNIST using sharp pre-trained initialization (Figure 5.4a) and flat pre-trained initialization (Figure 5.4b). Upon visual inspection, we observe that the flat pre-trained initialization results in a wider loss basin for the MNIST minima (w_1), thereby, explaining lesser forgetting of MNIST when continually training on notMNIST in the case of pre-trained initialized models (see Figure 5.3b; Init:Flat, Optim:SGD). Furthermore, when SAM is applied, Optim:SAM (Figures 5.4c, 5.4d) yields an even wider loss basin compared to Optim:SGD (Figures 5.4a, 5.4b).

In Table 5.5, we present a comparison of average accuracy and forgetting on Split MNIST, focusing on sharp and flat supervised SVHN pre-trained initializations. Consistent with experiments involving diverse tasks (refer to Figure 5.3b), we find that Init:Flat, Optim:SGD (4.3) exhibits lower forgetting compared to Init:Sharp, Optim:SGD (7.6) in homogeneous tasks. This finding reinforces the notion that actively promoting flatness during pre-training, in addition to learning structure from abundant data, is beneficial for reducing forgetting in sequentially fine-tuned tasks. Moreover, initializing with a sharp pre-trained model and explicitly optimizing for flatness using

	Optim:SGD			Optim:SAM		
	Accuracy(↑)	Forgetting(↓)	LA(↑)	Accuracy(↑)	Forgetting(↓)	LA(↑)
Task-agnostic						
Init:Random	$80.9_{9.7}$	$17.6_{9.0}$	$96.9_{5.9}$	$91.9_{5.3}$	$7.7_{5.2}$	$99.6_{0.3}$
Init:Meta	$89.5_{9.1}$	$9.8_{8.0}$	$98.8_{3.2}$	$\mathbf{92.3_{6.4}}$	$\mathbf{7.5_{6.3}}$	$99.6_{0.3}$
Supervised pre-training (SVHN)						
Init:Sharp	$91.9_{8.1}$	$7.6_{7.6}$	$99.2_{1.8}$	$96.1_{4.7}$	$3.7_{4.6}$	$99.8_{0.2}$
Init:Flat	$95.2_{4.8}$	$4.3_{4.7}$	$99.1_{2.1}$	$\mathbf{97.8_{2.2}}$	$\mathbf{2.0_{2.1}}$	$99.7_{0.4}$

Table 5.5: Comparing the performance of Split MNIST in terms of average accuracy(%), forgetting(%), and learning accuracy(%), we analyze the impact of different initializations: random (Init:Random), task-agnostic meta-learned (Init:Meta), supervised pre-trained with explicit optimization for sharp minima (Init:Sharp), and supervised pre-trained with explicit optimization for flat minima (Init:Flat). We also consider different optimization approaches: vanilla SGD (Optim:SGD) and explicit optimization for flatness (Optim:SAM). ↑ indicates higher performance, while ↓ symbolizes lower performance. All metrics are averaged over 25 random task sequences. Our observations indicate that with Optim:SGD, MetaInit significantly reduces forgetting compared to random initialization, suggesting the benefits of initializing in flatter regions with minimal susceptibility to second-order effects. Also, pushing supervised pre-trained models towards flatter regions contributes to reduced forgetting. However, these gains diminish across all initialization schemes when explicitly optimizing for flatness (Optim:SAM) during lifelong learning. Nevertheless, we observe synergistic advantages when utilizing flat pre-trained or meta-initialized models in conjunction with the SAM procedure, resulting in minimal forgetting.

SAM, as seen in Init:Sharp, Optim:SAM (3.7), yields an even greater reduction in forgetting compared to flat pre-trained initialization with vanilla SGD, i.e., Init:Flat, Optim:SGD (4.3), aligning with our previous observations (refer to Figure 5.3). However, we observe synergistic advantages when utilizing flat pre-trained models in conjunction with the SAM optimization procedure, resulting in minimal forgetting. Specifically, the combination of Init:Flat and Optim:SAM yields a forgetting value of 2.0, showcasing the complementary benefits of these approaches.

To summarize, **initiating fine-tuning with pre-trained models that have converged to flat minima with respect to the pre-training task helps mitigate forgetting. However, explicitly promoting flatness with respect to the fine-tuning task leads to a more pronounced reduction in forgetting.**

5.5 Analyzing the influence of task-agnostic favorable initializations on forgetting

Initialization is widely recognized as a crucial factor in a model's learning dynamics and overall performance (Glorot and Bengio, 2010; LeCun et al., 2015; Mishkin and Matas, 2015). Various initialization schemes have been developed for specific network architectures, such as fully

connected networks (Pennington et al., 2017), residual networks (He et al., 2015), convolutional networks (Xiao et al., 2018), recurrent networks (Chen et al., 2018), and attention-based networks (Huang et al., 2020). However, these schemes are often architecture-specific and do not readily transfer to different or novel architectures. To overcome this limitation, Dauphin and Schoenholz (2019) propose *MetaInit*, an automated initialization search approach using task-agnostic meta-learning. On the other hand, pre-training has also been shown to yield favorable initializations in terms of optimization (Erhan et al., 2009; Hao et al., 2019; Neyshabur et al., 2020), making it a data-driven method for finding effective initialization schemes. In the previous sections, we observed that pre-trained initializations reduce forgetting during sequential fine-tuning. As these observations pertain to pre-trained initializations, this section aims to directly investigate the contribution of favorable initializations by removing the pre-training aspect. Specifically, we pose the question — **Do good initializations from MetaInit, shown to facilitate gradient descent by starting in locally linear (or wider) regions with minimal second-order effects, also mitigate forgetting when compared to a random initialization in a sharper region?**

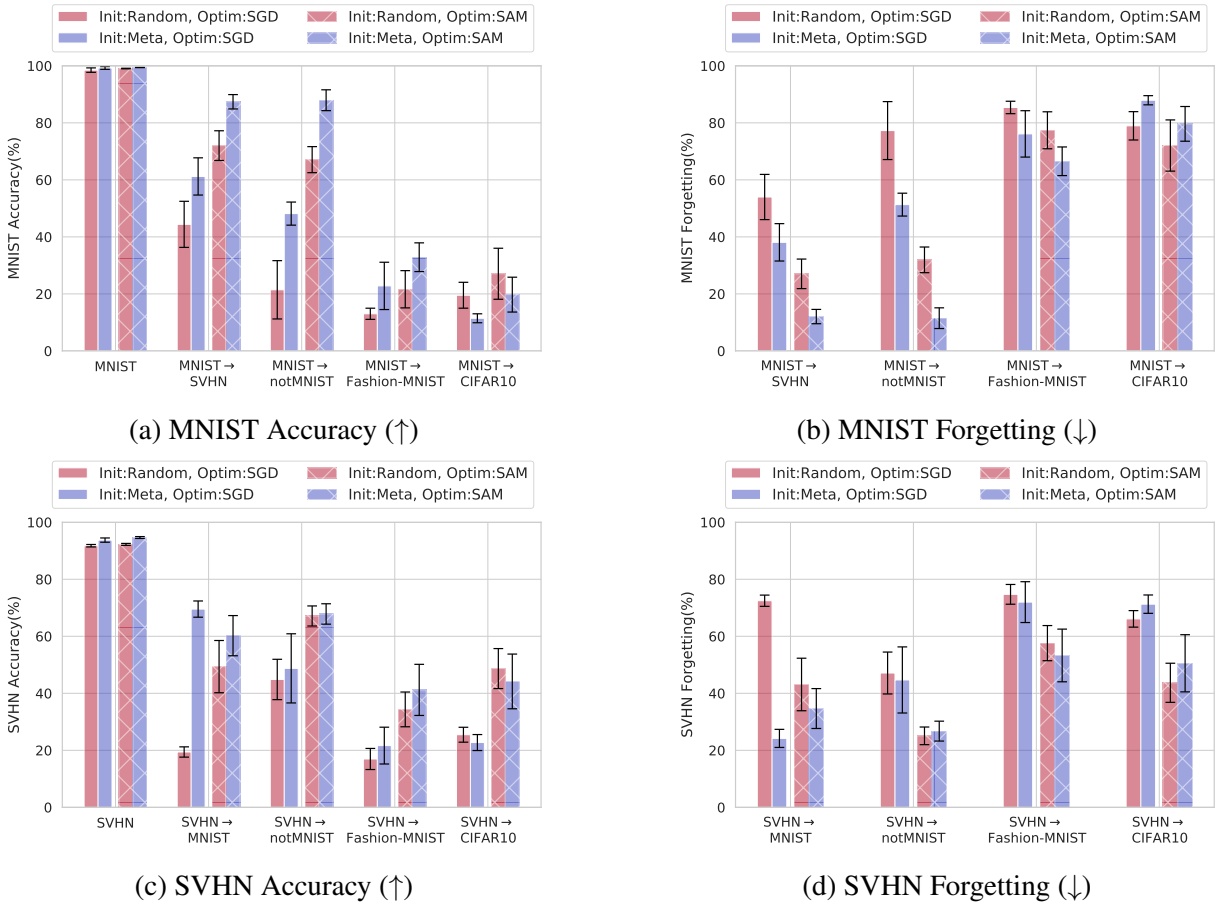
Experimental design. To address the above question, we perform controlled experiments to investigate the phenomenon of forgetting in the MNIST data set, both in the context of homogeneous and diverse task sequences. For this purpose, we employ the MetaInit algorithm (Dauphin and Schoenholz, 2019) to obtain a task-agnostic favorable initialization, which is independent of the specific task at hand. By comparing models initialized with the MetaInit approach to randomly initialized models, we aim to assess the impact of task-agnostic favorable initialization on forgetting. We examine the degree of forgetting specifically in the MNIST data set while sequentially learning four task sequences: MNIST $\rightarrow$ SVHN, MNIST $\rightarrow$ notMNIST, MNIST $\rightarrow$ Fashion-MNIST, and MNIST $\rightarrow$ CIFAR10, where MNIST serves as the initial task. Additionally, similar to the methodology described in Section 5.4, we conduct experiments involving homogeneous tasks from the Split MNIST data set.

5.5.1 MetaInit: Initializing learning by learning to initialize

Dauphin and Schoenholz (2019) demonstrate that effective initializations exhibit characteristics that aid gradient descent by beginning in regions with minimal susceptibility to second-order effects. To quantify the impact of curvature (or second-order effects) around an initial choice of parameters, they introduce a quantity known as the gradient quotient (GQ). GQ measures the change in the gradient of a function following a single gradient descent step. Mathematically, the GQ is defined as follows

$$\text{GQ}(L, w) = \frac{1}{N} \left\| \frac{\mathbf{g}(w) - \mathbf{H}(w)\mathbf{g}(w)}{\mathbf{g}(w) + \epsilon} - 1 \right\|_1 \approx \frac{1}{N} \left\| \frac{\mathbf{g}(w - \mathbf{g}(w))}{\mathbf{g}(w) + \epsilon} - 1 \right\|_1, \tag{5.6}$$

where $w \in \mathbb{R}^N$ are network parameters, L is an empirical loss computed over the batch of the examples, $\mathbf{g}(w) = \nabla L(w)$ is the gradient, $\mathbf{H}(w) = \nabla^2 L(w)$ denotes Hessian matrix, $\epsilon = \epsilon_0(2_{\mathbf{g}(w) \geq 0} - 1)$ with ϵ_0 as a small constant and $||.||_1$ is the L1 vector norm. Alternatively, the GQ can be interpreted as the relative change in the gradient per parameter after a single step of gradient descent. As a result, parameters that cause a rapid change in the gradient exhibit large



(a) MNIST Accuracy (↑)

(b) MNIST Forgetting (↓)

(c) SVHN Accuracy (↑)

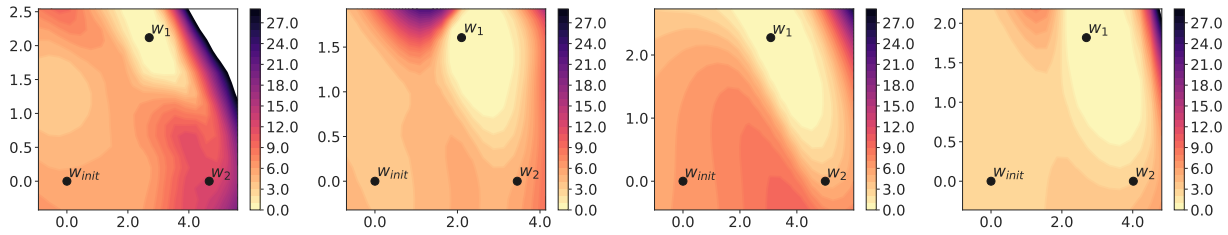
(d) SVHN Forgetting (↓)

Figure 5.5: Comparing the performance of the first task (*MNIST* in the top row, *SVHN* in the bottom row) after sequential training on the second task, we examine the impact of random and task-agnostic MetaInit initialization strategy (Init:Random, Init:Meta) and optimization procedures (Optim:SGD, Optim:SAM). ↑ indicates higher performance, while ↓ symbolizes lower performance. All metrics are averaged over 5 runs. The results show that task-agnostic MetaInit models (Init:Meta, Optim:SGD) exhibit reduced forgetting with SGD compared to random initialization (Init:Random, Optim:SGD). Similarly to Figure 5.3, explicitly promoting flatness (Optim:SAM) for the sequential task leads to an even greater reduction in forgetting.

gradient quotients, while an optimal GQ of 0 is achieved when the loss function $L(w)$ behaves almost linearly, indicating a negligible Hessian matrix $\mathbf{H}(w) \approx 0$. Having established this metric to assess initialization quality, Dauphin and Schoenholz (2019) present MetaInit, a task-agnostic meta-learning algorithm aimed at obtaining a good initialization from suboptimal ones. The meta-objective of MetaInit is defined as follows

$$\text{MetaInit}(L, w^*) = \arg\min_{w} \text{GQ}(L, w). \tag{5.7}$$

Following the methodology proposed by Dauphin and Schoenholz (2019), we optimize the aforementioned meta-objective using random input data ($\mathbf{x} \sim \mathcal{N}(0, 1)$). It can be argued that



(a) Init:Random, Optim:SGD (b) Init:Meta, Optim:SGD (c) Init:Random, Optim:SAM (d) Init:Meta, Optim:SAM

Figure 5.6: Loss contours are shown for MNIST, with w_{init}, w_1, and w_2 representing either random or task-agnostic initialization, followed by sequential training on MNIST and nonMNIST, respectively. The models are initialized either with a random strategy (Init:Random) or a MetaInit strategy (Init:Meta). (a), (b) starting with a MetaInit initialization results in a flatter loss basin for MNIST during sequential fine-tuning. (c), (d) explicitly optimizing for flat MNIST minima using SAM (Optim:SAM) leads to even wider task minima compared to vanilla SGD.

solving for the meta-objective resembles a form of pre-training; however, no task-specific data is utilized in the learning process, thereby characterizing it as a task-agnostic initialization. Moreover, consistent with previous studies (Glorot and Bengio, 2010; Dauphin and Schoenholz, 2019), we solely adjust the norms of the initial weight matrices. We begin with five ResNet-18 models that are randomly initialized (*Init:Random*), with a GQ averaging $5476.2(\pm 2804.3)$. Through the MetaInit procedure (*Init:Meta*), we attain a final GQ value of $0.9(\pm 0.0)$.

Discussion. To investigate the impact of task-agnostic favorable initializations on forgetting, we sequentially train starting from the aforementioned initialization (Init:Random or Init:Meta) on MNIST, followed by one of four diverse tasks: SVHN, notMNIST, Fashion-MNIST, and CIFAR10. Furthermore, to explore the relationship between meta-learned initializations and optimization dynamics, we use either vanilla SGD (*Optim:SGD*) or the SAM procedure (*Optim:SAM*). The accuracy and forgetting values for the MNIST task are visualized in Figures 5.5a and 5.5b, respectively. Figure 5.5b demonstrates that when optimized with vanilla SGD (Optim:SGD), meta-initialized models (Init:Meta; shown in blue) exhibit lower forgetting levels compared to randomly initialized models with high gradient quotient (Init:Random; shown in red) across various task sequences, except for CIFAR10, which differs significantly from MNIST. However, the advantage of meta-initialized models appears to diminish when employing the SAM optimization procedure (see Init:Random, Optim:SAM and Init:Meta, Optim:SAM), highlighting the more significant role of MNIST task minima flatness in reducing forgetting compared to the curvature of task-agnostic initialization. Similar observations are reported in Figures 5.5c, 5.5d when starting from SVHN task followed by one of MNIST, notMNIST, Fashion-MNIST, and CIFAR10 task during lifelong learning.

In Figure 5.6, we provide a comparison of the loss contours for MNIST using random initialization (Figure 5.6a) and task-agnostic MetaInit initialization (Figure 5.6b). Upon visual inspection, we observe that the MetaInit initialization results in a wider loss basin for the MNIST minima (w_1), thereby, explaining lesser forgetting of MNIST when continually training on

notMNIST in the case of MetaInit initialized models (see Figure 5.5b; Init:Meta, Optim:SGD). Furthermore, when SAM is applied, Optim:SAM (Figures 5.6c, 5.6d) yields an even wider loss basin compared to Optim:SGD (Figures 5.6a, 5.6b), explaining superior results with Optim:SAM (refer to Figures 5.5a, 5.5b).

Table 5.5 presents a comparative analysis of average accuracy and forgetting on the Split MNIST data set, focusing on random initialization and task-agnostic MetaInit initialization. Consistent with our experiments involving diverse tasks (see Figure 5.5b), we observe that Init:Meta, Optim:SGD (9.8) exhibits significantly lower levels of forgetting compared to Init:Random, Optim:SGD (17.6) in the context of homogeneous tasks. This observation supports the claim that initializing model parameters with regions that are locally linear (or wider) and are less susceptible to second-order effects can be beneficial in mitigating forgetting in lifelong learning. Furthermore, we find that random initialization and explicit optimization for flatness using SAM, as demonstrated by Init:Random, Optim:SAM (7.7), result in an even greater reduction in forgetting compared to MetaInit initialization with vanilla SGD, i.e., Init:Meta, Optim:SGD (9.8), which aligns with our previous observations (see Figure 5.5).

To summarize, **initializing models with lower gradient quotients (achieved through task-agnostic meta-learning) in regions that are less susceptible to second-order effects help reduce forgetting during lifelong learning. However, the reduction in forgetting is more significant when explicitly promoting flatness in relation to the specific sequential learning task.**

5.6 Discussion

In this chapter, we show that explicitly seeking flat loss basins using the SAM procedure yields even lower forgetting than existing methods. Integrating SAM into the baselines surpasses state-of-the-art techniques, underscoring its valuable contribution to advancing lifelong learning methods. Our analysis of various initializations, including task-agnostic meta-learned and supervised pre-trained models explicitly guided towards flat loss regions, showcases the synergistic behavior that arises when combined with the SAM procedure during sequential fine-tuning.

Recent research (Na et al., 2022) indicates that optimizing for flatter minima results in greater parameter compressibility compared to standard optimization, suggesting potential sparsity in the learned network. In this chapter and Chapter 3, we explored the relationship between flatness and reduced forgetting. If flatness indeed leads to sparsity, it could imply less interference between sequential task solutions and, consequently, less forgetting. Further investigation into this potential explanation could be a promising direction for future research.

Chapter 6

Objective: Efficient Meta Lifelong Learning with Limited Memory

6.1 Overview

Previously (Chapters 3, 4, 5), we examined task-incremental learning (Section 2.1.1) or domain-incremental learning (Section 2.1.2) scenarios in lifelong learning, where the task identifier is available during training (Section 4.2.1) and/ or testing and multiple passes over the dataset are allowed during training (Section 3.2.1). However, in real-world situations, the input distribution may change without warning; therefore, the system must learn from an online stream with just one pass over the training examples and without any task identifiers (both during training and testing). To enable lifelong learning systems operating under such realistic assumptions (Section 2.1), in this chapter, we investigate domain-incremental (Section 2.1.2) and class-incremental (the most challenging; Section 2.1.3) scenarios while allowing only one pass over the training examples, and without any task identifiers during training as well as evaluation.

One approach to lifelong learning under realistic assumptions has been augmenting the learning model with an episodic memory module (Sprechmann et al., 2018). The underlying idea is to first store previously seen training examples in memory and later use them to perform experience replay (Rolnick et al., 2019) or to derive optimization constraints (Lopez-Paz et al., 2017; Chaudhry et al., 2018b) while training on new tasks. Recently, d'Autume et al. (2019) proposed to use such a memory module for sparse experience replay and local adaptation in the language domain, achieving then state-of-the-art results for lifelong learning on text classification and question-answering tasks. Despite its success, the method has three critical downsides, which we demonstrate later in our experiments

1. It requires an unrealistically large memory module, i.e., storing **all** training examples, to achieve optimal performance.
2. While the model can mitigate catastrophic forgetting over *previous tasks*, its local adaptation step is prone to negative transfer such that it performs worse on the *current task* than the naive baseline without any lifelong learning regularization.
3. Its inference speed is extremely slow due to a non-trivial amount of local adaptation steps required for **each** test example.

This chapter addresses these limitations and tackles the problem of *efficient* lifelong language learning. That is, we focus on storing limited training examples in memory. Driven by the CLS theory (McClelland et al., 1995), we identify three components of an existing method (de Masson D'Autume et al., 2019)—generic representations, experience rehearsal, and local adaptation, each corresponding to one of the two learning phases, that are independent rather than complementary. We introduce a "synergistic" framework that makes two learning phases complementary. We propose a novel first-order meta-learning objective that formulates the slow-learning phase as the meta-task (generic representation and experience rehearsal) and the fast-learning phase as the base task (local adaptation). Across different challenging scenarios (domain/class-incremental learning, online learning with no task boundaries), we show that our framework prepares the slow-learning phase for faster local adaptation and the fast-learning phase to support reduced memory buffer size. Our contributions are three-fold

- We identify three common principles underlying lifelong learning methods. We seek to characterize them in language learning and glean insights into overlooked downsides of the existing method.
- Stemming from the above analysis, we propose a meta-lifelong framework that unifies the three identified principles. Our approach is a direct extension of d'Autume et al. (2019), but explicitly meta-learns the model as a better initialization for local adaptation.
- We conduct extensive experiments to demonstrate that our proposed approach can use the identified three principles to achieve efficient lifelong language learning. Our framework outperforms prior methods while using *100 times* less memory storage. Moreover, we demonstrate that our method can effectively alleviate catastrophic forgetting and negative transfer, closing the performance gap with the multi-task learning upper bound. It can also potentially obtain *22 times* faster inference speed.

6.2 Experimental Setup

6.2.1 Problem Formulation

We consider the lifelong learning setting where a model needs to learn multiple tasks in a sequential order from an online stream of training examples without any task identifier, i.e., the model does not know which task an example comes from during both training and testing. This setup is ubiquitous in practice, as environments consistently evolve without sending an explicit signal (Chaudhry et al., 2019; d'Autume et al., 2019). Formally, during training, the model makes a single pass over the training example stream consisting of $\mathcal{T}$ tasks in an ordered sequence, $\mathcal{D}_{train} = \{\mathcal{D}^1_{train}, \cdots, \mathcal{D}^{\mathcal{T}}_{train}\}$, where $\mathcal{D}^t_{train} = \{(x^t_i, y^t_i)\}_{i=1}^{n_t}$ is drawn from the task-specific distribution $P_t(\mathcal{X}, \mathcal{Y})$ of the t-th task. Also, in a class-incremental learning scenario, the output label space may keep evolving, i.e., $\mathcal{Y}^i \subset \mathcal{Y}^j,\ \forall i < j,\ i, j \in \{1, \cdots, \mathcal{T}\}$. Overall, the goal is to learn a predictor $f_w : \mathcal{X} \rightarrow \mathcal{Y}$ such as a neural network, parameterized by $w \in \mathbb{R}^P$, to minimize the average empirical risk of all $\mathcal{T}$ tasks as defined in Equation 2.5. Notice that while the average empirical risk is most commonly evaluated after the model has seen all tasks, we can also evaluate a specific task at different stages to demonstrate the model's training behavior and evaluate its robustness against catastrophic forgetting and negative transfer.

6.2.2 Benchmarks and Task Sequences

To evaluate our proposed framework, we conduct experiments on text classification benchmark (Section 2.3.1) for class-incremental learning scenario and question-answering benchmark (Section 2.3.2) for domain-incremental learning scenario. The text classification benchmark consists of five datasets: Yelp, AGNews, DBPedia, Amazon, and Yahoo and the question-answering benchmark includes four datasets: SQuAD v1.1, TriviaQA (Web), TriviaQA (Wikipedia), and QuAC. For additional details, see Section 2.3.1 and Section 2.3.2. Following prior work, we consider each dataset a separate task, and the model needs to sequentially learn several tasks of the same category (e.g. all text classification tasks). As pointed out in (McCann et al., 2018; Raffel et al., 2020), many NLP tasks can be formulated as question-answering or text-to-text format; thus, our setup is general. For task sequences, we examine the initial four sequences outlined in Chapter 3 (see Section 3.2.2) for the text classification benchmark and in Chapter 4 (see Section 4.2.2) for the question-answering benchmark. These sequences align with previous work (de Masson D'Autume et al., 2019) to ensure fair comparisons.

We use the following text classification dataset orders for our experimentation

Seq1 Yelp→AGNews→DBPedia→Amazon→Yahoo

Seq2 DBPedia→Yahoo→AGNews→Amazon→Yelp

Seq3 Yelp→Yahoo→Amazon→DBpedia→AGNews

Seq4 AGNews→Yelp→Amazon→Yahoo→DBpedia

We consider the following dataset orders for question-answering

Seq1 QuAC→TrWeb→TrWik→SQuAD

Seq2 SQuAD→TrWik→QuAC→TrWeb

Seq3 TrWeb→TrWik→SQuAD→QuAC

Seq4 TrWik→QuAC→TrWeb→SQuAD

6.2.3 Baselines

We compare our method against then state-of-the-art continual learning methods tailored for lifelong learning from an online stream of data without task identifiers during training and inference. Specifically, we compare with Online EWC in the parameter-regularization approaches (Section 2.4.1), ER and A-GEM with their online variants, from episodic memory-based approaches (Section 2.4.2), and MbPA++ from test-time adaptation-based approaches (Section 2.4.3). We exclude comparisons with optimization-based approaches, which are designed for task-incremental learning scenarios. We exclude comparisons with our G2DfER approach from Chapter 4 and other architecture-based methods (Sodhani et al., 2022). These approaches assume task identifiers during training, a condition not assumed in this chapter. We provide chapter-specific details of the baselines mentioned above below, with a detailed description available in Section 2.4.

- **Finetune** (FT): The model is sequentially fine-tuned on an online data stream and does not utilize any lifelong learning regularization.

- **Online Elastic Weight Consolidation** (Online EWC; Schwarz et al., 2018): An online parameter-based regularization approach that tries to mitigate forgetting by restricting learning to parameters important to previously learned domains.
- **Averaged Gradient Episodic Memory** (A-GEM; Chaudhry et al., 2018b): This approach augments a base model with an episodic memory. This memory preserves examples from previously encountered domains. During training, the stored examples are utilized to impose a constraint on gradients, preventing the model from forgetting knowledge acquired in earlier tasks.
- **Episodic Replay** (ER; Chaudhry et al., 2019): This approach employs an episodic memory to store and replay past experiences during the training process, thereby, mitigating forgetting of knowledge acquired from an online data stream. Following de Masson D'Autume et al. (2019), we retain 1% of the total examples in the replay buffer and perform experience replay by sampling 100 examples from the memory and performing a gradient update after every 10,000 training steps, which gives us a 1% replay rate.
- **Improved Memory-based Parameter Adaptation** (MbPA++; de Masson D'Autume et al., 2019): This approach uses an episodic memory to perform sparse experience replay during training and domain-specific test-time adaptation during inference. Similar to ER, 1% samples are retained in the buffer and the replay rate is set to 1%.

6.3 Principles of Lifelong Language Learning

While different methods have been developed to optimize Equation 2.5, we abstract away from their specific assumptions and instead focus on identifying common principles, among which we stress the following three points that are most relevant to lifelong language learning

6.3.1 Generic Representation

Stemming from transfer learning (Weiss et al., 2016; Ganin and Lempitsky, 2015), a key idea of transferring knowledge across diverse tasks is to learn a generic representation (such as a neural network encoder) that can encode useful information for all tasks. For instance, parameter-based regularization methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Schwarz et al., 2018) add an extra constraint to prevent the model parameter w from drastically deviating when training on new tasks, thereby learning a generic model for old tasks as well. In the language domain, as language models have proven successful in generating highly generic representations for many language understanding tasks (Yogatama et al., 2019; Raffel et al., 2020), both d'Autume et al. (2019) and Sun et al. (2020) propose utilizing a pre-trained language model (Devlin et al., 2019; Radford et al., 2019) to initialize parameters, and further training the model on $\mathcal{D}_{train}$.

6.3.2 Experience Rehearsal

Motivated by the complementary learning systems (CLS) theory (McClelland et al., 1995) that humans rely on episodic memory to store past experiences and conduct experience rehearsal, we

can also retrain lifelong learning models on previously seen tasks to reduce forgetting. While prior methods use memory to define optimization constraints (Lopez-Paz et al., 2017; Chaudhry et al., 2018b; Sodhani et al., 2020), recent works use either stored examples (Sprechmann et al., 2018) or generated synthetic data (Shin et al., 2017; Sun et al., 2020) to perform experience replay. Further, d'Autume et al. (2019) shows that a sparse 1% rate of replaying to learning new examples is sufficient for lifelong language learning, similar to memory consolidation in human learning (McGaugh, 2000).

6.3.3 Task-specific Fine-tuning

In multi-task learning, injecting task-specific parameters and finetuning on individual tasks have proven effective for different language understanding tasks (Houlsby et al., 2019) or even diverse languages (Bapna and Firat, 2019). Prior work (Rusu et al., 2016; Yoon et al., 2018) exploits this idea to expand model parameters for new tasks in a lifelong learning setting. However, all these methods require a task identifier to know when to add new parameters. When no such signal exists, local adaptation (Sprechmann et al., 2018) uses k stored nearest neighbors of each test example to perform extra finetuning at inference time. Recent work (d'Autume et al., 2019; Khandelwal et al., 2020) demonstrates that the sentence embeddings produced by pre-trained models can effectively measure query similarity and that local adaptation can improve performance on text classification, question answering, and language modeling.

6.4 Synergistic Meta-Lifelong Framework (Meta-MbPA)

To motivate our proposed framework, we first review the then state-of-the-art method in Section 2.4.3, MbPA++ (de Masson D'Autume et al., 2019), and show how the above principles (Section 6.3) help us to identify the limitation. Despite its effectiveness, the performance gain of MbPA++ comes at the cost of large memory storage and slow inference speed. The root of this inefficiency is the non-synergistic nature of the method—the three principles are performed independently without close interaction. In particular: (i) the generic representation learned is not optimized for local adaptation, and thus more steps are required for robust performance, (ii) the memory module is selected randomly and lacks a systematic selection method to reduce its size effectively, (iii) local adaptation only utilizes a few neighbors for each testing example, so it is prone to overfitting and negative transfer when memory size is small.

We highlight a discrepancy between training and testing in MbPA++. Specifically, the generic representation is trained on the task loss in Equation 2.19 directly while it makes a prediction *after* the local adaptation at test time. Therefore, the model always overfits the latest task it has seen and never learns how to incorporate experience rehearsal efficiently. According to the CLS theory McClelland et al. (1995), however, human learning systems are complementary—we learn structured knowledge in a manner that allows us to adapt to episodic information quickly. Thus, to resolve the training-testing discrepancy of MbPA++, we change the training goal of generic representation from *how to perform better on the current task* to *how to adapt to episodic memory efficiently*.

Algorithm 1 Meta-MbPA

Procedure Train
Input: online training data stream $\mathcal{D}_{train}$
Output: parameters w, episodic memory $\mathcal{M}$
Initialize w and ϕ with some pre-trained model
for $(x_i, y_i) \in \mathcal{D}_{train}$ **do**
  [Generic Representation] Perform a gradient update on w to minimize Equation 6.1
  if training step mod n_{tr} = 0 **then**
    Sample n_{re} examples from $\mathcal{M}$
    [Experience Rehearsal] Perform a gradient update on w to minimize Equation 6.2
  end if
  Compute $p(x_i)$ according to Equation 6.3
  if Bernoulli($p(x_i)$) = 1 **then**
    Update memory $\mathcal{M} \leftarrow \mathcal{M} \cup (x_i, y_i)$
  end if
end for
Procedure Test
Input: test examples x
Output: predictions $\hat{y}$
for $l = 1, ..., n_l$ **do**
  Sample k examples from $\mathcal{M}$
  [Task-specific Finetuning] Perform a gradient update on w to minimize Equation 2.21
end for
Output prediction $\hat{y} = f_{\tilde{w}_x}(x)$

In particular, we propose an extension of MbPA++ that exploits a meta-learning paradigm to interleave the three key principles: (i) to resolve the training-testing gap, our framework learns a generic representation that is tailored for local adaptation, (ii) to enable robust local adaptation, the memory module uses a diversity-based selection criterion to reduce memory size, (iii) to accommodate small memory, the framework utilizes a coarse local adaptation to alleviate negative transfer. The full framework is outlined in Algorithm 1 and below, we detail how each principle is instantiated systematically.

1. Generic Representation. We incorporate local adaptation into training the generic representation. In particular, we exploit meta-learning by formulating local adaptation as the base task and representation learning as the meta-task. That is, the generic representation is trained to perform well *after* the local adaptation (a.k.a learning to adapt). Thus, for each training example $(x_i, y_i) \in \mathcal{D}_{train}$, we formulate the task loss in Equation 2.19 into a meta-task loss as

$$\begin{aligned} L^{\text{meta}}_{\text{TASK}}(w; x_i, y_i) &= \ell(f_{\tilde{w}_{x_i}}(x_i), y_i) \\ \text{s.t.} \quad \tilde{w}_{x_i} &= w - \alpha \nabla_w L_{\text{LA}}(w; \mathcal{N}_{x_i}) \end{aligned} \tag{6.1}$$

where α is the current learning rate. Notice that differentiation requires computing the gradient of gradient, which modern automatic differentiation frameworks can implement. Intuitively, we first

approximate local adaptation using gradient step(s) and then optimize the adapted network.

2. Experience Rehearsal. With a similar rationale to the meta-task loss, we reformulate the memory replay loss in Equation 2.20 into a meta-replay loss to stimulate efficient local adaptation for all tasks

$$L_{\text{REP}}^{\text{meta}}(w;\mathcal{S}) = \frac{1}{n_{re}} \sum_{x,y\in\mathcal{S}} \ell(f_{\tilde{w}_x}(x), y) \quad \text{s.t.} \quad \tilde{w}_x = w - \alpha\nabla_w L_{\text{LA}}(w;\mathcal{N}_x) \tag{6.2}$$

We use the same replay ratio as in MbPA++ to keep the meta-replay sparse. In addition, we propose a diversity-based selection criterion to determine if a training example $(x_i, y_i) \in \mathcal{D}_{train}$ should be added to the memory module. Here, we exploit the key network g_ϕ to estimate diversity via the minimum distance of x_i to existing memory as:

$$\log(p(x_i)) \propto -\frac{\min_{x,y\in\mathcal{M}} \|g_\phi(x_i) - g_\phi(x)\|_2^2}{\beta}, \tag{6.3}$$

where $p(x_i)$ is the probability of the example being selected and β is a scaling parameter. The intuition is to select examples less similar to the existing memory, thereby covering diverse parts of data distribution. As shown later, the proposed method outperforms the uncertainty-based selection rule (Ramalho and Garnelo, 2019), which picks examples based on the certainty level of the predictor network f_w. This is because local adaptation is prone to negative transfer when the memory $\mathcal{M}$ misrepresents the true data distribution.

3. Task-specific Fine-tuning. With small memory, local adaptation for each testing example is prone to negative transfer. This is because less related memory samples are more likely to be included in $\mathcal{N}_{\boldsymbol{x}_i}$, and the model can easily overfit. Thus, we consider local adaptation with more coarse granularity. For example, we can cluster testing examples and independently conduct local adaptation for each cluster. In our experiments, we find it is sufficient to take this to the extreme, considering all test examples as a single cluster. Note in this case, we may not require g_ϕ to retrieve nearest neighbors. Consequently, we consider the whole memory as neighbors and randomly sample from it to be comparable with the original local adaptation formulation (i.e. same batch sizes and gradient steps). As shown in the next section, it has two benefits: (1) it is more robust to negative transfer, and (2) it is faster when we evaluate testing examples as a group.

6.5 Experiments

Implementation Details. We utilize the pre-trained BERT-base (Wolf et al., 2019) for initializing the encoder network. BERT-base has 12 Transformer layers, 12 self-attention heads, and 768 hidden dimensions (110M parameters). Similar to (d'Autume et al., 2019), we use a separate pre-trained BERT-base for the key network and freeze it to prevent it from drifting while training on a non-stationary data distribution. For text classification, we use the encoded representation

of the special beginning-of-document symbol [CLS] as our key. For question answering, we use the question part of the input to get the encoded representation. We store the input example as its associated memory value for both tasks. Further, we use Faiss (Johnson et al., 2019) for efficient nearest neighbor search in the memory, based upon the key network.

We mainly set hyper-parameters as mentioned in (d'Autume et al., 2019). We use Adam (Kingma and Ba, 2014) as our optimizer, set dropout (Srivastava et al., 2014) to 0.1 and the base learning rate to $3e^{-5}$. For text classification, we use a training batch of size 32 and set the maximum total input sequence length after tokenization to 128. For question answering, we use a training batch of size 8 and set the maximum total input sequence length after tokenization to 384. To deal with longer documents, we set the document stride to 128. We also set the maximum question length to 64.

For Online EWC (Schwarz et al., 2018), we set the regularization strength $\lambda = 5000$ and forgetting coefficient $\gamma = 0.95$. For all models with memory module (Replay, MbPA++, Meta-MbPA), we replay 100 examples for every $10,000$ new examples, i.e., $n_{tr} = 10,000$ and $n_{re} = 100$. As mentioned in (d'Autume et al., 2019), for MbPA++, we set the number of neighbors $k = 32$, the number of local adaptation steps $n_l = 30$, and $\lambda_l = 0.001$. We tune the local adaptation learning rate α for MbPA++ in our re-implementation (MbPA++ Our Impl.) and report the improved numbers as well as their reported numbers in Table 6.1, B.1, and B.2. For text classification, we set $\alpha = 5e^{-5}$ and for question answering we set $\alpha = 1e^{-5}$.

For our framework, Meta-MbPA , unless stated otherwise, we set the number of neighbors $k = 32$ and control the memory size through a write rate $r_{\mathcal{M}} = 1\%$ (the write rate is defined as the fraction of examples retained in episodic memory relative to the total number of seen examples). We use $n_l = 30$ local adaptation steps and perform local adaptation for the whole testing set. We randomly draw $k = 32$ examples from memory and perform a local adaptation step. Through this, the computational cost is equivalent to MbPA++, but we only need to perform the whole process once, while MbPA++ requires conducting local adaptation independently for each testing example. We set $\alpha = 1e^{-5}$ (in Equation 6.1, 6.2), $\beta = 10$ (in Equation 6.3) and $\lambda_l = 0.001$ (in Equation 2.21). All of the experiments are performed using PyTorch (Paszke et al., 2017), which allows for automatic differentiation through the gradient update as required for optimizing the meta-task loss Equation 6.1 and meta-replay loss Equation 6.2.

We use four different orderings of task sequences as in d'Autume et al. (2019) (see Section 6.2.2) and evaluate the model at the end of all tasks. Following prior work, we report the macro-averaged accuracy for classification and F_1 score for question answering. Table 6.1 provides our main results. Notice that the results on the right are not comparable due to different setups. The complete per-task results are available in Appendix B.1.

6.5.1 How much does Meta-MbPA help in alleviating forgetting?

We first compare our framework (Meta-MbPA) with all baselines. Even using only 1% of total training examples as memory, the proposed framework outperforms existing baselines on text classification and question answering. Specifically, while regularization-based methods (A-GEM and Online EWC) perform better than the standard FT model, their performance varies depending on the task ordering and, thus, is not robust. On the other hand, methods that involve local adaptation (MbPA++ and ours) perform consistently better for all orderings. In particular, **our**

Order	FT	Online EWC	A-GEM†	Replay	MbPA++†	MbPA++ (Our Impl.)	Meta-MbPA (1%)	MTL (100%)	MTL (1%)	LAMOL$^{0.2}_{TASK}$‡
Text Classification										
i.	35.5	43.8	70.7	63.4	70.8	75.3	**77.9**	-	-	76.7
ii.	44.8	49.8	65.9	73.0	70.9	74.6	**76.7**	-	-	77.2
iii.	42.4	59.5	67.5	65.8	70.2	75.6	**77.3**	-	-	76.1
iv.	28.6	52.0	63.6	74.0	70.7	75.5	**77.6**	-	-	76.1
Avg.	37.8	51.3	66.9	69.1	70.6	75.3	**77.3**	78.9	50.4	76.5
Question Answering										
i.	60.9	58.0	56.1	62.3	62.0	63.3	**64.8**	-	-	-
ii.	57.3	57.2	58.4	61.3	62.4	63.5	**65.3**	-	-	-
iii.	47.0	49.5	52.4	58.3	61.4	61.6	**64.4**	-	-	-
iv.	61.0	58.7	57.9	62.9	62.4	62.4	**65.0**	-	-	-
Avg.	56.6	55.9	56.2	61.2	62.1	62.7	**64.9**	68.6	44.1	-

Table 6.1: **Accuracy and F_1 scores for text classification and question answering, respectively.** Methods that use the defined lifelong learning setup in Section 6.2.1 are listed on the left. Where applicable, all methods use $r_{\mathcal{M}} = 100\%$ memory size unless denoted otherwise. The best result for lifelong learning methods is made **bold**. † Results obtained from (d'Autume et al., 2019). ‡ LAMOL (Sun et al., 2020) is not directly comparable due to their different problem setup where task identifiers are available. Our framework, Meta-MbPA, outperforms MbPA++ and narrows the performance difference with MTL (100%) while employing just 1% episodic memory size.

Model / Task	$r_{\mathcal{M}} = 1\%$		$r_{\mathcal{M}} = 10\%$	
	Text Classification	Question Answering	Text Classification	Question Answering
MbPA++	73.1	61.9	73.5	62.6
Meta-MbPA	77.3	64.9	78.0	65.5
MTL (subsampled)	50.4	44.1	70.5	56.2

Table 6.2: **Performance of models using different memory sizes.** We report accuracy and F_1 scores for text classification and question answering, respectively. MTL (subsampled) is trained on subsampled training data, equivalent to only performing local adaptation without training the generic representation. Notice that this variant of MTL is not an upper-bound as it uses fewer training samples. In summary, our framework Meta-MbPA demonstrates a more efficient utilization of the episodic memory module compared to existing methods.

framework improves over MbPA++ while using 100 times less memory, demonstrating the effectiveness of the proposed approach. We then compare lifelong learning methods to the multitask model MTL, which can be viewed as an upper bound on performance when there is no significant negative transfer between tasks. As shown in Table 6.1, there is still a non-trivial gap between MbPA++ and MTL, albeit MbPA++ stores all training examples as memory. Our framework narrows the gap while using smaller memory.

	Replay	MbPA++	Meta-MbPA
Text Classification			
Random	69.2	73.1	76.8
Diversity	69.1	73.0	77.3
Uncertainty	65.4	41.2	62.7
Forgettable	62.7	50.5	61.8
Question Answering			
Random	61.2	61.9	63.8
Diversity	61.5	62.2	64.9
Uncertainty	56.1	50.4	54.2
Forgettable	59.7	52.1	57.5

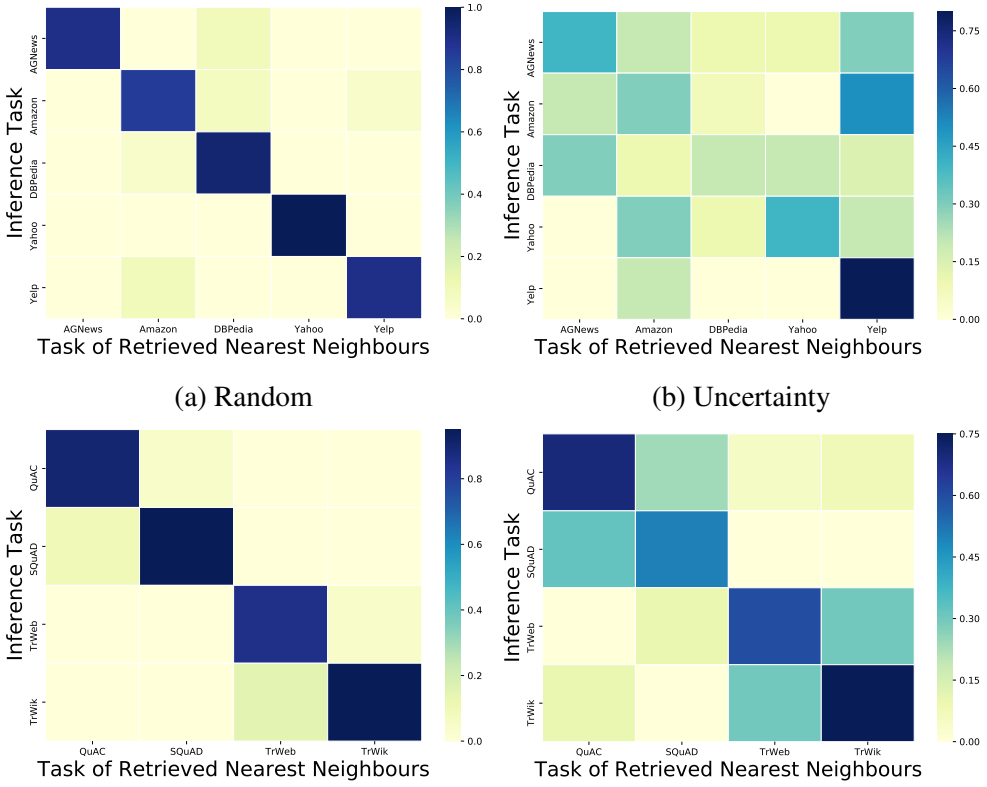
Table 6.3: **Performance of models using different memory selection criteria.** "Uncertainty" utilizes model's confidence level (Ramalho and Garnelo, 2019). "Forgettable" picks examples according to forgetting events (Toneva et al., 2019). We tune hyper-parameters that result in $r_{\mathcal{M}} = 1\%$ memory size for all methods. Memory selection criteria significantly affect performance; the proposed diversity method outperforms all other criteria.

6.5.2 Analyzing the (episodic) memory efficiency of Meta-MbPA

In Table 6.1, MbPA++ uses 100% memory while our framework only uses 1% memory. To test memory efficiency, we present results for models using equivalent memory resources in Table 6.2. The results demonstrate that the performance of MbPA++ degrades significantly as the memory size decreases. It is then natural to ask if memory alone is sufficient to obtain good performance. We thus compare with MTL trained on sub-sampled training data, equivalent to only performing local adaptation without training the generic representation. Notice that this variant of MTL is *not* an upper bound as it uses fewer resources. Our method significantly outperforms it, showing that the meta generic representation in our method is also crucial to achieving good performance. These results validate that **our framework can utilize the memory module more effectively than existing methods**.

6.5.3 Memory selection rule: How is episodic memory populated?

We then study the source of improvement of our method. In particular, we show that the prior method is prone to negative transfer. We first conduct a case study of the memory selection rule to see this. We consider two popular paradigms in active learning (Donmez et al., 2007), namely the diversity-based method that picks the most representative examples and the uncertainty-based method that picks the most unsure examples. In particular, we compare four selection criteria belonging to these two categories: random selection, our proposed diversity-based method in Equation 6.3, and two uncertainty-based methods (Ramalho and Garnelo, 2019; Toneva et al., 2019). Random selection is a diversity-based method since it picks examples representing the true data distribution. As shown in Table 6.3, **we observe that the choice of memory selection**



(a) Random (b) Uncertainty

(c) Diversity (d) Forgettable

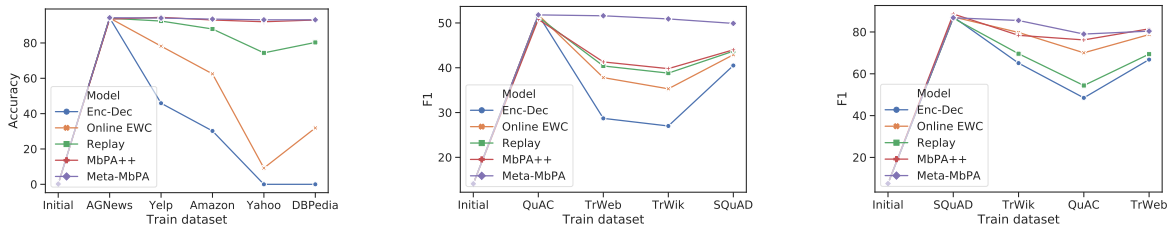
Figure 6.1: **Proportions of a source of neighbors used in local adaptation for each task when different memory selection rule is used**, e.g., 10% of neighbors retrieved for Yelp belong to Amazon. Numbers in each row sum to 1. The top two figures are for text classification (5 tasks) while the bottom two are for question answering (4 domains). For task/ domain ordering, check Seq1 in Section 6.2.2. Overall, uncertainty-based methods result in more examples from other tasks being used as nearest neighbors, compared to diversity-based methods.

criteria impacts performance. While the proposed diversity method outperforms random selection, the two uncertainty-based methods perform *worse* than the random baseline, consistent with similar findings reported in d'Autume et al. (2019).

We seek an explanation for this phenomenon and visualize the heat maps in Figure 6.1 to show which tasks each testing example's retrieved neighbors come from during the local adaptation phase. Ideally, the model should always use neighbors from the same task, and the heat map should be diagonal. **Compared to diversity-based methods, we observe that more examples from other tasks are used as nearest neighbors when models use uncertainty-based methods**. This is because the selected uncertain examples are usually less representative of the true distribution and could be outliers. Thus, the resulting memory does not have good data distribution coverage, and no similar examples exist for certain test examples. Consequently, fewer related examples from other tasks are used for the local adaptation, which causes negative transfer. This is verified in Table 6.4, where models without local adaptation outperform their locally adapted counterparts. More importantly, Meta-MbPA obtains much smaller performance gaps, indicating that it is more

Model / Task	Uncertainty		Forgettable	
	Text Classification	Question Answering	Text Classification	Question Answering
Meta-MbPA	62.7	54.2	61.8	57.5
w/o LA	65.8	55.8	67.9	59.2
MbPA++	41.2	50.4	50.5	52.1
w/o LA	65.4	56.1	68.4	59.2

Table 6.4: **Performance of models using the uncertainty-based memory selection methods (correspond to Table 6.3).** "LA" refers to local adaptation.



(a) Text Classification-AGNews (b) Question Answering-QuAC (c) Question Answering-SQuAD

Figure 6.2: **Catastrophic forgetting of the first dataset as training progresses.** "Enc-Dec" refers to the FT baseline. Complete results in Appendix B.3

robust to negative transfer. We further verify this in the following section.

6.5.4 Trade-off: Catastrophic forgetting (stability) vs. Negative interference (plasticity)

We first verify the models' robustness to catastrophic forgetting. As shown in Table B.1 and B.2 (Appendix B.1), the standard FT(or Enc-Dec) model performs poorly on previously trained tasks, indicating the occurrence of catastrophic forgetting. While all baselines can somewhat alleviate the forgetting, our framework achieves the best performance on previously learned tasks. We also evaluate the model's performance on the first task as it continues to learn more tasks. Figure 6.2 illustrates how each model retains its previously acquired knowledge as it learns new knowledge. Our framework is consistently better than the baselines at mitigating forgetting.

In addition, as prior works have shown, transferring from diversely related sources can hurt performance in the target (Ge et al., 2014; Wang and Carbonell, 2018), we study if transferring from multiple tasks learned in the past can induce negative transfer (or negative interference), which is often overlooked in existing studies on lifelong learning. Table 6.5 shows the averaged results on the last task in each task ordering (see Appendix B.1 for complete results). Surprisingly, compared to the FT baseline, MbPA++ performs *worse* on the last task despite its improved macro-averaged performance (Table 6.1). This suggests that while it is robust to catastrophic

	FT	Replay	MbPA++	Meta-MbPA
class.	82.1	81.8	78.6	82.1
QA	72.6	72.7	70.7	72.1

Table 6.5: **Average performance on the last task across all four task orderings.**

Model / Task	$r_{\mathcal{M}} = 1\%$		$r_{\mathcal{M}} = 50\%$	
	Text Classification	Question Answering	Text Classification	Question Answering
Meta-MbPA	77.3	64.9	78.2	66.1
w/o Meta	73.1	58.5	74.0	59.6
w/o MS	76.8	63.8	78.1	66.1
w/o LA	75.9	62.0	75.8	62.1

Table 6.6: **Ablation Study on different memory size.** "Meta" refers to the proposed meta optimization in Eq.equation 6.1 and equation 6.2."MS" denotes memory selection based on Eq.equation 6.3. "LA" refers to local adaptation.

forgetting, MbPA++ fails to utilize prior knowledge to benefit later tasks and thus is prone to negative transfer. Apart from some practical bottlenecks such as limited model capacity, local adaptation is a critical factor of negative transfer as Replay[1] outperforms MbPA++ in Table 6.5. Intuitively, this shows that since Replay already performs well on the last task, further using local adaption can overfit and hurt the performance. On the other hand, the proposed method is trained to learn a more robust initialization for adaptation and uses a coarse adaptation that is less prone to negative transfer. Therefore, it outperforms MbPA++ and closes the gap with FT on the last task, consistent with results in Table 6.4. These experiments illustrate a trade-off between catastrophic forgetting and negative transfer, such that more adaptations are desired for earlier tasks while less is better for later tasks. **While prior studies focus on catastrophic forgetting only, we are the first to show the importance of balancing the trade-off to avoid both negative effects**.

6.5.5 Ablation study: Investigating the effectiveness of CLS theory

We report the results of the ablation study in Table 6.6 and analyze the effects of the three components in our framework subject to different memory sizes. First, we observe that the model without the meta-learning optimization performs the worst, which shows the importance of learning a generic representation tailored for local adaptation. More importantly, Meta-MbPA achieves worse performance without any local adaptation step. This demonstrates that learning the generic representation alone is insufficient and that the meta-learning mechanism and local adaptation are complementary, which mimics the complementary human learning systems in the CLS theory. Finally, while the diversity-based memory selection rule contributes

[1]Replay is equivalent to MbPA++ without local adaptation.

to the performance gain when we use a small memory module, it becomes less effective as the memory size increases. This is expected since the memory distribution can represent the true data distribution with a larger capacity. Thus it demonstrates that the proposed methods mostly contribute to robustly reducing the memory sizes for better efficiency. **Overall, these results validate the effectiveness of each component and highlight the importance of complementary lifelong learning systems. This is the first work to formulate the slow learning of structured knowledge as a meta-task and the fast learning from episodic memory as a base task.**

6.5.6 Analyzing the inference efficiency of Meta-MbPA

The ordinary local adaptation requires customized gradient updates for each testing example, and thus it is notoriously slow. Using 1 Nvidia Tesla V100 GPU and 128 GB of RAM, it takes 66.6 hours and 89.3 hours to evaluate test classification and question answering, respectively. On the other hand, we use coarse local adaptation in our method, which uses the same updates for all testing examples. Consequently, it takes 2.9 hours and 4.2 hours for our method to finish the evaluation process, achieving a maximum of 22 times speedup. Notice that our method will obtain a similar inference speed in a purely online learning setup as MbPA++. In addition, we hypothesize that using a different granularity (e.g., clustering testing examples) is beneficial for more conflicting tasks, as it can balance the trade-off between overfitting to nearest neighbors of small memory and performing more sample-specific adaptation for each test example. We leave this exploration for future work.

6.6 Discussion

In this chapter, we identify three principles underlying different lifelong language learning methods and show how to unify them in a meta-lifelong framework, Meta-MbPA. Our experiments demonstrate the effectiveness of the proposed framework on text classification and question answering tasks. We report new state-of-the-art results while using 100 times less memory space. These results illustrate that it is possible to achieve efficient lifelong learning by establishing complementary learning systems. Our analysis also shows that negative interference is an overlooked factor that could cause sub-optimal performance, and we highlight the importance of balancing the trade-off between catastrophic forgetting and negative interference for future work.

Part III

Data: Limited Labeled & Unlabeled

Chapter 7

Limited Labeled Data: Lifelong Semi-Supervised Learning for Updating Transformer Memory

7.1 Overview

In this chapter, we work on a lifelong (semi-supervised) learning setting that deals with continuous learning from an unlabeled data stream (as opposed to a labeled data stream considered in the previous chapters 3, 4, 5, 6). To motivate this setting, we consider Differentiable Search Indices (DSIs; Tay et al. (2022)), a new modeling paradigm for information retrieval tasks using sequence-to-sequence learning. Specifically, DSIs leverage Transformer memory (Vaswani et al., 2017) to encode all of the information in a corpus of documents and then use that memory to answer user queries directly, thereby simplifying the retrieval process. DSIs achieve this functionality by jointly optimizing for indexing (or memorization) and retrieval tasks. The indexing task requires learning a mapping from document content to its identifier, typically represented by integers or short strings (document identifiers, abbreviated *docids*). Then, the retrieval task necessitates mapping user queries to relevant docids.

DSI constitutes a well-suited candidate for lifelong learning; the indexing process is continuous and cumulative. In other words, as new documents arrive, the neural indexer needs updation. Index construction using a DSI involves training a Transformer model. Therefore, the model must be re-trained from scratch every time the underlying corpus is updated, thus incurring prohibitively high computational costs and necessitating continual learning capabilities. Furthermore, one may not have access to ground-truth queries corresponding to the incoming documents; nevertheless, the indexer should still be able to answer input queries for already and newly-indexed documents. This application scenario motivates designing lifelong learning systems capable of continuously learning from an unlabeled data stream, like humans, who continually learn in an unsupervised manner from our surroundings rather than relying on direct supervision.

Toward this end, we propose *DSI++ (DSI + new documents)*, a lifelong learning challenge for DSI to incrementally index new documents while maintaining the ability to answer user queries related to both previously and newly indexed documents. To enable DSI++, we introduce novel

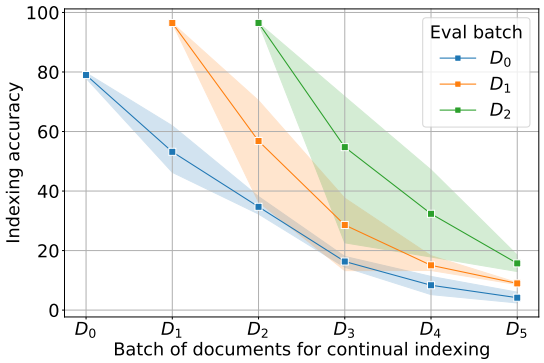


Figure 7.1: Indexing accuracy of D_0, D_1, and D_2 document corpora visualized as we continuously index new documents (averaged over three runs). We observe that continual indexing of new documents leads to severe forgetting of the previously memorized documents.

benchmarks constructed from the existing Natural Questions (Kwiatkowski et al., 2019) and MS MARCO (Nguyen et al., 2016) datasets, simulating the continual addition of documents to the system. To our knowledge, there is no prior work studying incremental learning for DSI.

A naive solution for DSI++ is to continuously fine-tune the model with an indexing objective over new documents. However, Figure 7.1 shows that continual indexing of new documents leads to catastrophic forgetting of the previously memorized documents (more details in §7.2.1), a common phenomenon in connectionist networks wherein learning of the new concepts interferes with the previously acquired knowledge (McCloskey and Cohen, 1989; French, 1999). Furthermore, with new documents, we do not get ground-truth queries. Therefore, forgetting from continual indexing of new documents and continuous learning from unlabeled data are two key challenges to overcome for successfully implementing a DSI++ system. Therefore, we introduce a generative memory to sample pseudo-queries for already indexed documents and use them to alleviate forgetting of the retrieval task during incremental indexing of the new documents. Also, generative memory enables lifelong semi-supervised learning of the retrieval task by generating pseudo-queries for an incoming batch of new documents. Our main contributions in this chapter can be summarized as follows:

- We introduce DSI++, a continual learning challenge for the recently proposed Differentiable Search Indices (DSI) paradigm. To enable DSI++ evaluations, we create two benchmarks based on existing Natural Questions and MS MARCO datasets.
- To understand the severity of the forgetting phenomenon across multiple scenarios, we analyze a suite of pre-trained models (T5-Base, T5-Large, T5-XL) and different document identifier representations (unstructured atomic, naively structured, and semantically structured).
- We propose a generative memory-based experience rehearsal approach to alleviate explicit forgetting during continual indexing and improve the average Hits@1 by $+25.0\%$ and Hits@10 by $+21.1\%$ over competitive baselines for MS MARCO and NQ, respectively.

7.2 DSI++: Continual learning challenge for DSI

7.2.1 Problem Formulation

We focus on a setup where we receive an initial corpus of documents, $D_0 = \{d_1, \cdots, d_n\}$, and user queries corresponding to a subset of them, $R_0 = \{< q_j, j >, \forall j \in \mathcal{Y}_D\}$, where $D \subset D_0$. DSI paradigm involves two tasks: (i) a *memorization task* where the goal is to learn an indexer $f_w : \mathcal{X} \rightarrow \mathcal{Y}$, a text-to-text model like T5 (Raffel et al., 2020) parameterized by $w \in \mathbb{R}^P$, that takes document tokens ($x \in \mathcal{X}$) as input and maps it to a document identifier (docid) $j \in \mathcal{Y}$, and (ii) a *retrieval task* where the goal is to use the same indexer f_w to directly map a user query q to a relevant docid $j \in \mathcal{Y}$. Following Tay et al. (2022), two different prompts are used to differentiate between these tasks.

Tay et al. (2022) discusses several variants for representing docids – *unstructured atomic* and *structured string* docids, where each document is assigned a unique token and tokenized string, respectively. Under the unified text-to-text format, both of the above tasks are cast as generation tasks, i.e., decoding one unique token (unstructured atomic) or decoding a tokenized string sequentially, one token at a time (naively/ semantically structured).

In the dynamic corpus scenario, we simulate the arrival of new documents by updating the initial corpus D_0 with a sequence of batches $D_1, D_2, ...$, with D_1 arriving first, followed by D_2, and so on. In DSI++, we have access to the new batch of documents D_i, but we do not have any queries related to these documents. The goal is to learn a DSI++ system that incrementally indexes $D_1, D_2, \cdots$ in f_w while being able to answer queries related to previously as well as additionally indexed documents.

7.2.2 Benchmark for DSI++

To enable research on DSI++, we introduce two benchmarks constructed from the Natural Questions (NQ; Kwiatkowski et al. (2019)) and MS MARCO (Nguyen et al., 2016) datasets. The NQ dataset consists of Wikipedia articles and corresponding natural language questions. Similar to Tay et al. (2022), we consider Wikipedia articles for memorization and the retrieval task as identifying the Wikipedia article that answers the given question. We use the original NQ train split to construct train(80%)/ validation(20%) splits and use NQ validation as a test split for our setup. We randomly sample $50K$ unique articles to constitute the initial D_0 corpus. Next, we construct five corpora ($D_1, \cdots, D_5$), each containing $10K$ unique articles, to add them to the DSI model sequentially. Corresponding to articles in each of these corpora, we filter queries from original NQ train/ validation splits to construct $R_i^{train}, R_i^{val}, R_i^{test}$ ($\forall i \in \{0, \cdots, 5\}$) splits. We use R_0 to train the DSI model for the retrieval task and use R_i^{test} to evaluate on previously and newly indexed articles.

The full MS MARCO dataset has approx. $500K$ passage-query training pairs and $6,980$ validation pairs. Like the benchmark created from the NQ dataset, we randomly sample $50K$ unique passages to constitute the initial D_0 corpus and five more corpora, each with $10K$ passages. See Table C.1 (in the Appendix C.1) for exact dataset statistics for NQ and MS MARCO.

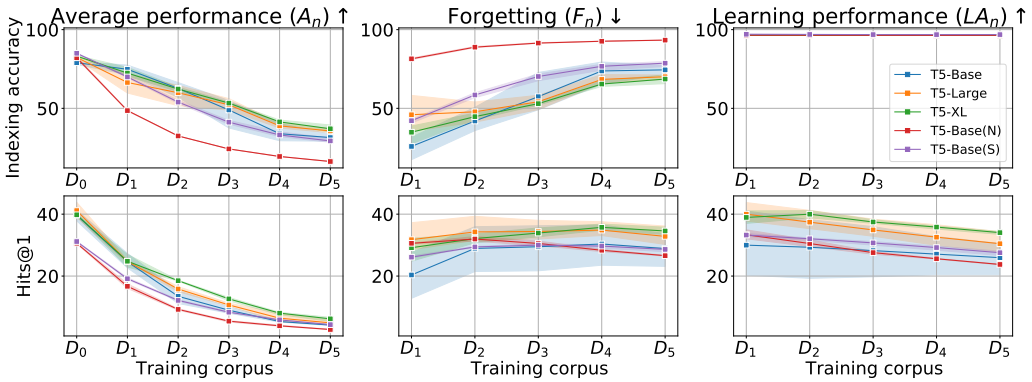


Figure 7.2: Systematic study about forgetting and forward transfer when incrementally indexing new corpus of documents across different model sizes (T5-Base, T5-Large, T5-XL) and docid representations. We use atomic docids by default and denote (N)/(S) for naively/ semantically structured docids. ↑ indicates higher is better, ↓ indicates lower is better. We observe that the average A_n and learning LA_n performance improves by increasing the model scale. However, forgetting F_n is severe across all model scales. Moreover, we observe that naively structured docids, T5-Base(N), underperform unstructured atomic docids, T5-Base, across all metrics - indexing accuracy, Hits@1, (see Figure C.1 in Appendix C.2 for Hits@10 results). Imbuing the docid space with a semantic (S) structure alleviates the forgetting compared to an arbitrary/ naive (N) structure.

7.2.3 Evaluation Metrics

For DSI evaluation, we report *indexing accuracy* for the memorization task and *Hits@k* ($k \in \{1, 10\}$) metric for the retrieval task. Indexing accuracy and Hits@k are defined as the proportion of documents correctly memorized, and the proportion of correct documents ranked in the top k predictions, respectively. We formally define metrics to summarize the model performance as we incrementally index new documents. Let $P_{n,o}$ denote the performance (e.g., indexing accuracy) on corpus D_o after training on corpus D_n. Following prior works (Lopez-Paz et al., 2017; Riemer et al., 2019), we compute the **average performance** (A_n), **forgetting** (F_n) and **learning performance** (LA_n) metrics after indexing the corpus D_n (see Section 2.2 for more details).

7.2.4 Case study: Catastrophic Forgetting and Forward Transfer

After introducing the DSI++ problem setup, benchmark, and evaluation metrics, we study the behavior of the DSI model as new documents are continuously added to the system. Concretely, we are interested in investigating the following for continual training of the DSI model with an indexing objective on new documents

(Q1) How severe is the forgetting of the originally indexed documents?

(Q2) How does continual updating of the DSI model over a sequence of corpora affect the forgetting?

(Q3) How does the updated DSI model perform on newly indexed documents, especially the retrieval task?

(Q4) How do different docid representation strategies affect forgetting?

(Q5) How does the DSI model scale affect forgetting?

Figure 7.2 visualizes results (averaged over 3 runs) on the validation split of DSI++ and helps us convincingly answer the above questions.

Forgetting (or negative backward transfer). From Figure 7.2, we see that the T5-Base model with atomic docid representation (blue line plots) undergoes significant forgetting. This trend holds across all DSI evaluation metrics - indexing accuracy, Hits@1, and Hits@10 (see C.1 in Appendix C.2). For the originally indexed D_0 corpus, indexing accuracy and Hits@1 drop by approx. 25 and 20 points, respectively. Further, as we continue indexing the sequence of $D_1, \cdots, D_5$ corpora, we see that forgetting becomes even more severe. For example, after continually indexing the D_5 corpus, F_5 (forgetting) for indexing accuracy increases to 75. These results provide evidence to answer (Q1) & (Q2) that the DSI model undergoes severe forgetting under continual indexing of new documents.

Forward transfer. To answer (Q3), we visualize the learning performance (LA_n) for all DSI metrics for sequential indexing. From Figure 7.2, we see LA_n increases in indexing accuracy, suggesting that the DSI model is plastic enough to index new documents. However, from Figure 7.2, we see a declining trend for Hits@1. Due to the continuous indexing updates, the underlying DSI model drifts and becomes less effective for the retrieval task. These findings hint at an approach that replays indexing and retrieval tasks during continual learning (hence our proposed method in §7.3).

Docid representations. For studying (Q4), we consider unstructured atomic, naively(N) structured, and semantically(S) structured docid representations. From Figure 7.2, we see that T5-Base(N) underperforms T5-Base. For example, the average performance A_0 for the Hits@1 metric is approx. 30 and 39 for naive and atomic docids, respectively. Furthermore, as the naively structured approach treats unstructured docids as tokenizable strings as opposed to dedicated unique tokens in the case of atomic docids, they are relatively more prone to interference from new docids (see F_n subplot for indexing accuracy). Imbuing semantic structure to the naive docid space helps to reduce forgetting however still underperforms unstructured atomic docids.

Model scale. As atomic docids are superior to naive docids, we only consider atomic docids for answering (Q5). From Figure 7.2, we observe that larger models outperform their smaller counterparts in terms of the average performance A_n and the learning performance LA_n (T5-XL $>$ T5-Large $>$ T5-Base). However, empirically we report that forgetting F_n is severe across all model scales, without any clear best performer, and therefore, we focus on T5-Base for the rest of our experimentation.

7.3 Lifelong Semi-Supervised Learning with Generative Memory

DSI paradigm consists of two tasks – memorization and retrieval. In this section, we focus on the forgetting phenomenon that arises from the continual indexing of new documents, specifically

in the context of the retrieval task. Through our systematic study (in §7.2.4), we show that irrespective of the model scale and docid representations, DSI models undergo severe forgetting. Moreover, we observe that the learning performance LA_n keeps declining for the retrieval task (see Figures 7.2 and C.1 for Hits@1 and Hits@10, respectively). This observation suggests that as we continuously update the DSI model with the indexing objective, the model forgets the retrieval task. In DSI, both memorization and retrieval tasks return docid for input. By setup, we can assume access to the contents of the previous documents and continue indexing both old and new documents to reduce forgetting of the retrieval task. However, in Figure 7.3, we see that the model still undergoes forgetting (more discussion in §7.4).

Episodic memory. Memory-based approaches (see Section 2.4.2) for continual learning use a subset of previous task data to regularize future task learning while minimizing forgetting. Experience Replay (ER; Chaudhry et al., 2019) is one such approach that samples previous task data (from episodic memory) for multi-task learning with the current task. ER outperforms other sophisticated memory-based approaches like Gradient Episodic Memory (GEM; Lopez-Paz et al., 2017), Averaged GEM (Chaudhry et al., 2018b), even with a sparse replay. Based upon this line of work, one approach for DSI++ is to retain ground-truth queries for the retrieval task in episodic memory and use them for multi-task learning with the incremental indexing task. However, in DSI++, we do not have access to ground-truth queries for an incoming batch of new documents. Even if one retains queries for the initial D_0 corpus, we show in Table 7.1 that such a method suffers from forward transfer to newly indexed documents.

Generative memory. Recent years have seen significant progress in the capabilities of the generative language models (Raffel et al., 2020; Brown et al., 2020). Motivated by the success of these models and the in-applicability of the episodic memory for DSI++, we pose a question: *instead of retaining the ground-truth queries, can we learn a parametric model to generate such queries given a document*? Concretely, we propose to train a *query generator model* to sample queries for previously seen documents and supplement them during incremental indexing. Since we use the generator model to sample queries for sparse experience replay, hence our proposed method – *generative memory*. Moreover, generative memory is also used to generate pseudo-queries for the incoming batches of new documents, thus, enabling continual semi-supervised learning of the retrieval task.

7.4 Experiments

In this section, we revisit some of the questions (Q1)-(Q3) raised in our case study (see Section 7.2.4) to investigate the effectiveness of our proposed generative memory-based approach. To answer these questions, in Table 7.1, we report the performance of the DSI model on D_0 (to study the forgetting phenomenon) and D_1 corpora (to answer forward transfer question) after continual indexing on D_1 for both NQ and MS MARCO datasets. In Figures 7.3 and C.2 (NQ) and Figure C.3 (MS MARCO), we report overall performance across DSI metrics as we continuously update the model with the sequence of five corpora ($D_1, \cdots, D_5$).

Implementation details. We utilize the pre-trained T5-Base (Raffel et al., 2020) model to initialize all models and randomly initialize the additional parameters for atomic docid tokens.

While indexing D_0 corpus, we train all the models for a maximum of 1M steps with a warmup of 100K steps. During continual indexing of other corpora, we train for a maximum of 100K steps with a warmup of 100 steps. For the rest of the hyper-parameters, we follow Tay et al. (2022) – set a learning rate to 0.001, batch size to 128, and input sequence length to 32. We evaluate models after every 5K steps and retain the checkpoint yielding the best performance. For the initial training with D_0 corpus, we co-train on indexing and retrieval tasks; therefore, we use the average of all DSI metrics (indexing accuracy, Hits@1, and Hits@10) for model selection. For the continual learning experiments, we have access to only indexing accuracy for all involved corpora, so we use it for model selection. To train a parametric model for generative memory, we utilize the retrieval dataset R_0, which corresponds to the D_0 corpus. We set the maximum sequence length for document contents to 1024, the target length for generated queries to 32, batch size to 128, train for a maximum of 100K steps, and use BLUE for model selection. We use beam decoding to generate pseudo-queries. We tune the learning rate amongst $\{0.001, 0.0005\}$ and linear warmup amongst $\{1K, 10K\}$. For all our experiments, we use the T5X (Roberts et al., 2022) framework along with 4-8 TPUv4 chips to train the models.

Methods. We compare our proposed generative memory-based approach with the following methods:

- **continual indexing, cl($\mathbf{D_n}$).** The DSI model is sequentially fine-tuned with the indexing objective on the incoming corpus of documents D_n.
- **continual indexing with all seen documents, cl($\mathbf{U_n}$).** The DSI model is continuously fine-tuned with the indexing objective on the updated corpus U_n ($= \bigcup_{i=0}^{n} D_i$). Also, we sample documents from old ($\bigcup_{i=0}^{n-1} D_i$) and new (D_n) corpora in equal proportion.
- **continual experience replay using generative memory, genmem($\mathbf{D_n}$).** In this method, the proposed generative memory model is used to sample pseudo-queries corresponding to the corpus D_n. Next, these pseudo-queries are used for (sparse) experience replay of the retrieval task samples.
- **continual experience replay using episodic memory, epsmem($\mathbf{D_n}$).** In this method, ground-truth queries corresponding to the D_n^{th} corpus are used for experience replay of the retrieval task.
- **train from scratch, (no cl).** In this method, the DSI model is trained from scratch every time a new corpus, D_n, is added. This method corresponds to a non-continual learning setup and is computationally expensive.

7.4.1 Does generative memory alleviate forgetting of old documents?

In Table 7.1, for the NQ dataset, we report Hits@1 to be 35.9 for the model after training on D_0. We see that continually indexing both D_0 and D_1 corpora (cl(U_1) - 28.9), significantly reduce forgetting the retrieval task (Hits@1) over just indexing the new corpora D_1 (cl(D_1) - 19.2). Next, we look at the performance of the ER approaches when augmented with the continual indexing of all documents. We see that both episodic memory (cl(U_1)+epsmem(D_0) - 22.9), and generative memory (cl(U_1)+genmem(D_0) - 26.0) reduce forgetting compared to cl(D_1)

Added corpus	Method	Eval corpus = D_0 (Catastrophic forgetting) Index acc.	Hits@1	Hits@10	Eval corpus = D_1 (Forward transfer) Index acc.	Hits@1	Hits@10
Natural Questions (NQ) – $\|D_0\| = 50K$, $\|D_1\| = 10K$							
D_0	-	$81.8_{1.2}$	$35.9_{2.2}$	$66.9_{0.9}$	-	-	-
D_1	cl(D_1)	$52.4_{3.5}$	$19.2_{3.9}$	$43.6_{5.7}$	$96.5_{0.0}$	$31.7_{6.4}$	$55.6_{4.9}$
	cl($U_1 = D_0 \cup D_1$)	$78.2_{0.5}$	$28.9_{8.9}$	$59.0_{7.9}$	$91.8_{0.4}$	$34.0_{2.4}$	$60.2_{1.9}$
	cl(U_1)+epsmem(D_0)	$77.8_{0.5}$	$22.9_{1.5}$	$51.4_{0.5}$	$93.1_{0.0}$	$13.1_{2.1}$	$39.6_{3.1}$
	cl(U_1)+genmem(D_0)	$77.8_{0.3}$	$26.0_{6.9}$	$54.9_{8.3}$	$93.0_{0.5}$	$8.6_{4.8}$	$31.6_{11.8}$
	cl(U_1)+epsmem(D_1)	$53.2_{3.1}$	$7.7_{2.1}$	$26.0_{2.0}$	$96.5_{0.0}$	$48.3_{2.3}$	$70.7_{1.9}$
	cl(U_1)+genmem(D_1)	$50.1_{0.8}$	$7.0_{1.2}$	$23.1_{2.2}$	$96.5_{0.0}$	$57.7_{1.5}$	$76.7_{0.9}$
	cl(U_1)+genmem(U_1)	$78.2_{0.3}$	$18.4_{2.8}$	$47.5_{3.9}$	$92.1_{0.3}$	$48.5_{6.1}$	$73.8_{2.9}$
	train from scratch	$78.7_{0.6}$	$35.9_{1.4}$	$66.4_{0.0}$	$79.2_{0.3}$	$32.9_{1.8}$	$63.9_{1.2}$
MS MARCO – $\|D_0\| = 50K$, $\|D_1\| = 10K$							
D_0	-	$99.4_{0.2}$	$78.2_{0.2}$	$95.0_{0.1}$	-	-	-
D_1	cl(D_1)	$46.7_{18.6}$	$68.0_{2.0}$	$87.3_{1.3}$	$99.8_{0.0}$	$36.1_{9.5}$	$65.8_{6.9}$
	cl(U_1)	$99.4_{0.0}$	$76.5_{0.7}$	$94.2_{0.3}$	$99.8_{0.0}$	$35.3_{4.1}$	$64.4_{3.3}$
	cl(U_1)+genmem(U_1)	$99.3_{0.1}$	$73.7_{0.2}$	$93.9_{0.3}$	$99.8_{0.0}$	$80.6_{1.0}$	$95.5_{0.1}$
	train from scratch	$99.5_{0.0}$	$75.0_{0.2}$	$93.9_{0.1}$	$99.6_{0.0}$	$73.4_{1.3}$	$93.4_{0.9}$
MS MARCO (full) – $\|D_0\| = 8M$, $\|D_1\| = 842K$							
D_0	-	99.4	16.3	46.8	-	-	-
D_1	cl(D_1)	0.0	0.1	0.6	97.9	18.2	40.5
	cl(U_1)+genmem(U_1)	20.4	7.3	31.3	86.6	31.6	65.8

Table 7.1: Comparing performance on incremental indexing of D_1 corpus across different methods - cl(D_1): continue fine-tuning with indexing task on D_1, cl(U_1): continue fine-tuning on the updated corpus U_1, cl(U_1)+epsmem(D): continual indexing of U_1 along with ER of queries for D, cl(U_1)+genmem(D): continual indexing of U_1 along with ER of pseudo-queries for D. We notice that continually indexing the updated corpus cl(U_1) results in less forgetting of D_0 compared to indexing only the new corpus cl(D_1), observed in both NQ and MS MARCO datasets. Next, ER with either D_0 or D_1 hurts forward transfer or forgetting. Our proposed approach of augmenting pseudo-queries for all documents along with continual indexing, cl(U_1)+genmem(U_1), alleviates forgetting of D_0 corpus and improves forward transfer to D_1 corpus. We also show that our proposed solution reduces forgetting of $D_0 (= 8M)$ passages while incremental indexing in a large corpus setting, MS MARCO (full) containing $8.9M$ passages.

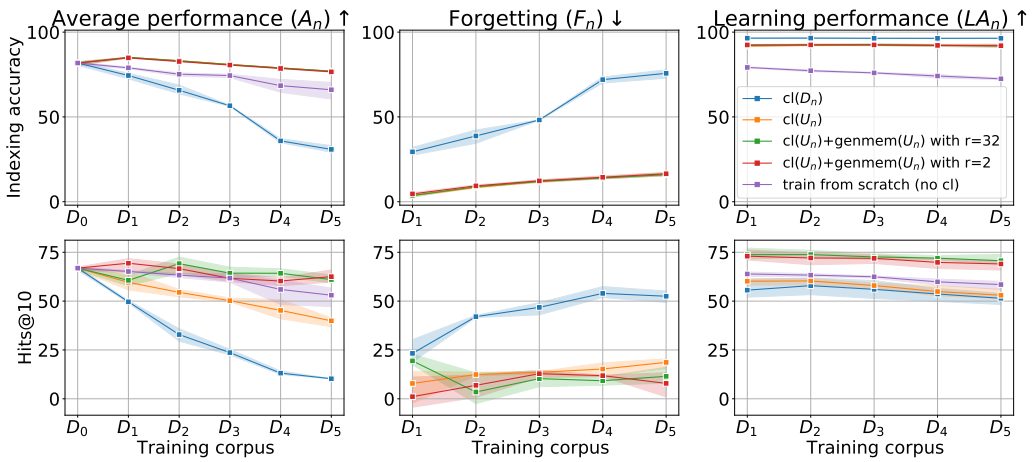


Figure 7.3: Investigating the effectiveness of generative memory in mitigating forgetting when continuously indexing new corpus D_n (T5-Base model and atomic docids representation) for the NQ dataset. ↑ indicates higher is better, ↓ indicates lower is better. We observe that continual indexing of old and new documents cl(U_n) helps to alleviate forgetting of older documents when evaluated on retrieval tasks. However, average Hits@10 (A_n) still undergo 23 points drop after sequential updates ($D_0 \rightarrow D_1 \cdots \rightarrow D_5$). Generative memory enables sparse replaying of pseudo-queries for old documents and continual semi-supervised learning with new documents. We observe that augmenting generative memory during continual indexing not only reduces the forgetting (F_n) but also improves average Hits@10 (A_n) by $+21.1\%$ over considered baselines (see Figure C.2 for Hits@1 results and Figure C.3 for MS MARCO results in the Appendix C.2).

when we replay (pseudo-)queries corresponding to D_0 corpus. Moreover, generative memory outperforms episodic memory, without retaining original queries. Although from Table 7.1, we see generative memory, cl(U_1)+genmem(U_1), underperforms cl(U_1), from Figures 7.3 and C.2, we see that generative memory, cl(U_5)+genmem(U_5), outperforms cl(U_5) both in terms of average performance A_n and forgetting F_n over five sequential updates. **These results convincingly show that the ER with generative memory significantly alleviates forgetting the retrieval task compared to considered baselines.**

7.4.2 Does generative memory enable forward transfer to new documents?

One of the goals of DSI++ is to enable answering queries related to newly indexed documents. Towards this goal, in Table 7.1, for the NQ dataset, we look at the retrieval task performance (Hits@1) for D_1 after incrementally indexing D_1. To compare different methods, we consider a baseline in the form of ER with ground-truth queries for D_1 (cl(U_1)+epsmem(D_1) - 48.3). We see that without any fine-tuning on the retrieval task for D_1, incremental learning with indexing objective shows impressive forward transfer (or zero-shot gains, cl(D_1) - 31.7 and cl(U_1) - 34.0). Moreover, ER with generative memory outperforms supervised baseline (cl(U_1)+genmem(D_1) - 57.7). However, we notice that replaying queries corresponding to either D_0 or D_1 hurt forward

transfer to D_1 (cl(U_1)+genmem(D_0) - 8.6) or amplify forgetting of D_0 (cl(U_1)+genmem(D_1) - 7.0), respectively. These results suggest that the memory module should include (pseudo-)queries corresponding to old and new documents. From Figure 7.3, we see that continual indexing method cl(U_n) has a downward trend for LA_n (Hits@10), therefore, eventually forgetting the retrieval task. On the other hand, ER with generative memory is relatively constant, providing evidence against forgetting. **In summary, we show that ER with the generative memory improves the overall performance for the retrieval task, reducing forgetting of previously indexed documents and enabling forward transfer to newly indexed documents.**

7.4.3 Does generative memory generalize to different datasets?

In Table 7.1, for the MS MARCO dataset, we report Hits@1 to be 78.2 after training on D_0 passages. We see that continually indexing both D_0 and D_1 corpora (cl(U_1) - 76.5 and cl(U_1)+genmem(U_1) - 73.7), significantly reduce forgetting the retrieval task (Hits@1) over just indexing the new corpora D_1 (cl(D_1) - 68.0). Next, we look at the retrieval task performance (Hits@1) for D_1 after incrementally indexing D_1. We see that without any fine-tuning on the retrieval task for D_1, incremental learning with indexing objective shows impressive forward transfer (cl(D_1) - 36.1 and cl(U_1) - 35.3). Moreover, ER with generative memory, cl(U_1)+genmem(U_1) - 80.6, is far superior to just the incremental indexing objective. Similar to the results with the NQ dataset, we show that ER with generative memory, cl(U_n)+genmem(U_n), improves the overall performance for the retrieval task, reducing forgetting of previously indexed documents and enables forward transfer to new documents compared to continual indexing of all documents, cl(U_n). **We show that our results hold across two datasets – Natural Questions and MS MARCO, thus, showcasing the generalizability of our approach.**

7.4.4 Investigating the effectiveness of the generative memory with the scale of a corpus.

We conduct experiments with a full MS MARCO passage retrieval dataset containing approx. $8.9M$ passages. We construct two corpora – $D_0 = 8M$ and $D_1 = 841,823$ passages. We train the DSI model (T5-Base with unstructured atomic docid representations) using D_0 passages and incremental add D_1 passages. In Table 7.1, we report our results for MS MARCO (full). We see that continual fine-tuning with the indexing task on D_1, cl(D_1), completely forgets the retrieval task for D_0 passages (Hits@1 goes to 0.1 from 16.3). However, the generative memory-based approach significantly reduces forgetting (Hits@1 of 7.3). Moreover, generative memory enables continual semi-supervised learning by augmenting pseudo-queries for D_1 passages and thereby improving forward transfer (Hits@1 of 31.6 as compared to 18.2 for cl(D_1)). **We convincingly demonstrate that our proposed solution also reduces forgetting in a large corpus setting.**

7.4.5 Investigating sparsity of experience replay (ER) on forgetting.

ER with generative memory co-trains the indexing and pseudo-labeled retrieval tasks. Tay et al. (2022) introduces a mixing ratio *r* to define the ratio of indexing to retrieval samples. The mixing

ratio is inversely related to the sparsity of ER, i.e., higher r (more indexing samples) corresponds to sparse updates from pseudo-labeled retrieval samples. Following (Tay et al., 2022), we consider $r = \{2, 32\}$ for our analysis. From Figure 7.3, we see that $r = 32$ (sparse replay) slightly outperforms $r = 2$ in terms of average performance, forgetting, and learning accuracy. **These results suggest that even sparse regularization updates from ER positively influence the backward and forward transfer in DSI++.**

7.4.6 Analyzing index construction time for DSI++.

Index construction using DSI involves training a Transformer model. DSI++ setup enables incremental updating of the indexer. In Figures 7.3, C.2, and C.3, we show that our proposed incremental indexer updating method outperforms the "train from scratch" baseline in terms of the average performance (A_n). Furthermore, in the case of the NQ dataset, we require 350K training steps to index D_0 corpus of 50K documents. If we index an additional D_1 to D_5 corpora (10K each) by re-training the DSI model every time, the total number of steps would be around 1.75M. On the other hand, our proposed approach requires just above 300K additional updates to incrementally index all five corpora, almost 6 times fewer updates. **We show that our approach outperforms re-training the model from scratch in terms of overall performance and is computationally efficient.**

7.5 Related Work

We review relevant prior works along two dimensions – application setups related to DSI++ and continual learning methods to alleviate forgetting and enable forward transfer.

Language models (LMs) as knowledge bases (KBs). Petroni et al. (2019) shows that pre-trained BERT (Devlin et al., 2019) models capture relational knowledge comparable to that of the KBs constructed using off-the-shelf techniques. Concretely, these models can be used to extract factual knowledge about relations between entities by providing a prompt to predict missing words in a cloze-style template (e.g., "New Delhi is the capital of _"). Similarly, Roberts et al. (2020) demonstrates that pre-trained T5 (Raffel et al., 2020) models can be employed to answer open-domain questions without access to any external knowledge or context. However, unlike structured KBs, it is non-trivial to update knowledge stored implicitly in the weights of these models. Therefore, Zhu et al. (2020) introduces an experimentation setup where the task is to update facts stored within the pre-trained models and proposes a constrained optimization method, similar to Elastic Weight Consolidation (Kirkpatrick et al., 2017), to alleviate catastrophic forgetting. With similar motivation, (Dhingra et al., 2022) introduces a diagnostic dataset to probe LMs for facts that change over time and suggests jointly modeling text with its timestamp for improved memorization of seen facts. Recent works have been investigating efficient ways to localize and edit facts stored with the LMs (AlKhamissi et al., 2022) using finetuning (Zhu et al., 2020; Dhingra et al., 2022), hyper-networks (De Cao et al., 2021; Mitchell et al., 2022), and direct editing (Meng et al., 2022). Although a crucial line of work around updating facts in the pre-trained LMs, using prompting as our probing mechanism only provides a lower bound

estimate of the knowledge contained in these models (Jiang et al., 2020). On the other hand, we explicitly focus on the memorization task in DSI++. This task helps us to answer questions related to catastrophic forgetting more convincingly rather than bounded by the mechanism of how we probe these models.

Optimization-based approaches for continual learning encode the necessary inductive biases required to enable continual learning by modifying the training dynamics. Flatter minima are shown to alleviate forgetting (Mirzadeh et al., 2020b). Further, Mehta et al. (2023b) showed that explicitly optimizing for flatter loss basins using Sharpness-Aware Minimization (SAM; Foret et al. (2021)) reduces forgetting. Building on these works, we show that flatter minima induced by SAM reduce implicit forgetting during memorization, thereby leading to more stable memorization (see Section 8.3).

Memory-based (aka data-based regularization) approaches for continual learning constrain the parameter updates based on the previous task examples sampled from memory. Sparse experience replay using episodic memory (Chaudhry et al., 2019) is a prominent approach and in Section 7.3, we discuss its limitations of it for DSI++. Next, Shin et al. (2017); Sun et al. (2020) learns a parametric model to reconstruct the examples for seen tasks. However, in DSI++, we do not see queries for the new documents. Therefore, we use a parametric memory to generate pseudo-queries for already indexed (older) documents and an incoming batch of new documents, thus, enabling us to leverage unlabeled data (in the form of new documents) for continual semi-supervised learning. On the other hand, Sun et al. (2020) assumes that the incoming data are fully labeled, which is not applicable in DSI++ (we do not get to see queries for the new documents). Furthermore, Sun et al. (2020) shows that using a parametric model underperforms episodic memory. In our work, we do not generate example pairs (x, y) but rather generate pseudo-queries (y), similar to concurrent works (Zhuang et al., 2022; Bonifacio et al., 2022). We show that our approach outperforms episodic memory. Lastly, in the context of pseudo-query generation, neural models are prone to hallucinate additional content not supported by the input documents. Future works can study methods to filter out noisy pseudo-queries (Mehta et al., 2022) during incremental indexing.

Test time adaptation approaches for continual learning use episodic memory at the inference time to alter the model weights before making predictions (Rebuffi et al., 2017; Sprechmann et al., 2018; de Masson D'Autume et al., 2019; Mehta et al., 2020). Updating the DSI indexer for every user query is computationally expensive, so we focus on continual learning methods during training. Apart from continual learning-focused approaches, retrieval augmented generation (Guu et al., 2020; Izacard and Grave, 2021; Borgeaud et al., 2022) family of approaches retrieve auxiliary passages/documents to enhance pre-trained language models. These approaches alter test-time predictions of the generative models by augmenting their input with relevant passages retrieved from external retrievable memory. Moreover, one explicitly disables the updates to the employed pre-trained (and retrieval) model using the external retrievable memory. Such approaches do not faithfully assess the fundamental challenge of learning continually, specifically catastrophic forgetting. On the other hand, our work focuses on the recently introduced DSI

paradigm (Tay et al., 2022), where information in the document corpus is encoded into the model parameters. Therefore, any updates to the underlying corpus necessitate updating the model parameters hence, undergoing severe forgetting. Our work tackles a more challenging setup for studying the forgetting phenomenon in detail. However, retrieval-augmented generation-based methods do not analyze the forgetting phenomenon at all, only looking at overall performance metrics. We agree that continual learning is broader than catastrophic forgetting; however, in this work, we decide to study the forgetting phenomenon in detail on one of the challenging setups, if not the most challenging.

Parameter isolation-based approaches for continual learning assign different dedicated subsets of the model parameters to each task to prevent forgetting (De Lange et al., 2021). While learning a new task, these methods either freeze a subset of the parameters corresponding to older tasks or dynamically add new parameters per new task. At the prediction time, these methods typically require a task identifier to activate the corresponding subset of parameters for inference. In the DSI paradigm, we are given user queries at the inference time, and the goal is to predict relevant document identifiers. Now during incremental indexing, if we consider every new document corpus as a new task, then a typical parameter isolation-based approach would require a corpus identifier for every user query at the test time, defeating the whole purpose of the DSI paradigm. Due to this, the parameter isolation-based approaches in their current form are rendered less useful for DSI++. Nevertheless, we believe that by masking the weights for the already indexed corpus, one is explicitly disabling the updates to the underlying DSI model; therefore, parameter isolation-based methods would be robust to forgetting and future works should explore them for DSI++.

7.6 Discussion

DSI++ introduces a new approach to address a crucial requirement of DSI models for practical use in production setups, where continuous addition of new documents to the corpus is necessary. Through experiments, we demonstrate the effectiveness of our generative memory-based approach to reduce catastrophic forgetting. This work establishes a foundation for further research, benefiting both DSI models and the broader community of lifelong (semi-supervised) learning.

In this chapter, we leverage a generative language model to sample pseudo-queries and enable lifelong semi-supervised learning for updating Transformer memory. However, generative language models are prone to hallucinate additional content not supported by the inputs (Maynez et al., 2020). Therefore, one should select quality pseudo-labels to alleviate the risk of reinforcing the model's mistake. In our recent work (Mehta et al., 2022), we focus on generating quality pseudo-labels using limited labeled data to enable semi-supervised learning. Extending this work to lifelong semi-supervised learning where the data stream continuously undergoes distribution shift is an open problem. We leave such analysis to future work.

Chapter 8

Unlabeled Data: Role of Lifelong Learning in Pre-training

8.1 Overview

In previous chapters, we studied several lifelong learning scenarios – task-incremental learning (Chapters 3, 5), domain-incremental learning (Chapters 4, 6), and class-incremental learning (Chapters 6, 7). In each scenario, the lifelong learning system encounters (labeled or unlabeled) data stream, and the goal is to learn new tasks while alleviating forgetting of previous tasks. However, lifelong learning is not restricted to multi-task scenarios, as deep neural networks undergo forgetting even when there is no clear data distribution shift; i.i.d setup for single-task learning (Toneva et al., 2019). Studying single-task learning through the lens of lifelong learning is open to investigation. In this chapter, we investigate the role of lifelong learning in pre-training where data does not undergo a clear distributional shift, and provide justification for our exploration.

Lifelong and transfer learning aim to utilize prior knowledge to improve performance on new tasks. Recently, large language models like BERT (Devlin et al., 2019), RoBERTa(Liu et al., 2019b), GPT-2 (Radford et al., 2019), T5 (Raffel et al., 2020), GPT-3 (Brown et al., 2020), Gopher (Rae et al., 2021), OPT (Zhang et al., 2022), GLAM (Du et al., 2022), PaLM (Chowdhery et al., 2023), LLaMA 2 (Touvron et al., 2023) have become popular for transfer learning in NLP. These models are pre-trained on self-supervised tasks using large text corpora before being fine-tuned on downstream tasks. Pre-training helps these models learn general language features, reducing the labeled data requirements for new tasks (Devlin et al., 2019). However, pre-training is still "data hungry"; for instance, GPT-3 is trained using 499B tokens (Brown et al., 2020), LLaMa 2 is trained using 2T tokens (Touvron et al., 2023), making it expensive to retrain new language models continuously. Further, investigating the learning dynamics of these models reveals that they acquire different types of knowledge differently over the course of pre-training (Liu et al., 2021). Specifically, RoBERTa quickly and stably acquires linguistic knowledge, slowly acquires facts and commonsense, and does not consistently acquire reasoning abilities. If we view pre-training as continuous learning from a stream of "potentially redundant" data with a limited replay of previously seen examples, the above results hint at forgetting different skills (or knowledge)

over the course of pre-training. Recent findings that deep neural models are prone to forgetting even when data streams are stationary (i.i.d) (Toneva et al., 2019) motivate us to investigate the role of lifelong learning in pre-training, especially in alleviating forgetting and improving sample complexity (or *efficient* pre-training).

To enable *efficient* lifelong learning systems, the earlier chapters in this thesis suggest modifying different components of data-driven machine learning: model initialization (Chapters 3, 4), training dynamics (Chapters 5, 6), and data (Chapter 7). Building on these works, in this chapter, we investigate each component across different pre-training scenarios and answer the following research questions:

1. What should be the initialization scheme for pre-training? For instance, can we learn the Transformer architecture's initialization to benefit stable learning of self-supervised tasks?
2. What should be the training dynamics for pre-training? For example, can we explicitly optimize for the flatness of the loss landscape to alleviate forgetting throughout pre-training?
3. What data should be used for pre-training? Low-quality or noisy samples may hurt learning by amplifying negative interference (or forgetting); therefore, can we filter them out before pre-training for a sample-efficient pre-training?

8.2 What should be the initialization scheme for pre-training?

In Chapter 5 (Section 8.2), we show that initializing models with MetaInit (Dauphin and Schoenholz, 2019) helps reduce forgetting during lifelong learning. As we examine pre-training from the perspective of lifelong learning, we pose a question—**Can the advantageous initialization provided by MetaInit, known to facilitate gradient descent by commencing in locally linear (or wider) regions with minimal second-order effects, also play a role in alleviating forgetting during pre-training?**

Experimental design. To study the effectiveness of MetaInit during pre-training, we consider the task of language model pre-training. Concretely, we focus on the BERT-style Transformer encoder and train it with the masked language modeling (MLM) objective (Devlin et al., 2019). We closely follow Geiping and Goldstein (2023) to pre-train large language models under resource-constrained environments, i.e., training a language model on a single GPU in one day. Geiping and Goldstein (2023) proposes three-way modifications to the original BERT pre-training recipe — (i) architectural modifications like disabling of biases in Query, Key, and Value, as well as the Feed Forward layer, resulting in enhanced computational efficiency; (ii) modified training to include 25% masking rate, eliminating dropout, and implementing batch size schedules; and (iii) use of The Pile (Gao et al., 2020) dataset instead of the English subset of the Wikipedia along with the Toronto Book Corpus (Zhu et al., 2015), as it yields superior downstream MNLI performance. This entire recipe is denoted as *Crammed BERT* (with 120M total parameters), and for a more in-depth understanding, we direct readers to Geiping and Goldstein (2023).

To explore the forgetting dynamics during pre-training, we adopt a controlled setting for training the language model. Specifically, we utilize the publicly available processed sample of the Pile dataset (Geiping and Goldstein, 2023), featuring a sequence length of 128. Our training split

comprises approximately 10 million tokens (78,336 sequences), with an additional 1.05 million tokens (8,192 sequences) forming our validation split. The batch size is set to 512 sequences, and the language model undergoes pre-training for 200 epochs (or 30,600 update steps), equivalent to encountering a total of 2 billion tokens and takes around 6 hours on a single Nvidia A6000 GPU with 48GB memory. Consistent with Geiping and Goldstein (2023), we employ the AdamW optimizer (Kingma and Ba, 2014) with a triangular learning rate schedule. The learning rate starts at 2.5e^{-4} and linearly increases to 1.0e^{-3} over 75% of the update steps, followed by a decay to zero over the remaining 25% steps. For the initialization using MetaInit, we opt for the SGD optimizer with a learning rate of 0.01 and a momentum of 0.9, employing a batch size of 32 and training for 750 updates.

8.2.1 Forgetting Curves in Language Models

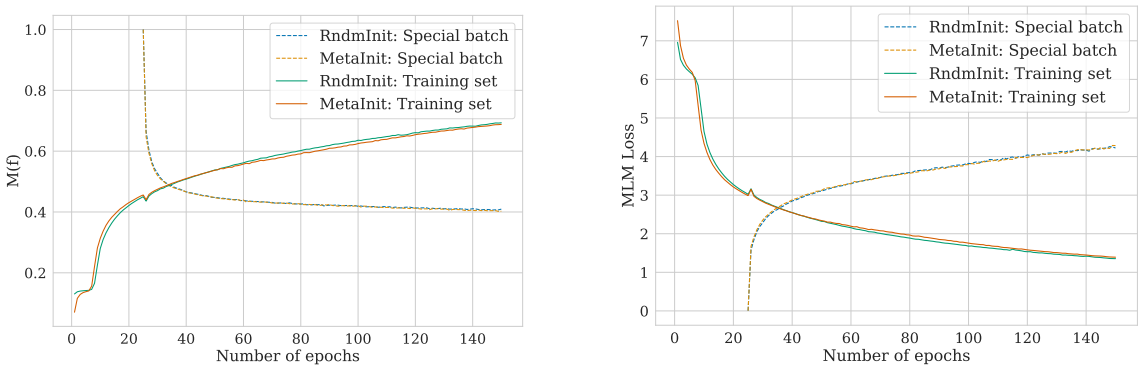
To study the learning dynamics of language models, Liu et al. (2021) probes language models for different types of knowledge throughout pre-training. They show RoBERTa quickly and stably acquires linguistic knowledge, slowly acquires facts and commonsense, and does not consistently acquire reasoning abilities; hinting at forgetting different skills (or knowledge) over the course of pre-training. Although a crucial work on understanding the learning dynamics, using prompting as our probing mechanism only provides a lower bound estimate of the knowledge contained in these models (Jiang et al., 2020). Therefore, similar in spirit to our work in Chapter 7, we explicitly focus on the memorization of the data. This helps us to answer questions related to forgetting more convincingly rather than bounded by the mechanism of how we probe these models. Concretely, we compute the memorization metric, introduced by Tirumala et al. (2022), for self-supervised settings and discuss it below.

Let V represent the vocabulary size. Consider a set of contexts C, defined as a collection of tuples (s, y) where s represents an input context (an incomplete block of text), and y denotes the index of the ground truth token in the vocabulary that completes the block of text. Let S denote the set of input contexts, and $f_w : S \to \mathbb{R}^V$ represent a language model. A context $c = (s, y) \in C$ is considered *memorized* if the index that maximizes $f(s)$ is equal to y. Formally, the memorization, $M(f)$, is defined as

$$M(f) = \frac{\sum_{(s,y)\in C} \mathbb{I}\{\arg\max(f(s)) = y\}}{|C|}. \tag{8.1}$$

It is worth noting that the above-defined metric can be interpreted as accuracy, as it measures the frequency with which the arg max of the language model aligns with the ground-truth token.

Motivated by the hypothesis of the *forgetting curve* (Loftus, 1985), which posits that human memory declines over time without attempts to retain it, Tirumala et al. (2022) investigates forgetting curves in language models. Their study reveals that larger models exhibit slower degradation in memorization compared to smaller models. In our work, we leverage the same methodology to explore whether MetaInit influences the forgetting curve. Following the approach of Tirumala et al. (2022), we construct forgetting curves in language models. Initially, we select a batch of data distinct from the training set, specifically from a validation set (termed a "special batch"). Subsequently, we take a checkpoint during the ongoing model training and memorize



(a) Forgetting curves

(b) Masked language modeling loss

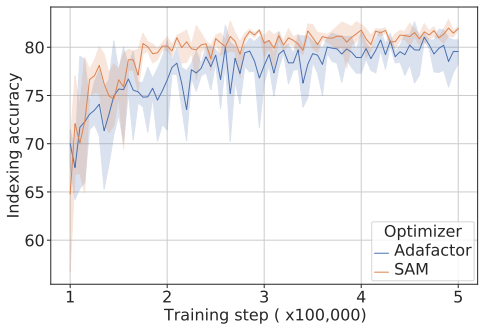
Figure 8.1: Exploring the impact of model initialization (RandomInit vs. MetaInit) on forgetting curves during pre-training in the crammed BERT setting. We inject the special batch into the training set at the 25th epoch and evaluate the proportion of special batch memorized as we continue training. (a) Our observations indicate that memorization, denoted as $M(f)$, for the special batch deteriorates rapidly, converging to a baseline value of 0.4 for both initializations. This suggests that more investigation is needed to determine the role of MetaInit initialization on forgetting dynamics during pre-training. (b) For both initializations, the MLM loss for the special batch continues to rise. In contrast, the memorization of the special batch levels off, suggesting that pre-trained models do not completely forget, a trend not fully captured by the MLM loss.

this special batch by allowing the model to undergo training on this specific data. Afterward, we resume standard training on the original training set. Our analysis focuses on the degradation in memorization (Equation 8.1) on the special batch, examining whether the model initialization factor (RndmInit vs. MetaInit) influences the forgetting curve. Throughout our experiments, we consistently employ the entire validation set as the special batch. Additionally, we note that the special batch is encountered only once upon its initial introduction.

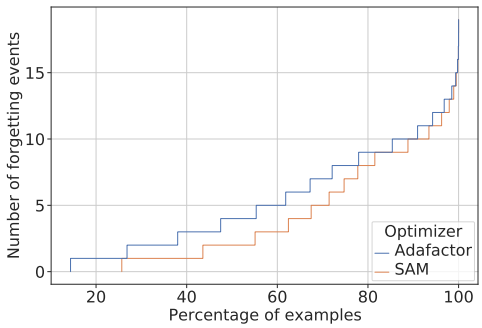
Discussion. In Figure 8.1a, we examine the impact of model initialization (RandomInit vs. MetaInit) on forgetting curves during pre-training. By introducing the special batch into the training set at the 25th epoch, we track the proportion of the special batch memorized throughout training. Our observations reveal a rapid degradation in memorization for the special batch, converging to a baseline value of 0.4 for both initializations. This indicates that **further investigation is needed to determine the role of MetaInit initialization on forgetting dynamics during pre-training**. Conversely, in Figure 8.1b, we observe a continuous increase in the MLM loss for the special batch. However, the memorization of the special batch levels off, indicating that pre-trained models do not completely forget, a phenomenon not fully captured by the MLM loss.

8.3 What should be the training dynamics for pre-training?

In Chapter 7, we discuss memorization (or indexing) as a pre-training task in the DSI paradigm (Tay et al., 2022) where the goal is to learn a neural corpus indexer that takes document content as



(a) Indexing accuracy during memorization



(b) Cumulative histogram of forgetting events

Figure 8.2: Investigating the effectiveness of SAM for alleviating implicit forgetting in the T5-Base model. (a) We observe serious fluctuations in the indexing accuracy in the case of the Adafactor optimizer, thereby suggesting unstable memorization. SAM leads to relatively stable memorization of documents. (b) A forgetting event (Toneva et al., 2019) is defined when an individual document goes from being classified correctly to incorrectly over the course of memorization. SAM increases the percentage of examples experiencing zero forgetting events by an absolute 12% over Adafactor.

input and maps it to a document identifier (docid). The retrieval task constitutes a downstream task and necessitates mapping user queries to relevant docids. Under the unstructured atomic docid representation strategy, each docid is assigned a unique token/class label. Given the large number of documents in the corpus (even more than a million), memorization constitutes an instance of challenging extreme classification setting (Bengio et al., 2019). Furthermore, for every class, we have only one labeled example (i.e., document and its identifier), making this task setup rare. Motivated by this largely unexplored setup, we investigate the learning dynamics for the memorization task over the course of pre-training. We consider the experimental setup as discussed in Section 7.2.1.

8.3.1 Implicit forgetting during memorization

In Figure 8.2a, we visualize the indexing accuracy for the T5-Base model, optimized with Adafactor (Shazeer and Stern, 2018). We note that the model performance fluctuates throughout training, suggesting unstable memorization. We hypothesize that the model continuously undergoes the forgetting phenomenon wherein subsequent mini-batch updates interfere with the previously memorized documents. To differentiate this phenomenon from forgetting due to adding new documents (or tasks), we refer to the earlier one as *implicit forgetting* during memorization in this chapter. To quantify instability during memorization, we compute forgetting event (Toneva et al., 2019) statistics. A *forgetting event* is defined as being when an individual document goes from being classified correctly (mapped to correct docid) to incorrectly over the course of memorization. In Figure 8.2b, we plot the cumulative histogram of forgetting events and see that almost 88% of the documents undergo forgetting at least once, thus, validating our hypothesis about implicit forgetting.

8.3.2 SAM alleviates implicit forgetting

In Chapter 5 (Mehta et al., 2023b), we show that pre-trained initialization implicitly alleviates forgetting as they prefer flatter minima and explicitly optimizing for the flatness using Sharpness-Aware Minimization (SAM; Foret et al. (2021)) further lessens forgetting. Based on these observations, we hypothesize that modifying the training dynamics of the memorization tasks using SAM should alleviate implicit forgetting.

We investigate the applicability of SAM for alleviating the implicit forgetting phenomenon. Concretely, we continually pre-trained the T5-Base model to memorize D_0 corpus containing 50K unique documents. We compare the performance of the SAM optimizer with the vanilla Adafactor optimizer. In Figure 8.2a, we see that SAM outperforms Adafactor in terms of the overall indexing accuracy. Further, we note that SAM undergoes less severe fluctuations during training, thus, hinting at lesser forgetting. To bolster this claim, in Figure 8.2b, we see that SAM has a significantly more percentage of documents corresponding to a lower cumulative number of forgetting events. For example, SAM stably (with zero forgetting events) memorizes $+12\%$ more documents than Adafactor. We also note that SAM (35.9 ± 2.2) outperforms Adafactor (32.5 ± 6.4) when evaluated on the downstream retrieval task (Hits@1) corresponding to D_0.

Discussion. In Chapter 5, we show that explicitly optimizing for flatter loss basins using SAM leads to less forgetting, especially in task-incremental learning settings where data undergoes a clear distributional shift. We extend this work to the recent DSI paradigm (Tay et al., 2022) and convincingly demonstrate that SAM helps with the stable memorization of documents. **Our results generalize even to the pre-training settings where data does not undergo a clear distributional shift (i.e., memorization task)**. Although SAM helps stably memorize documents, there is still room for improvement, and our work invites more future work in this direction. Also, future research can explore the effectiveness of SAM on forgetting dynamics during language model pre-training, similar to the analysis presented in Section 8.2.1.

8.4 What data should be used for pre-training?

Recently there has been a growing body of research in data-centric AI that empirically demonstrates the significant impact of training data quality on the final performance of large models (Abbas et al., 2023; Brown et al., 2020; Gadre et al., 2023; Radford et al., 2023; Schuhmann et al., 2022; Xie et al., 2023). Simultaneously, data pruning (Toneva et al., 2019; Paul et al., 2021; Sorscher et al., 2022; Wang et al., 2023) has gained attention in the community for enhancing training efficiency and reducing (semantic) redundancy in training data. For instance, Toneva et al. (2019) investigates the forgetting dynamics of neural networks during training on individual classification tasks and demonstrates that a significant portion of "unforgettable" examples can be excluded from the training dataset while preserving state-of-the-art generalization performance. However, our work in Chapter 6 challenges the idea of retaining "forgettable" examples in the episodic memory buffer, highlighting that such examples may be less representative of the true data distribution and could lead to negative interference (see Section 6.5.3). In conclusion, the ideal data pruning metric differs among diverse datasets, and the approaches suggested in the above

studies often depend on manually crafted heuristics, typically leading to suboptimal performance (Sorscher et al., 2022).

Meta learning aims to learn the *inductive biases* (*e.g.* training data) of a machine learning program in such a way that a model trained with these inductive biases achieves optimal performance on user-specified *objectives* (*e.g.* quick generalization). This concept of meta learning can naturally be formulated as bilevel optimization, where the upper (meta) level problem encodes inductive biases and objectives, and the lower (base) level optimization problem represents the main machine learning program of interest, such as image classification or language modeling. Depending on the design of inductive biases and objectives, meta learning has found many applications in machine learning, including hyperparameter optimization (Franceschi et al., 2018), data optimization (Gudovskiy et al., 2021; Shu et al., 2019), neural architecture search (Liu et al., 2019a; Zhang et al., 2021), learned optimizers (Metz et al., 2022, 2019), few-shot learning (Finn et al., 2017; Rajeswaran et al., 2019), and lifelong learning (Mehta et al., 2020).

Hence, in this chapter, we focus on the application of (gradient-based) meta learning, specifically in the domain of *neural data optimization*. Concretely, we modify pre-training data, by reweighting or filtering, based on the specific objectives outlined in the meta optimization problem. Recognizing that the training data for large models inherently acts as highly dimensional inductive biases in machine learning, we hypothesize that the application of meta learning in data optimization will enable us to efficiently optimize these intricate high-dimensional inductive biases. This shift in focus aims to address the variability in optimal data pruning metrics across datasets and move beyond relying on manually designed heuristics for improved performance.

From a technical perspective, employing gradient-based meta learning for large-scale data optimization comes with a significant computational and memory overhead, particularly when dealing with models trained with adaptive optimizers (e.g., Transformers trained with Adam). Consequently, we use the ScalAble Meta learning Algorithm (SAMA; Choe et al., 2023) in our study. We review SAMA in Section 8.4.1, investigate data pruning for pre-training of vision models in Section 8.4.2, and continued pre-training of large language models in Section 8.4.3.

8.4.1 Neural Data Optimization using ScalAble Meta learning Algorithm (SAMA)

We begin by reviewing the fundamentals of gradient-based meta learning (GBML) and subsequently explore how SAMA tackles the crucial factors that hinder the scalability of GBML. Mathematically, meta learning is commonly formulated as bilevel optimization as follows

$$\begin{aligned} \lambda^* &= \underset{\lambda}{\operatorname{argmin}}\ L_{meta}(D_{meta}; \theta^*(\lambda)) \\ s.t.\ \ \theta^*(\lambda) &= \underset{\theta}{\operatorname{argmin}}\ L_{base}(D_{base}; \theta, \lambda) \end{aligned}$$

where λ (respectively, θ) are the parameters of meta (base) learners, D_{meta} (D_{base}) are meta (base) datasets, and L_{meta} (L_{base}) are meta (base) loss functions. An important implication of the above formulation is that meta learning changes the task of finding the optimal inductive biases from designing heuristics to designing meta optimization problems. To illustrate, let us consider the data pruning task, where the objective is to ascertain the importance weight of

each training datum and subsequently prune samples based on a specified cutoff threshold. One conventional "heuristics-based" approach involves utilizing a forgetting metric as an importance weight and filtering training samples with lower forgetting values (Toneva et al., 2019). In contrast, Meta-Weight-Net (MWN; Shu et al., 2019) automatically meta-learns the importance weight λ by designing a meta optimization problem (basically by constructing a meta dataset D_{meta}). In short, unlike heuristic-based methods that explicitly specify "how to learn," meta learning methods only specify "what to learn" and let the meta learner automatically determine "how."

GBML computes a meta gradient composed of two terms—the best-response Jacobian and direct gradient—with the chain rule, as follows

$$\frac{\partial L_{meta}}{\partial \lambda} = \underbrace{\frac{\partial \theta^*}{\partial \lambda}}_{\text{best-response Jacobian}} \cdot \underbrace{\frac{\partial L_{meta}}{\partial \theta^*}}_{\text{direct gradient}} \tag{8.2}$$

Since the direct gradient computation is straightforward with the underlying automatic differentiation library, the major challenge in GBML lies in computing the best-response Jacobian. SAMA employs an implicit differentiation (Rajeswaran et al., 2019) solution to GBML. Implicit differentiation essentially computes the best-response Jacobian using Cauchy's Implicit Function Theorem (IFT) and reframes the base optimization problem through the lens of fixed-point iteration with an iterative solver u, as follows

$$\frac{\partial \theta^*}{\partial \lambda} = - \underbrace{\frac{\partial u}{\partial \lambda}}_{\text{meta Jacobian}} \cdot \left(\underbrace{\frac{\partial u}{\partial \theta^*}}_{\text{base Jacobian}} \right)^{-1} \quad \text{where} \begin{cases} \theta^* = \lim_{t \to \infty} \theta_t \\ \theta_t = \theta_{t-1} - u(\theta_{t-1}; \lambda) \end{cases} \tag{8.3}$$

While exact implicit differentiation necessitates solving the base optimization problem to convergence (θ^*) by repeatedly applying an iterative solver u (e.g., SGD or Adam) to compute base and meta Jacobians, this is often computationally infeasible, particularly in large-scale learning scenarios. Consequently, in practical applications, we approximate θ^* by employing a small number of unrolled update steps of u. This approximation involves alternating gradient descent between base and meta optimization problems, where the base gradient is computed using standard backpropagation, and the approximate meta gradient is determined using Equations 8.2 and 8.3.

The computation of the meta gradient in GBML encounters notable challenges, including high computational and memory costs, algorithmic instability, and lack of support for efficient distributed training. To overcome these limitations and enhance scalability, we employ SAMA. SAMA addresses these challenges through three key strategies: (i) approximating the inversion of the base Jacobian with an identity matrix, (ii) algorithmic adaptation for adaptive optimizers to enhance stability, and (iii) implementing a distributed approach for the computation of the meta gradient. For more details, we defer readers to Choe et al. (2023).

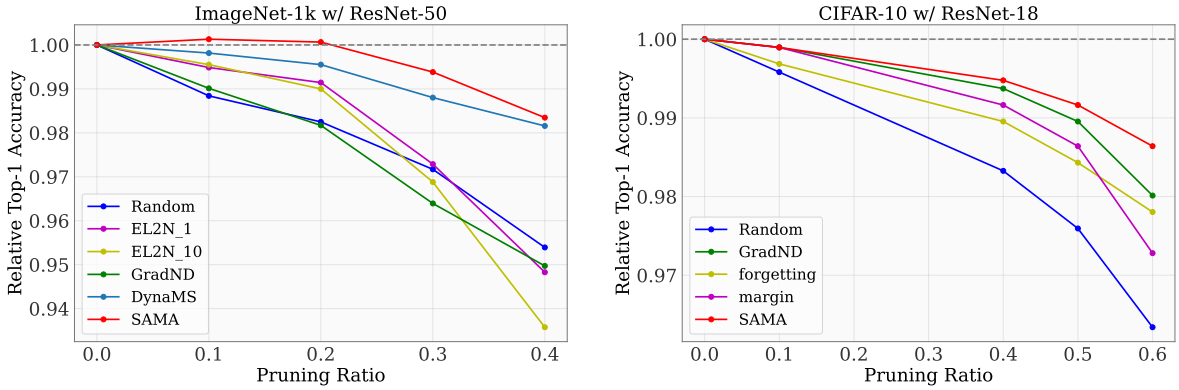
8.4.2 Efficient Data Pruning for Pre-training of Large Vision Models

In this section, we investigate the applicability of SAMA for data pruning during pre-training. Our approach involves deviating from manually crafted data pruning metrics and, instead, adopting an automated meta-learning process for determining the importance weight of each training

example, inspired by Meta-Weight-Net (MWN; Shu et al., 2019). We introduce four significant modifications to MWN. First, we replace their iterative differentiation meta gradient algorithm with SAMA to achieve improved memory and compute efficiencies. Second, we speed up meta learning by enabling distributed training using SAMA's efficient communication strategy. Third, we use the uncertainty of the prediction in addition to the loss value as an input to MWN to better estimate the importance weight for each training example. Last, we use training data *both* in the base and the meta levels, assuming no additional validation data. A bilevel optimization formulation of our method is as follows

$$\lambda^* = \underset{\lambda}{\text{argmin}}\ L(\mathcal{D}_{train}; \theta^*(\lambda))$$
$$\text{s.t. } \theta^*(\lambda) = \underset{\theta}{\text{argmin}}\ \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} w(L,\, U; \lambda) \cdot L(x, y; \theta)$$

where $w(\cdot; \lambda)$ is MWN that takes the loss value L and the uncertainty U of the training sample (x, y) as an input and outputs the importance weight. Under this setup, we run meta learning with SAMA for 30 / 50 epochs respectively for ImageNet-1k / CIFAR-10 and obtain the pruning metrics by averaging the importance weights of the last 5 epochs. We compare our method to several popular static—EL2N$_1$/EL2N$_{10}$/GradND (Paul et al., 2021), forgetting (Toneva et al., 2019) and dynamic—DynaMS (Wang et al., 2023) data pruning baselines, and present the results in Figure 8.3.



ImageNet-1k w/ ResNet-50	EL2N$_1$	EL2N$_{10}$	GradND	DynaMS	SAMA
Relative Search Time	0.16	1.65	1.65	0.16	0.46

Figure 8.3: **Top Left:** ImageNet-1k data pruning results with ResNet-50. Reported numbers are relative accuracy compared to full training accuracy (*i.e.*, pruned_acc/full_acc). Accuracy for other baseline methods is obtained from DynaMS (Wang et al., 2023). **Top Right:** CIFAR-10 data pruning results with ResNet-18. Accuracy for other baseline methods is obtained from Deepcore (Guo et al., 2022). The pruning ratio is defined as the fraction of total examples pruned using the pruning strategy. BML-based data pruning with SAMA outperforms heuristics-based data pruning across different dataset scales. **Bottom:** Relative time spent in finding data to prune compared to full ImageNet-1k training time.

We show that **GBML-based data pruning with SAMA not only outperforms heuristics-based data pruning but also works well across different dataset scales**. Surprisingly, we observe that GBML-based data pruning even leads to improvements in test accuracy at the pruning ratio of 0.1 and 0.2 on ImageNet-1k. The potential implication is that ImageNet-1k may have noisy labels or semantic redundancy and that GBML can automatically figure and filter out these samples. Further in-depth investigation of filtered data remains an interesting research direction. Considering that compute/memory inefficiency has traditionally been the major bottleneck in GBML applications, we also compare the relative search time for data pruning. Our result shows that SAMA demonstrates comparable or even shorter search time than heuristics-based methods. We also note that, while the original MWN (Shu et al., 2019) encounters the OOM error under our setup of batch size of 256, the throughput analysis with the reduced batch size reveals that efficient distributed training with SAMA on 4 GPUs achieves 15-20$\times$ speed up compared to the original MWN that lacks distributed training support.

8.4.3 Continued Pre-training of Large Language Models

Domain Adaptive Pre-Training (DAPT) or Task Adaptive Pre-Training (TAPT) (Gururangan et al., 2020) empirically demonstrate that additional pre-training (*i.e.*, continued pre-training) of the generic language model on the domain or task-specific data can further improve downstream performance on diverse benchmarks. However, the inclusion of low-quality samples for continued pre-training tasks can potentially hinder pre-training by amplifying negative interference, which could lead to suboptimal downstream performance. Here, we attempt to minimize such negative interference by re-weighting samples from the continued pre-training task with SAMA. To this end, we adopt the auxiliary learning technique from TARTAN (Dery et al., 2022) and simplify the two-stage pre-training and finetuning pipeline into a one-stage multitask learning pipeline with the re-weighting scheme applied to the pre-training loss. The bilevel optimization formulation is as follows

$$\lambda^* = \underset{\lambda}{\operatorname{argmin}}\ L_{ft}(\mathcal{D}_{ft}; \theta^*(\lambda))$$
$$\text{s.t. } \theta^*(\lambda) = \underset{\theta}{\operatorname{argmin}}\ L_{ft}(\mathcal{D}_{ft}; \theta) + \frac{1}{|\mathcal{D}_{pt}|} \sum_{x \in \mathcal{D}_{pt}} w(x; \lambda) \cdot L_{pt}(x; \theta)$$

where L_{ft}/L_{ft} are fine-tuning/pre-training loss functions, $\mathcal{D}_{ft}$/$\mathcal{D}_{pt}$ are fine-tuning/pre-training datasets, and $w(\cdot; \lambda)$ is the data re-weighting network. For more details, we defer readers to Dery et al. (2022). Following the experiment setup in TARTAN (Dery et al., 2022), we use task-specific data and a masked language modeling loss in our auxiliary task and perform experiments with RoBERTa-base on 4 datasets from the Gururangan et al. (2020). We compare our SAMA-based neural data optimization against DAPT and TARTAN-MT. We exclude TAPT and TARTAN-Meta respectively because (1) TAPT consistently underperforms TARTAN-MT (Deleu et al., 2019) and (2) TARTAN-Meta uses additional validation data in the meta level of the downstream tasks, making the comparison unfair. We report our experiment results in Table 8.1.

We show that **SAMA-based data optimization leads to improvements in downstream performance on most of the considered datasets**. This indirectly demonstrates that SAMA-based data reweighting can identify more/less relevant data in the auxiliary task and accordingly

	ChemProt	HyperPartisan	ACL-ARC	SciERC	Average
Baseline	82.70 (0.45)	89.03 (2.25)	68.17 (2.52)	79.83 (0.89)	79.93
DAPT	84.17 (0.50)	87.23 (3.65)	71.84 (4.78)	80.42 (1.57)	80.92
TARTAN-MT	84.18 (0.30)	94.64 (0.91)	**72.41** (1.94)	80.83 (0.71)	83.02
SAMA (ours)	**84.49** (0.13)	**95.18** (0.03)	71.63 (1.68)	**81.84** (0.08)	**83.29**

Table 8.1: Experiment results for auxiliary learning with the continued pre-training task. Following Gururangan et al. (2020), we report test micro-F1 for ChemProt and macro-F1 for the other datasets. The number in parentheses indicates the standard deviation for each experiment over 3 runs. SAMA-based data optimization leads to improvements in downstream performance on most of the considered datasets.

up-/down-weight them, unlike TARTAN-MT which allocates equal importance weights on all auxiliary data. Therefore, we expect that our method would likely benefit from additional auxiliary data by automatically figuring out and exploiting only relevant data, whereas TARTAN-MT is much more susceptible to negative transfer. While we only used task-specific data in our auxiliary task for the fair comparison with TARTAN-MT, extending auxiliary data to domain-specific or even general text data and comparing SAMA against DAPT or TARTAN-MT would be an intriguing future research direction.

8.5 Discussion

In this chapter, we approach pre-training through the lens of lifelong learning, reviewing three fundamental components—model initialization, optimization dynamics, and data. We apply methodologies derived from lifelong learning to assess the potential enhancement of pre-training. Our analysis of forgetting curves during pre-training suggests that certain learning initializations, such as MetaInit (Dauphin and Schoenholz, 2019), may not exert a significant influence on forgetting behavior. Future research could explore Transformer-specific learned initializations in a similar context (Zhu et al., 2021). Shifting focus to the DSI paradigm (Tay et al., 2022), where memorization is pivotal, we demonstrate that optimizing for flat minima markedly improves memorization. This reinforces our findings from Chapter 5, extending them to scenarios where data undergoes less distinct distribution shifts. Lastly, we investigate a meta-learning-based data selection strategy, revealing its superiority over a strategy that selects forgettable examples for pre-training vision models (Toneva et al., 2019). We conclude by showcasing the benefits of our meta-learning algorithm in re-weighting data during continual pre-training of language models, suggesting potential extensions to conventional language model pre-training in future endeavors.

Chapter 9

Conclusions and Future Work

In sum, this thesis is motivated by the overarching goal of enabling efficient lifelong learning systems, addressing the stability-plasticity dilemma while continuously learning from online and unlabeled data streams. This objective is realized through the incorporation of inductive biases into the key components of data-driven machine learning—modeling, training, and data. Specifically, in Chapter 3, our investigation into the role of pre-trained initializations in lifelong learning reveals that pre-training implicitly mitigates forgetting compared to random initialization (Mehta et al., 2023b). Utilizing loss landscape analysis, we ascertain that pre-trained initialization prevents forgetting by converging to flat minima. Additionally, in Chapter 4, we introduce a parameter-efficient expert architecture to dynamically expand the system's capacity and address the stability-plasticity dilemma (Byun et al., 2023). Building on these works, in Chapter 5, we demonstrate that explicit optimization for flat minima enhances stability in the learned networks (Mehta et al., 2023b). The introduction of a meta-learning objective in Chapter 6, incorporating a slow-learning phase for stability and a fast-learning phase for plasticity, further promotes a balance in stability-plasticity (Mehta et al., 2020). Furthermore, in Chapter 7, our exploration of lifelong semi-supervised learning supports continuous learning from unlabeled data streams, mitigating the stability-plasticity dilemma through the rehearsal of pseudo-labeled data (Mehta et al., 2023a). This thesis highlights the applicability of lifelong learning, even in scenarios without a clear distribution shift (Mehta et al., 2023a), and in Chapter 8 demonstrates substantial improvements through the application of developed techniques to continual pre-training of models (Choe et al., 2023). In the following sections, we outline promising directions for future research emerging from the work presented in this thesis.

9.1 Neural Data Optimization for Large Language Models

Reinforcement Learning from Human Feedback (RLHF) applies reinforcement learning techniques to align language models trained on general text corpora with complex human values (Lambert et al., 2022). Recently, there has been a notable increase in research leveraging RLHF for various applications, such as text summarization (Stiennon et al., 2020; Wu et al., 2021), training agents for web navigation (Nakano et al., 2021), developing open-book QA models to generate answers with specific citations (Menick et al., 2022), fine-tuning dialogue agents (Glaese et al.,

2022), and refining general large language models (Ouyang et al., 2022). However, during the fine-tuning stage with RLHF, there exists a risk of the model deviating from its initial pre-trained state, resulting in the generation of nonsensical text—essentially an instance of the forgetting phenomenon. In this scenario, gradient updates from RLHF may overwrite previously acquired knowledge. To mitigate this issue, a KL divergence term is introduced. This term penalizes the model for deviating significantly from its initial pre-trained state with each training batch. This approach proves beneficial in ensuring that the model produces reasonably coherent text. This concept aligns with regularization-based approaches to lifelong learning (Sodhani et al., 2022).

While RLHF fine-tuning has demonstrated remarkable improvements, particularly in the context of InstructGPT, it has been observed to exhibit lower performance compared to its non-RLHF counterpart (GPT-3) on specific public NLP datasets (Ouyang et al., 2022). One strategy to address this performance gap involves the continuous updating of the model with pre-training updates concurrently with RLHF fine-tuning. Ouyang et al. (2022) present evidence that this approach outperforms the simpler solution of merely increasing the KL-based regularization coefficient. Consequently, in the training process of InstructGPT, 10% of pre-training data is integrated during fine-tuning to enhance RLHF performance. This strategy, aimed at mitigating the forgetting of generic knowledge by incorporating pre-training data into RLHF fine-tuning, aligns with memory-based approaches to lifelong learning, as discussed in Section 2.4.2. Despite this approach effectively narrowing the performance drop, it does not eliminate performance drop and may potentially amplify undesirable behaviors if they exist in the pre-training data. This raises the question of how to filter the pre-training data, either for training the initial pre-trained model or incorporating it into RLHF fine-tuning.

Large language models, exemplified by GPT-3 trained on 499B tokens (Brown et al., 2020), LLaMa 2 trained on 2T tokens (Touvron et al., 2023), face challenges in continuous retraining due to the tremendous size of training data. Recently, data pruning (Sorscher et al., 2022; Wang et al., 2023) and data filtering (Ngo et al., 2021) have garnered attention in the community for their potential to enhance training efficiency, diminish semantic redundancy in training data, and filtering for toxic content. In particular, Sorscher et al. (2022) showed both theoretically and experimentally that neural scaling laws can be beaten by data pruning. In Chapter 8, we study scalable meta learning algorithms (SAMA) for data pruning (or sample reweighting) and successfully apply them to continual pre-training of language models (Choe et al., 2023). A concrete future direction involves investigating SAMA for neural data optimization, particularly in the context of pre-training large models and fine-tuning with RLHF.

9.2 Unifying Sparse and Semi-Parametric Models for Lifelong Learning

In Chapter 7, we explored continual learning within the Differentiable Search Indices (DSIs) paradigm (Tay et al., 2022), uncovering the susceptibility of the DSI model to catastrophic forgetting during incremental indexing (Mehta et al., 2023a). A plausible explanation is that document corpora are encoded into monolithic model parameters, leading to uncontrolled parameter drift during incremental indexing. In Chapter 4, we introduced parameter-efficient sparse expert

models for continual learning (Byun et al., 2023). Various data views (or domains) are encoded into distinct parameters, and during inference, inputs are routed to the relevant expert. This approach has the potential to facilitate controlled and selective updates to subsets of experts, mitigating forgetting. Recently, Li et al. (2022) proposed Branch-Train-Merge (BTM) for large language models. In BTM, independent expert language models are trained for different domains, allowing dynamic addition or removal of experts. This enables generalization to unseen domains and efficient inference through expert averaging. Further, Fernandez et al. (2023) introduced Continually and Stochastically Averaging Weights (CSAW), which achieves gradual temporal generalization through the exponential moving average of expert weights trained over discrete timesteps. While the aforementioned sparse expert architectures provide beneficial inductive biases for continual learning, the central question lingers—Is it necessary to encode everything into model parameters?

Language models relying solely on parametric models for knowledge encoding face challenges in generalization and adaptation to new knowledge over time (Lazaridou et al., 2021). To address these limitations, recent architectures have introduced non-parametric memory modules alongside parametric components. This non-exhaustive list of architectures includes the k-Nearest Neighbors Language Model (kNN-LM; Khandelwal et al., 2020), Retrieval-Augmented Generation (RAG; Lewis et al., 2020), REtrieval-Augmented Language Model (REALM; Guu et al., 2020), Semi-PArametric Language Model (SPALM; Yogatama et al., 2021), Retrieval-Enhanced Transformer (RETRO; Borgeaud et al., 2022), Non-Parametric Masked language model (NPM; Min et al., 2022), and others. These architectures differ primarily in their utilization during training or inference and the granularity of data stored in the non-parametric memory module. However, these aforementioned architectures necessitate extremely large memory modules and end up retaining all information within them. Hence, the question arises—Is it necessary to store everything in a memory module?

In Chapter 6, we presented a test-time adaptation system inspired by human learning, featuring an episodic memory module (non-parametric) alongside a parametric model (Mehta et al., 2020). This semi-parametric system selectively retains a subset of evolving data using a diversity-based data selection rule, complementing the parametric model by retrieving it during inference. One can further develop our work to explore methods for reducing the size of the memory module. However, our approach necessitates computationally prohibitive gradient updates at inference time, particularly in large language models. Recent advancements in continual learning have seen the emergence of prompt-based approaches that leverage in-context learning for efficient test-time adaptation. Prominent methods include Learning to Prompt (L2P; (Wang et al., 2022c)), Dualprompt (Wang et al., 2022b), S-Prompts (Wang et al., 2022a), and Progressive Prompts (Razdaibiedina et al., 2023). These approaches entail learning a small number of parameters per domain or task in the form of continuous token embeddings or prompts while keeping the remaining pre-trained model fixed. The appropriate prompt is then selected based on the input data. Although these methods facilitate continual learning without a memory module, they can be synergistically combined with our work in Chapter 6. This integration allows for the retention of a subset of samples in the memory module, utilizing them as prompts, similar to an in-context retrieval augmented language model (In-Context RALM; Ram et al., 2023).

In conclusion, sparse expert architectures enable targeted updates to the parametric model, while semi-parametric models allow the utilization of memory modules for in-context learning.

Each architectural element possesses unique advantages, and their integration can be mutually advantageous. Given the scale of recent large models, supporting incremental updates is vital. Simultaneously, it is crucial to maintain a reasonable size for a memory module when dealing with extensive data corpora. Combining sparse and semi-parametric models enables selective updates to sparse models and the use of non-parametric components for frequently evolving data. However, the absence of a unified framework unifying both aspects marks an essential future research direction.

9.3 Lifelong Learning and Unlearning

In this thesis, our primary focus has been on mitigating the issue of forgetting previously acquired knowledge during lifelong learning. However, there are scenarios where intentional forgetting becomes relevant. A notable example is observed in large language models, which tend to inherit biases from their pre-training corpora (Schramowski et al., 2022). Given the computational challenges associated with retraining these models every time biases are uncovered, there is a need to actively edit or update the model to unlearn the uncovered biases. Nevertheless, the insights from Chapters 3, 5, and 8 indicate that pre-trained models exhibit robust retention of information (Mehta et al., 2023b), posing challenges to the straightforward process of intentional forgetting.

Recent studies have explored efficient methods for localizing and modifying facts stored within language models (AlKhamissi et al., 2022). These approaches include fine-tuning techniques (Zhu et al., 2020; Dhingra et al., 2022), implementations with hyper-networks (De Cao et al., 2021; Mitchell et al., 2022), and direct editing methods (Meng et al., 2022). While these efforts are crucial for updating information in pre-trained language models, relying on prompting as our probing mechanism only offers a lower bound estimate of the knowledge embedded in these models (Jiang et al., 2020). Additionally, the question arises—how can we facilitate unlearning in pre-trained models while preserving their broader, generic capabilities? The emerging field of machine unlearning has recently gained attention within the research community (Xu et al., 2023). However, its development is still in its early stages, and numerous questions persist, especially concerning pre-trained models.

Appendices

Appendix A

Additional experimental details and results for an empirical investigation of the role of pre-training in lifelong learning

A.1 Implementation Details

Vision Experiments For all vision experiments, we use the full ResNet-18 He et al. (2016) architecture, with the final linear layer replaced (the number of outputs corresponds to the total number of classes in all given tasks). During inference, only the subset of outputs corresponding to the given task is considered. All images are resized to 224×224, and normalized with $\mu = (0.485, 0.456, 0.406)$ and $\sigma = (0.229, 0.224, 0.225)$. We used a SGD optimizer with the learning rate set to $.01$ for all methods (we did a hyperparameter search for both pre-trained and randomly initialized models and found the learning rate 0.01 resulted in a good learning accuracy for both pre-trained and randomly initialized models). The batch size was set to 10 for the Split CIFAR-50 and Split CIFAR-100 experiments and 64 for the 5-dataset-CV experiments. The memory per class for ER was set to 1, and the λ parameter for EWC was also set to 1.

For Stable SGD, we performed a hyperparameter sweep over the parameters specified in the original paper, namely:

- initial learning rate: [.25 (Split CIFAR-100-R, Split CIFAR-50-R, 5-dataset-CV-R), .1, .01 (Split CIFAR-100-PT, Split CIFAR-50-PT), .001 (5-dataset-CV-PT)]
- learning rate decay: [0.9 (Split CIFAR-50-R, 5-dataset-CV-R, Split CIFAR-100-PT), 0.85 (Split CIFAR-100-R, Split CIFAR-50-PT), 0.8 (5-dataset-CV-PT)]
- batch size: [10 (all), 64]
- dropout: [0.5 (5-dataset-R), 0.25 (Split CIFAR-100-R, Split CIFAR-50-R, Split CIFAR-100-PT, Split CIFAR-50-PT, 5-dataset-CV-PT)]

NLP Experiments For most of the text classification experiments, we use the Transformer architecture-based text encoder, DistilBERT-base (Sanh et al., 2019) to encode our input. In a

single-sentence text classification task, x_t is an input sentence to be classified. In a sentence-pair classification task, the concatenation of x_t^1 and x_t^2 sentences separated by a $[SEP]$ symbol is considered as an input x_t. DistilBERT produces a contextual representation of each token in x_t including a special beginning of the sentence token symbol $[CLS]$. We use the representation of the $[CLS]$ symbol from the model as features for a linear task classifier. We have a separate classifier for each task. We mainly set hyper-parameters to default implementation from HuggingFace.[1] We use Adam as our optimizer, set dropout 0.1, the base learning rate to $2e^{-5}$, batch size to 32 and the maximum total input sequence length after tokenization to 128. For EWC, we set the regularization strength λ to 100 (as this ended up with comparable LA across other methods), and for ER, following (Chaudhry et al., 2019), the memory per class per task is set to 1. For SAM, we set $\rho = 0.02$ for all models (random as well as pre-trained) on 5-dataset-NLP and 15-dataset-NLP. For SplitYahooQA we set $\rho = 0.001$.

Sharpness metric. The matrix $A \in \mathbb{R}^{n \times p}$ used for projecting the parameters onto a subspace is randomly sampled and then normalized row-wise. Since this matrix is very large, the computation of the pseudo-inverse A^+ (required for calculating the bounds in Equation 3.2) is very memory intensive and unstable. Instead, we directly calculate A^+x by finding the least squares solution to $Ab = x$. To find the maximum referenced in Equation 3.3, we use the L-BFGS-B algorithm.[2] We set the maximum number of iterations for the algorithm to 10, and to speed up computation, we directly provide the gradients along with the loss to the algorithm, instead of using the default 2-point finite difference gradient estimation.

For ResNet-18 ($n = 11M$), we set $p = 100$. However, for DistilBERT ($n = 66M$) when we set $p = 100$, we notice extremely small values for the sharpness metric. With the increase in the number of parameters, n, we should ideally increase random subspace projection dimension p. Setting larger $p(> 100)$ values for DistilBERT, however, leads to memory issues relating to allocating space for A and computing the bounds (even with the more efficient method discussed above). So instead of evaluating the sharpness metric in a random manifold, we perform the maximization in the entire space $\mathbb{R}^n$ (basically setting $A = I_n$). According to Keskar et al. (2017), when ϵ is small enough and $A = I_n$, the sharpness metric in Equation 3.3 relates to the largest eigenvalue of $\nabla^2 L(x)$.

A.2 Task-specific results

In order to understand the evolution of task-specific performance during continuous training, we visualize the task-specific results in Figures A.1 and A.2. Specifically, we compare the performance of pre-trained and randomly initialized ResNet-18/ DistilBERT, for the first three tasks in a sequence, across five random task ordering, when evaluated on 5-dataset-CV/5-dataset-NLP (diverse tasks). In general, we see that both models start with approximately equal task accuracy (except for CIFAR-10), but pre-trained initialization leads to lesser forgetting than randomly initialized models (consistent with our observation in Figure 3.1 for Split YahooQA).

[1] https://github.com/huggingface/transformers

[2] We used the implementation provided by scipy at https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html

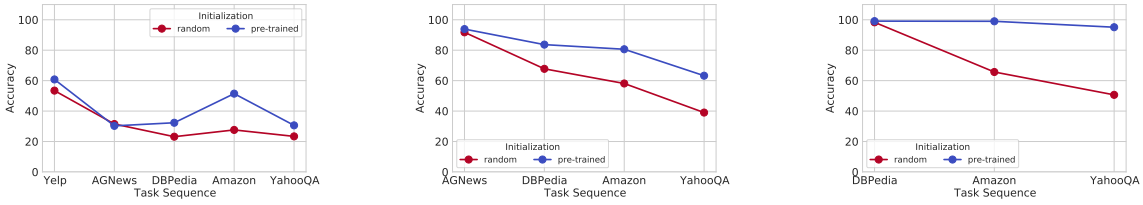
Moreover, given the heterogeneous nature of the downstream tasks, we see that performance gains (in terms of forgetting) from pre-trained initialization vary across different tasks.

5-dataset-NLP For example, in the case of DBPedia (Figures A.1c, A.1d, A.1o) and AGNews (Figures A.1b, A.1f, A.1j) datasets, we see pre-trained DistilBERT undergoes little to almost no forgetting. One plausible explanation for these results is that both datasets are for the article classification tasks, DBPedia is Wikipedia article classification (14 classes) and AGNews is news article classification (4 classes), and share similar domains with the pre-training corpora (Wikipedia and Books). On the other hand, we see a significant forgetting in the case of Yelp (Figures A.1a, A.1g, A.1k) and Amazon datasets (Figure A.1l). Both of these datasets are review sentiment classification tasks (5 classes). We know that the reviews domain (noisy text from Yelp.com and Amazon.com) is less similar with the pre-training corpora (clean text from Wikipedia and Books), and might be one of the reasons behind the drop in performance. Further, note that as we train on the sequence of tasks, we expect to see positive/ negative transfer from related/ unrelated tasks. For example, we see that the performance on Yelp improves significantly after training on Amazon (Figures A.1a, A.1g, A.1n), demonstrating an example of positive transfer from the related task.

5-dataset-CV Here, we report that the forgetting is more severe for SVHN (Figures A.2a, A.2d, A.2h) and CIFAR-10 (Figures A.2g, A.2l, A.2m) as compared to MNIST (Figures A.2e, A.2n), notMNIST (Figures A.2b, A.2f, A.2i, A.2o). Although SVHN and MNIST both are digit recognition tasks, we believe that the realistic nature (house numbers in Google Street View images) of SVHN images makes them more susceptible to forgetting, even in the case of pre-trained ResNet-18 models.

A.3 Loss Contours

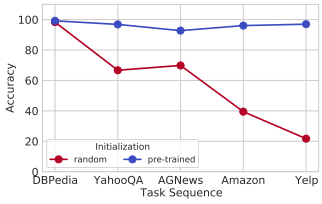
In this section we present loss contours for task 1/ task 2 for all task sequences (refer to Section 3.2 for task sequences) for **5-dataset-NLP**, **Split YahooQA**, **Split CIFAR-50**, and **5-dataset-CV**. In line with our observation from the sharpness and linear model interpolation analyses, pre-trained initialized models lead to flatter task minima for subsequent tasks as well.



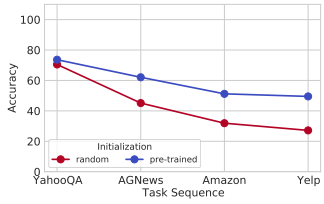
(a) Yelp (Seq1)

(b) AGNews (Seq1)

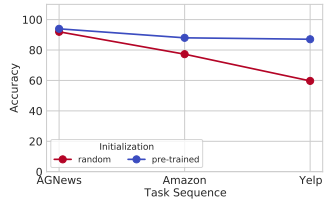
(c) DBPedia (Seq1)



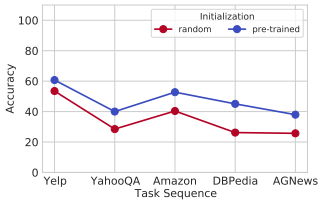
(d) DBPedia (Seq2)



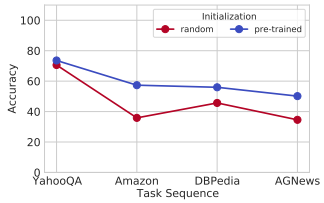
(e) YahooQA (Seq2)



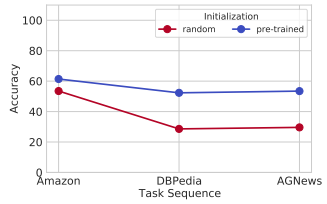
(f) AGNews (Seq2)



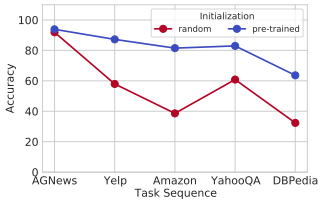
(g) Yelp (Seq3)



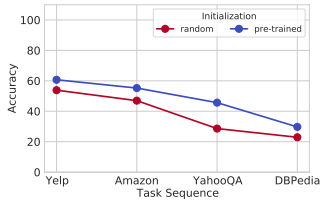
(h) YahooQA (Seq3)



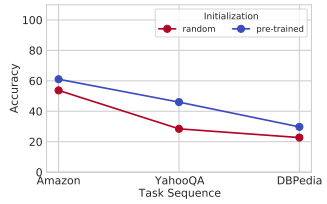
(i) Amazon (Seq3)



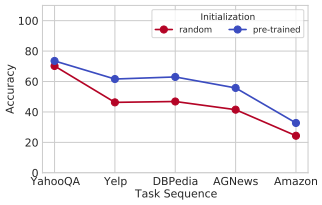
(j) AGNews (Seq4)



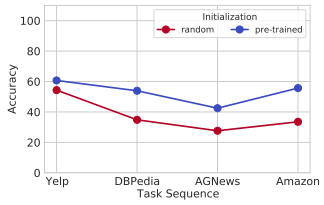
(k) Yelp (Seq4)



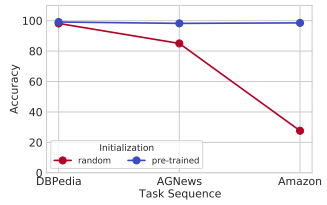
(l) Amazon (Seq4)



(m) YahooQA (Seq5)

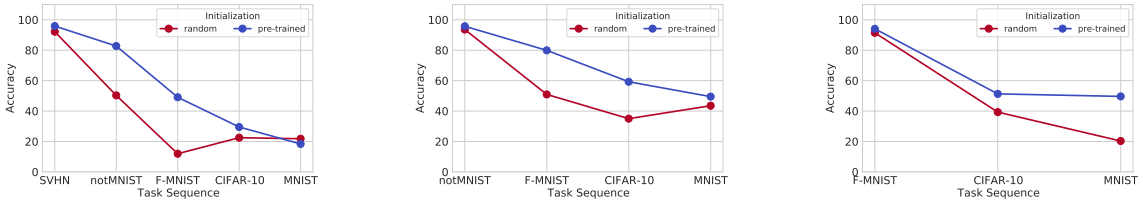


(n) Yelp (Seq5)

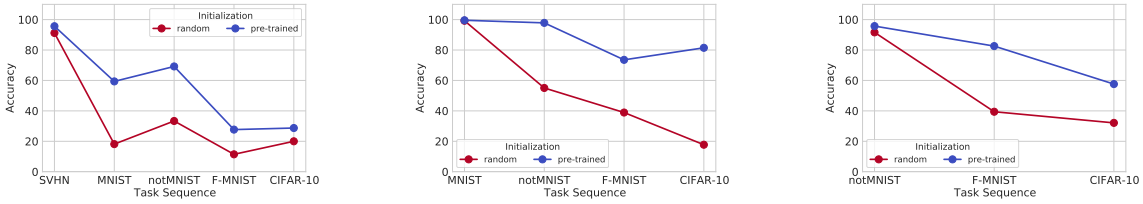


(o) DBPedia (Seq5)

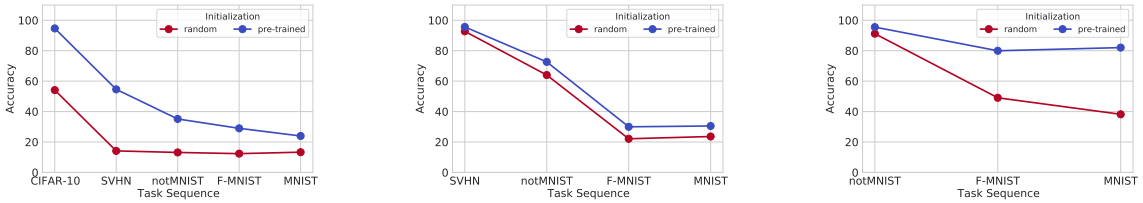
Figure A.1: Evolution of task accuracy during sequential training on **5-dataset-NLP**. We compare the performance of pre-trained and randomly initialized models, for first three tasks in a sequence, across five different random task orderings (Seq1, Seq2, Seq3, Seq4, Seq5). We see that both models start with approximately equal task accuracy, but pre-trained initialized models undergo lesser forgetting than randomly initialized models.



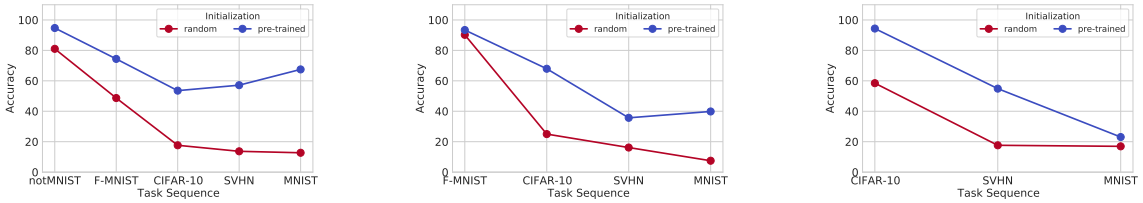
(a) SVHN (Seq1) (b) notMNIST (Seq1) (c) Fashion-MNIST (Seq1)



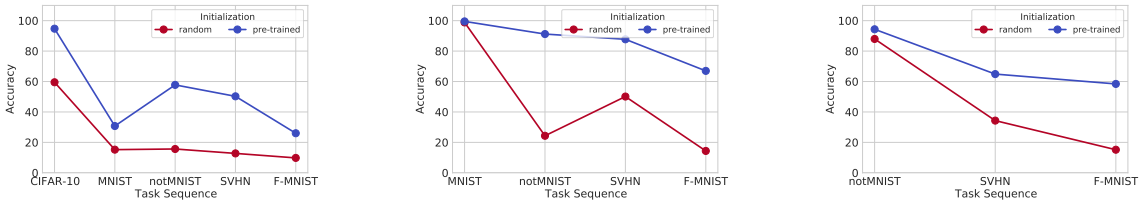
(d) SVHN (Seq2) (e) MNIST (Seq2) (f) notMNIST (Seq2)



(g) CIFAR-10 (Seq3) (h) SVHN (Seq3) (i) notMNIST (Seq3)

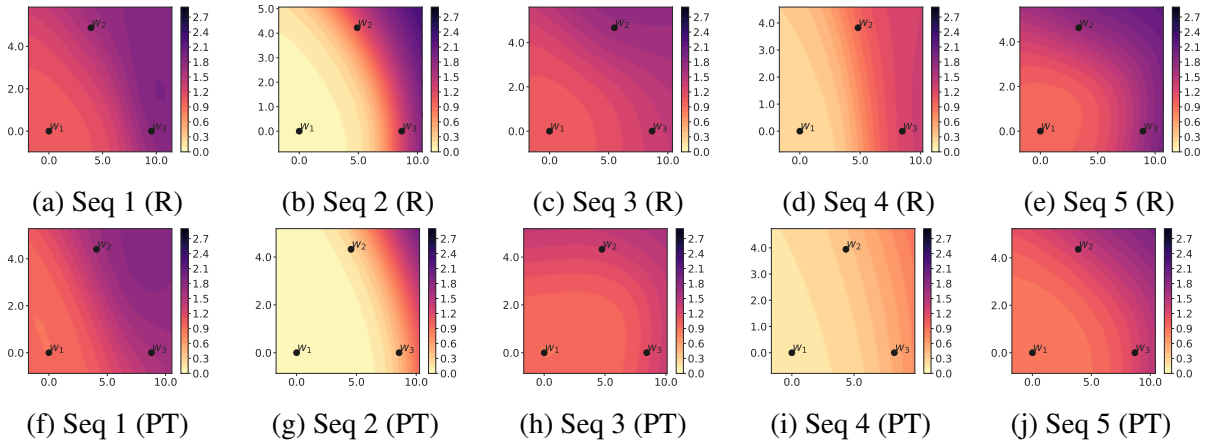


(j) notMNIST (Seq4) (k) Fashion-MNIST (Seq4) (l) CIFAR-10 (Seq4)



(m) CIFAR-10 (Seq5) (n) MNIST (Seq5) (o) notMNIST (Seq5)

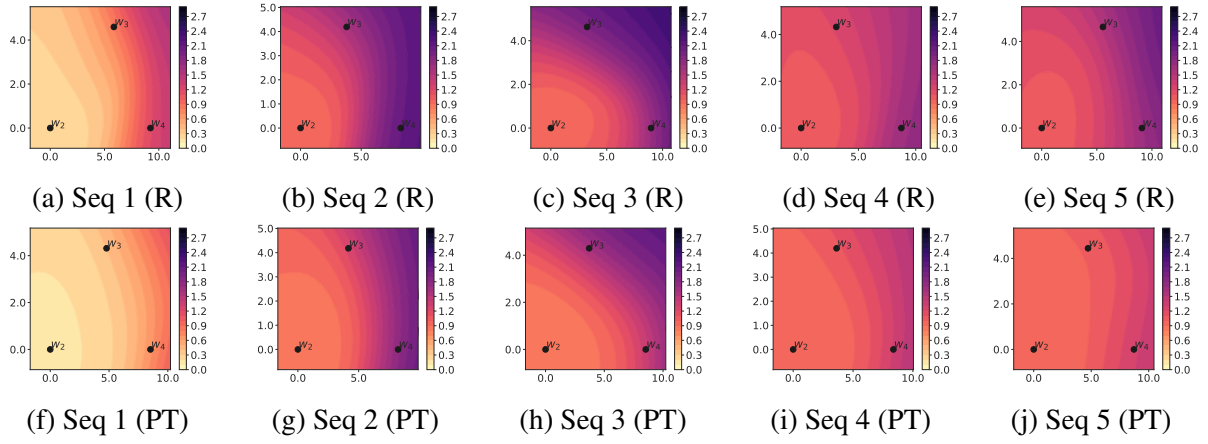
Figure A.2: Evolution of task accuracy during sequential training on **5-dataset-CV**. We compare the performance of pre-trained and randomly initialized models, for first three tasks in a sequence, across five different random task orderings (Seq1, Seq2, Seq3, Seq4, Seq5). We see that both models start with approximately equal task accuracy (except for CIFAR-10), but pre-trained initialized models undergo lesser forgetting than randomly initialized models.



(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

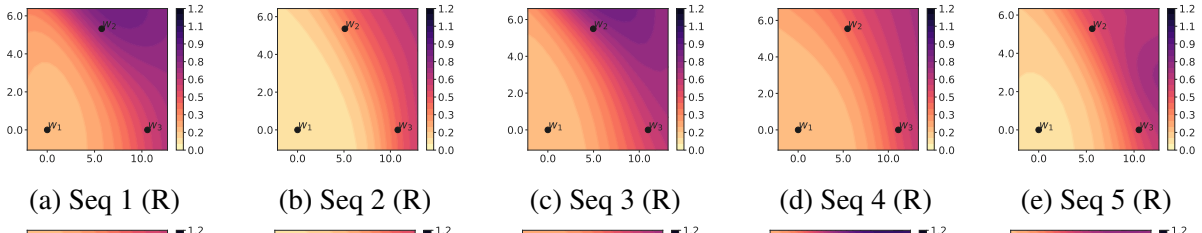
Figure A.3: Loss contours for **Task 1** on 5 task sequences of **5-dataset-NLP**. Each contour shows the location of the model parameters after training sequentially on **Task 1 ($\mathbf{w}_1$), Task 2 ($\mathbf{w}_2$), Task 3 ($\mathbf{w}_3$)**. The top row shows contours for randomly initialized models (R) and the bottom row shows contours for pre-trained initialized models (PT).



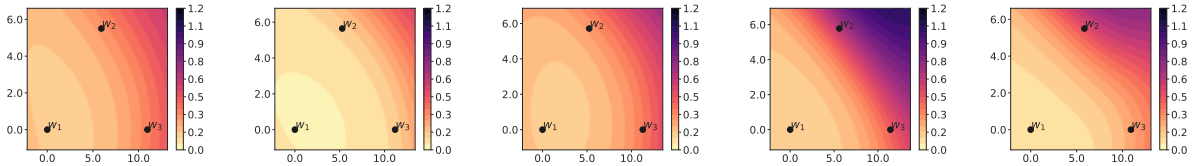
(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

Figure A.4: Loss contours for **Task 2** on 5 task sequences of **5-dataset-NLP**.

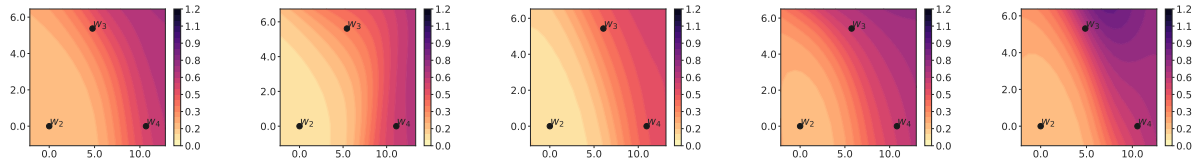


(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

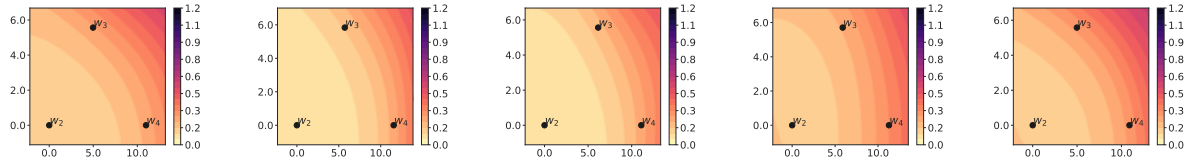


(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

Figure A.5: Loss contours for **Task 1** on 5 task sequences of **Split YahooQA**.

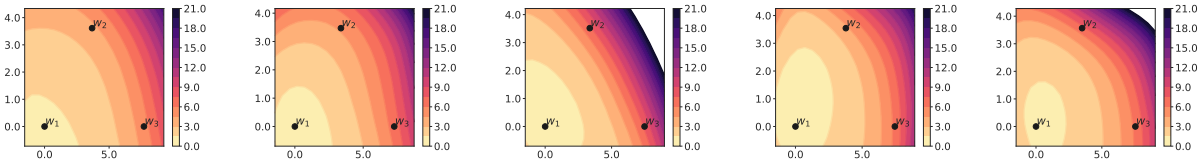


(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

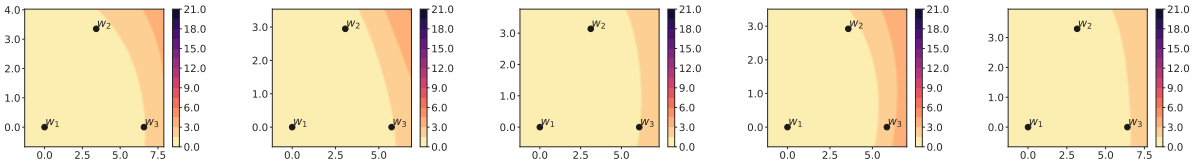


(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

Figure A.6: Loss contours for **Task 2** on 5 task sequences of **Split YahooQA**.

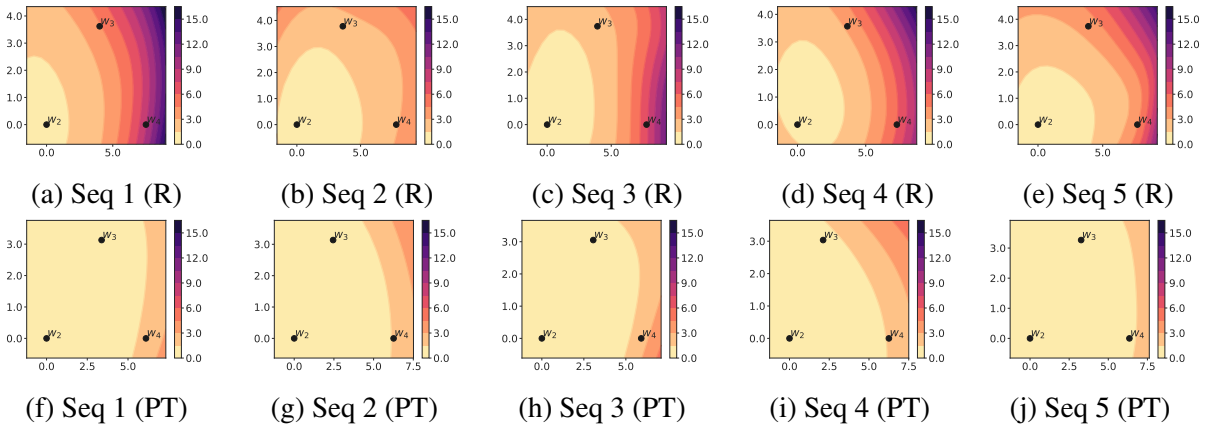


(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)



(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

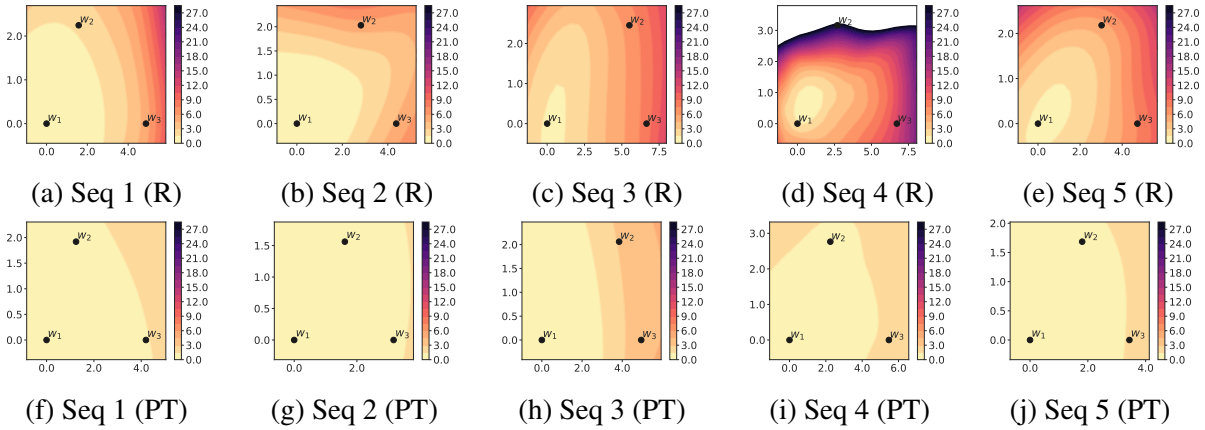
Figure A.7: Loss contours for **Task 1** on 5 task sequences of **Split CIFAR-50**.



(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

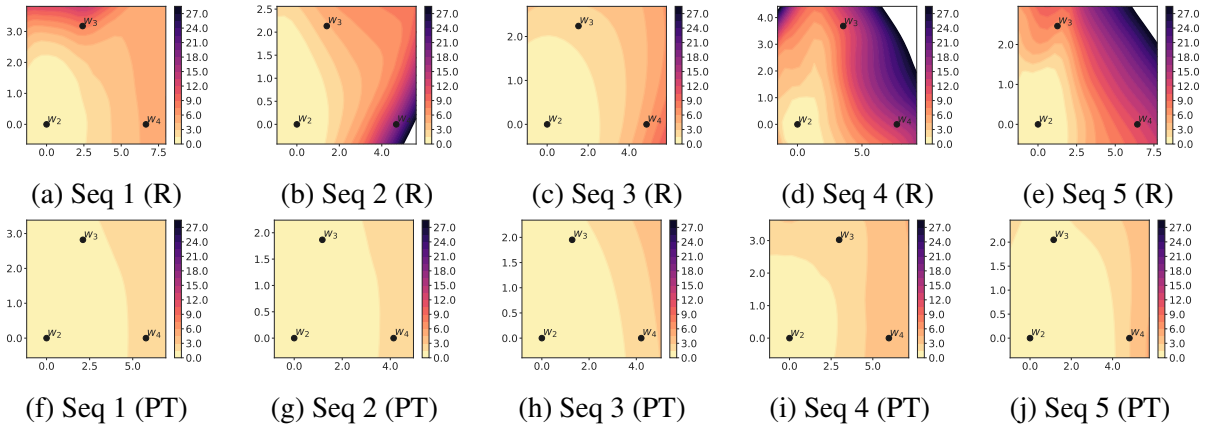
Figure A.8: Loss contours for **Task 2** on 5 task sequences of **Split CIFAR-50**.



(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

Figure A.9: Loss contours for **Task 1** on 5 task sequences of **5-dataset-CV**.



(a) Seq 1 (R) (b) Seq 2 (R) (c) Seq 3 (R) (d) Seq 4 (R) (e) Seq 5 (R)

(f) Seq 1 (PT) (g) Seq 2 (PT) (h) Seq 3 (PT) (i) Seq 4 (PT) (j) Seq 5 (PT)

Figure A.10: Loss contours for **Task 2** on 5 task sequences of **5-dataset-CV**.

Appendix B

Additional experimental details and results for efficient meta lifelong-learning with limited memory

B.1 Dataset Specific Results

We report per-dataset specific results for text classification in Table B.1 and for question answering in Table B.2. For A-GEM and MbPA++ baselines, we obtain results from (d'Autume et al., 2019). A-GEM, Replay, MbPA++ and MbPA++ (Our Impl.) methods use $r_{\mathcal{M}} = 100\%$ memory size while our proposed framework, Meta-MbPA, and MbPA++(1%) use $r_{\mathcal{M}} = 1\%$ memory size.

B.2 Single Task and Multi-task Models Results

We report results for single-task models that uses only single-task data and multi-task learning models using different amounts of training data in Table B.3. For text classification, we report accuracy scores and for question answering, we report F_1 scores.

B.3 Catastrophic Forgetting

To understand the severity of the catastrophic forgetting of different models, in Figure 6.2 and Table B.4, we report the performance on the first dataset as training progresses. For example, we show results for AGNews as we train different models on AGNews→Yelp→Amazon→Yahoo→DBpedia dataset order in lifelong learning setup. We also show the results prior to training on any dataset (denoted by "Initial").

Order	Dataset	FT	Online EWC	A-GEM†	Replay	MbPA++†	MbPA++ (Our Impl.)	MbPA++ ($r_{\mathcal{M}} = 1\%$)	Meta-MbPA ($r_{\mathcal{M}} = 1\%$)
i	1	2.0	29.7	42.5	49.2	45.7	59.2	54.2	62.1
	2	4.3	0.1	89.8	50.1	91.6	94.0	91.0	93.7
	3	95.8	97.5	96.0	98.7	96.3	98.5	98.5	99.1
	4	1.3	18.5	56.8	45.2	54.6	57.7	56.7	60.7
	5	74.2	73.2	68.2	74.0	65.6	67.2	66.7	73.8
	Average	35.5	43.8	70.7	63.4	70.8	75.3	73.4	77.9
ii	1	62.2	89.9	80.1	98.7	95.8	98.5	98.0	99.0
	2	0.0	0.1	50.3	54.6	63.1	69.7	61.7	70.2
	3	39.4	40.3	91.3	89.3	92.2	95.0	93.0	92.5
	4	61.3	60.0	57.3	61.5	55.7	55.2	55.2	60.1
	5	61.2	58.5	50.6	61.1	47.7	54.7	52.7	61.5
	Average	44.8	49.8	65.9	73.0	70.9	74.6	72.1	76.7
iii	1	11.4	52.5	41.1	54.8	44.3	59.2	53.7	59.6
	2	2.1	14.9	55.0	31.9	62.7	67.7	60.2	70.2
	3	12.8	40.3	54.6	52.0	54.4	58.2	60.7	63.8
	4	92.5	98.0	93.3	97.4	96.2	98.5	98.0	98.9
	5	93.3	91.8	93.6	93.1	93.4	94.5	92.5	94.1
	Average	42.4	59.5	67.5	65.8	70.2	75.6	73.0	77.3
iv	1	0.0	31.9	90.8	80.3	91.8	94.0	91.0	93.1
	2	8.3	33.3	44.9	59.3	44.9	57.2	54.2	60.8
	3	3.6	22.2	60.2	59.6	55.7	59.7	61.2	61.6
	4	31.8	73.5	65.4	71.9	65.3	68.7	63.7	73.6
	5	99.1	98.9	56.9	99.1	95.8	98.0	98.5	99.1
	Average	28.6	52.0	63.6	74.0	70.7	75.5	73.7	77.6

Table B.1: **Dataset specific accuracy for text classification tasks for different dataset orders and models.** † Results obtained from (d'Autume et al., 2019). Where applicable, we use $r_{\mathcal{M}} = 100\%$ unless denoted otherwise.

Order	Dataset	FT	Online EWC	A-GEM†	Replay	MbPA++†	MbPA++ (Our Impl.)	MbPA++ ($r_{\mathcal{M}} = 1\%$)	Meta-MbPA ($r_{\mathcal{M}} = 1\%$)
i	1	40.5	42.9	36.7	44.1	47.2	44.3	42.6	49.9
	2	60.1	57.4	51.8	60.7	57.7	62.9	60.0	63.1
	3	58.2	53.8	53.4	58.7	58.9	61.2	58.8	61.5
	4	85.0	77.7	82.5	85.5	84.3	84.7	86.8	84.7
	Average	60.9	58.0	56.1	62.3	62.0	63.3	62.0	64.8
ii	1	66.8	78.8	64.2	73.1	72.6	80.4	81.8	80.4
	2	64.2	59.5	62.5	64.2	63.4	65.3	60.7	61.5
	3	31.4	28.6	43.4	41.0	50.5	42.0	41.6	52.1
	4	66.7	61.9	63.5	66.8	63.0	66.1	64.3	67.0
	Average	57.3	57.2	58.4	61.3	62.4	63.5	62.1	65.3
iii	1	41.6	57.2	47.6	58.7	56.0	62.0	59.4	65.7
	2	38.8	51.9	47.0	54.2	56.8	53.4	57.3	59.2
	3	54.4	63.1	57.4	67.7	78.0	81.8	83.9	80.7
	4	53.1	25.5	57.4	52.7	54.9	49.0	46.9	52.1
	Average	47.0	49.5	52.4	58.3	61.4	61.6	61.8	64.4
iv	1	58.1	60.5	54.8	59.4	59.0	58.9	60.8	61.3
	2	39.8	36.3	38.8	45.0	48.7	43.5	39.2	50.4
	3	60.5	60.4	53.4	61.6	58.1	64.2	61.3	63.7
	4	85.6	77.3	84.7	85.6	83.6	82.8	85.3	84.5
	Average	61.0	58.7	57.9	62.9	62.4	62.4	61.6	65.0

Table B.2: **Dataset specific F_1 scores for question answering tasks for different dataset orders and models.** † Results obtained from (d'Autume et al., 2019). Where applicable, we use $r_{\mathcal{M}} = 100\%$ unless denoted otherwise.

Dataset	Single Model	MTL (1%)	MTL (10%)	MTL (100 %)
Text Classification				
AGNews	93.6	83.1	88.7	94.0
Amazon	61.8	38.6	54.2	63.5
DBPedia	99.2	78.1	91.4	99.3
Yahoo	74.9	15.8	65.6	75.3
Yelp	61.9	36.4	52.8	62.6
Average	78.28	50.4	70.5	78.9
Question Answering				
QuAC	54.0	20.9	30.9	53.5
SQuAD	87.8	60.5	75.2	88.1
Trivia Web	65.8	49.2	62.2	67.7
Trivia Wikipedia	62.9	45.9	56.5	64.9
Average	67.6	44.1	56.2	68.6

Table B.3: **Single model and Multi-Task Learning (MTL) results for text classification and question answering tasks.** MTL ($X\%$) denotes $X\%$ of the training examples are used per dataset to train MTL models.

First Dataset	Dataset	FT	Online EWC	Replay	MbPA++ (Our Impl.)	Meta-MbPA ($r_{\mathcal{M}} = 1\%$)
Text Classification						
AGNews	0 (Initial)	0.2	0.2	0.2	0.2	0.2
	1 (AGNews)	94.2	94.1	94.0	93.5	94.3
	2 (Yelp)	45.9	78.2	92.4	94.5	94.1
	3 (Amazon)	30.2	62.5	87.9	93.0	93.5
	4 (Yahoo)	0.0	9.2	74.4	92.0	93.1
	5 (DBPedia)	0.0	31.9	80.3	93.0	93.1
Yelp	0 (Initial)	0.2	0.2	0.2	0.2	0.2
	1 (Yelp)	62.5	62.0	62.5	57.7	62.5
	2 (Yahoo)	4.3	32.3	58.1	56.7	61.0
	3 (Amazon)	60.4	61.7	60.1	55.7	61.2
	4 (DBPedia)	48.6	61.4	60.3	58.2	61.4
	5 (AGNews)	11.4	52.4	54.8	57.7	59.6
Question Answering						
QuAC	0 (Initial)	14.1	14.1	14.1	14.1	14.1
	1 (QuAC)	51.8	51.8	51.3	50.8	51.8
	2 (TrWeb)	28.7	37.8	40.4	41.3	51.6
	3 (TrWik)	27.0	35.3	38.8	39.8	50.9
	4 (SQuAD)	40.5	42.9	43.7	44.0	49.9
SQuAD	0 (Initial)	7.5	7.5	7.5	7.5	7.5
	1 (SQuAD)	87.2	87.2	86.6	88.6	86.8
	2 (TrWik)	65.1	79.8	69.6	78.4	85.5
	3 (QuAC)	48.5	70.0	54.4	76.2	79.0
	4 (TrWeb)	66.8	78.8	69.4	81.5	80.4

Table B.4: **Performance of the first dataset as training progresses for text classification and question answering tasks over different dataset orders and models.** Where applicable, we use $r_{\mathcal{M}} = 100\%$ unless denoted otherwise. “0 (Initial)" denotes model before training on any dataset.

Appendix C

Additional experimental details and results for DSI++

C.1 Dataset

Dataset	#D	Natural Questions (NQ)			MS MARCO		
		#Train	#Validation	#Test	#Train	#Validation	#Test
R_0	50K	53.8K	13.5K	3.9K	2M	25.0K	3.6K
R_1	10K	10.7K	2.7K	809	400K	5.1K	762
R_2	10K	10.6K	2.7K	787	400K	5.1K	770
R_3	10K	10.7K	2.7K	727	400K	4.9K	734
R_4	10K	10.9K	2.7K	772	400K	4.9K	730
R_5	10K	10.7K	2.7K	847	400K	4.9K	660

Table C.1: DSI++ dataset statistics for NQ and MS MARCO: memorization and retrieval tasks.

C.2 Additional Results

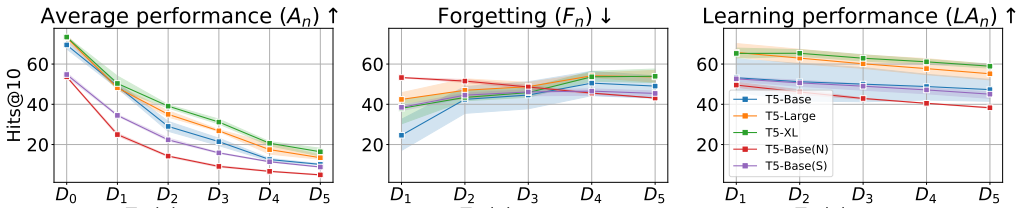


Figure C.1: Systematic study about forgetting and forward transfer when incrementally indexing new corpus of documents across different model sizes (T5-Base, T5-Large, T5-XL) and docid representations. We use atomic docids by default and denote (N)/(S) for naively/semantically structured string docids. ↑ indicates higher is better, ↓ indicates lower is better. We observe that by increasing the model scale, the average A_n and learning LA_n performance improves. However, forgetting F_n is severe across all model scales. Moreover, we observe that naive string docids (N) underperforms atomic docids across Hits@10 metric. Similar to Figure 7.2, imbuing the docid space with semantic (S) structure alleviates the forgetting compared to an arbitrary/ naive (N) structure.

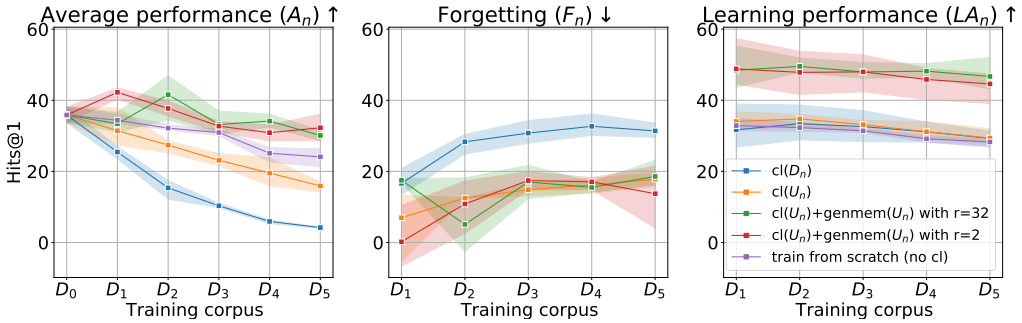


Figure C.2: Investigating the effectiveness of generative memory in mitigating forgetting when continuously indexing new corpus D_n (T5-Base model and atomic docids representation) for the NQ dataset. ↑ indicates higher is better, ↓ indicates lower is better. We observe that continual indexing of old and new documents cl(U_n) help to alleviate forgetting of older documents when evaluated on retrieval tasks. However, average Hits@1 (A_n) still undergo 19 points drop after sequential updates ($D_0 \rightarrow D_1 \cdots \rightarrow D_5$). We observe that by augmenting generative memory during continual indexing not only reduces the forgetting (F_n) but also improves average Hits@1 (A_n) by $+17.3\%$ over continual indexing.

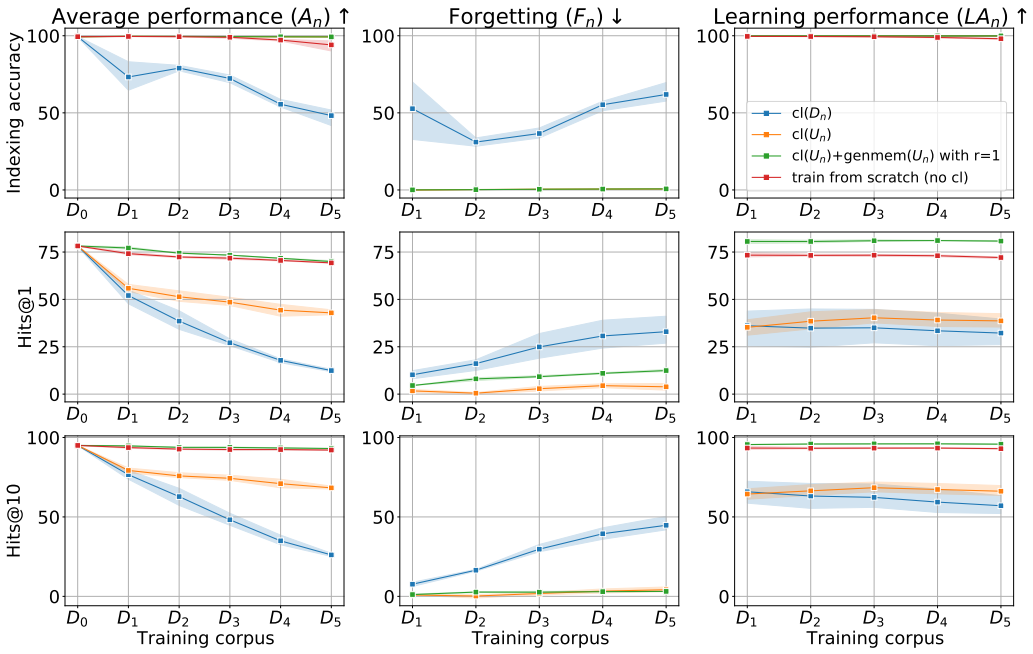


Figure C.3: Investigating the effectiveness of generative memory in mitigating forgetting when continuously indexing new corpus D_n (T5-Base model and atomic docids representation) for the MS MARCO dataset. ↑ indicates higher is better, ↓ indicates lower is better. We observe that continual indexing of old and new documents cl(U_n) helps to alleviate forgetting of older documents when evaluated on retrieval tasks. However, average Hits@10 (A_n) still undergo 25.0 points drop after sequential updates ($D_0 \rightarrow D_1 \cdots \rightarrow D_5$). Generative memory enables sparse replaying of pseudo-queries for old documents and continual semi-supervised learning with new documents. We observe that augmenting generative memory during continual indexing not only reduces the forgetting (F_n) but also improves average Hits@10 (A_n) by $+23.0\%$ over considered baselines.

Bibliography

Amro Kamal Mohamed Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*. 8.4

Mohamed Abdelsalam, Mojtaba Faramarzi, Shagun Sodhani, and Sarath Chandar. 2021. Iirc: Incremental implicitly-refined classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11038–11047. 3.3

Sandra Ackerman. 1992. Discovering the brain. National Academies Press (US). 1.1

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763. 4.1, 4.4.3

Rahaf Aljundi. 2019. Continual learning in neural networks. *arXiv preprint arXiv:1910.02718*. 1

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154. 3.5

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375. 3.5, 4.1, 4.5

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*. 7.5, 9.3

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. 2021. Ext5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*. 3.6

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giridharan Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeffrey Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Veselin Stoyanov. 2022. Efficient large scale language modeling with mixtures of experts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 1.1

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation.

In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics. 6.3.3

Samy Bengio, Krzysztof Dembczynski, Thorsten Joachims, Marius Kloft, and Manik Varma. 2019. Extreme classification (dagstuhl seminar 18291). In *Dagstuhl Reports*, volume 8. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 8.3

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics. 1, 1.1

Maureen M Black, Susan P Walker, Lia C H Fernald, Christopher T Andersen, Ann M DiGirolamo, Chunling Lu, Dana C McCoy, Günther Fink, Yusra R Shawar, Jeremy Shiffman, Amanda E Devercelli, Quentin T Wodon, Emily Vargas-Barón, and Sally Grantham-McGregor. 2017. Early childhood development coming of age: science through the life course. *The Lancet*, 389(10064):77–90. 1.1

Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144*. 7.5

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240. PMLR. 7.5, 9.2

Léon Bottou. 1999. *On-line Learning and Stochastic Approximations*. Cambridge University Press. 1.1

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901. 1, 1.1, 7.3, 8.1, 8.4, 9.1

Yaroslav Bulatov. 2011. Notmnist dataset. Technical report, Google (Books/OCR). 2.3.3

Yewon Byun, Sanket Vaibhav Mehta, Saurabh Garg, Emma Strubell, Bryan Wilder, and Zachary Chase Lipton. 2023. Generate to discriminate: Expert routing for continual learning. In *Submitted to The Twelfth International Conference on Learning Representations*. Under review. 1.2, 4.1, 1, 9, 9.2

Massimo Caccia, Pau Rodríguez, O. Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam H. Laradji, I. Rish, Alexandre Lacoste, David Vázquez, and Laurent Charlin. 2020. Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning. In *NeurIPS*. 3.5

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75. 1

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018a. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In

Proceedings of the European conference on computer vision (ECCV), pages 532–547. 3.5

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018b. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*. 2.3.3, 2.4.2, 2.4.2, 3.1, 3.2.3, 3.5, 4.5, 6.1, 6.2.3, 6.3.2, 7.3

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*. 2.4.2, 2.4.2, 2.4.2, 3.1, 3.2.2, 3.2.3, 3.5, 4.2.3, 4.5, 5.3, 6.2.1, 6.2.3, 7.3, 7.5, A.1

Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. 2018. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. In *International Conference on Machine Learning*, pages 873–882. PMLR. 1.1, 5.5

Zhiyuan Chen and Bing Liu. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207. 1, 1.1

Sang Keun Choe, Sanket Vaibhav Mehta, Hwijeen Ahn, Willie Neiswanger, Pengtao Xie, Emma Strubell, and Eric Xing. 2023. Making scalable meta learning practical. *Advances in neural information processing systems*, 36. 1.2, 8.4, 8.4.1, 9, 9.1

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184. 2.3.2

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113. 1, 1.1, 8.1

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936. 2

Yann N Dauphin and Samuel Schoenholz. 2019. Metainit: Initializing learning by learning to initialize. *Advances in Neural Information Processing Systems*, 32. 1.1, 5.5, 5.5, 5.5.1, 5.5.1, 5.5.1, 8.2, 8.5

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems*. (document), 2.3.1, 2.3.2, 6.1, 6.1, 6.2.1, 6.3.1, 6.3.2, 6.3.3, 6.5, 6.5, 6.1, 6.5.3, B.1, B.1, B.2

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506. 7.5, 9.3

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting

in classification tasks. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385. 7.5

Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. 2019. Torchmeta: A Meta-Learning library for PyTorch. Available at: https://github.com/tristandeleu/pytorch-meta. 8.4.3

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 3.5

Lucio M. Dery, Paul Michel, Ameet Talwalkar, and Graham Neubig. 2022. Should we be pre-training? an argument for end-task aware training as an alternative. In *International Conference on Learning Representations*. 8.4.3

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, page 4171–4186. (document), 1.1, 1.1, 2.4.3, 3.1, 3.3.3, 3.5, 4.3.1, 4.2, 6.3.1, 7.5, 8.1, 8.2

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. 2019. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146. 3.5

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273. 7.5, 9.3

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE. 1.1

Pinar Donmez, Jaime G Carbonell, and Paul N Bennett. 2007. Dual strategy active learning. In *European Conference on Machine Learning*, pages 116–127. Springer. 6.5.3

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. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. 1.1

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR. 1, 8.1

Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. 2020. Adversarial continual learning. In *Computer Vision – ECCV 2020*, pages 386–402, Cham. Springer International Publishing. 2.3.3, 3.2.2

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference*

on artificial intelligence and statistics, pages 201–208. JMLR Workshop and Conference Proceedings. 1.1

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Artificial Intelligence and Statistics*, pages 153–160. PMLR. 5.5

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. 2020. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR. 3.5

Sebastian Farquhar and Yarin Gal. 2018. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*. 1, 2.1, 5.1

William Fedus, Jeff Dean, and Barret Zoph. 2022a. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*. 1.1

William Fedus, Barret Zoph, and Noam Shazeer. 2022b. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270. 1.1

Jared Fernandez, Saujas Vaduguru, Sanket Vaibhav Mehta, Yonatan Bisk, and Emma Strubell. 2023. Adapting to gradual distribution shifts with continual weight averaging. In *HLD 2023: 1st Workshop on High-dimensional Learning Dynamics, 40th International Conference on Machine Learning (ICML), 2023*. 9.2

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org. 3.5, 8.4

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. 2019. Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1920–1930. PMLR. 3.5

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*. 5.1, 5.2, 7.5, 8.3.2

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR. 8.4

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135. 1, 7.1

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. 2023. Datacomp: In search of the next generation of multimodal datasets. *arXiv preprint arXiv:2304.14108*. 8.4

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189. 6.3.1

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of

diverse text for language modeling. *arXiv preprint arXiv:2101.00027*. 8.2

Liang Ge, Jing Gao, Hung Ngo, Kang Li, and Aidong Zhang. 2014. On handling negative transfer and imbalanced distributions in multiple source transfer learning. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 7(4):254–271. 6.5.4

Jonas Geiping and Tom Goldstein. 2023. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR. 8.2

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*. 9.1

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings. 1.1, 5.5, 5.5.1

Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. 2018. pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition. 5.4.1

Yuan Gong, Yu-An Chung, and James Glass. 2021. AST: Audio Spectrogram Transformer. In *Proc. Interspeech 2021*, pages 571–575. 1.1

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*. 1.1

Stephen Grossberg. 1987. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63. 1.1

Denis Gudovskiy, Luca Rigazio, Shun Ishizaka, Kazuki Kozuka, and Sotaro Tsukizawa. 2021. Autodo: Robust autoaugment for biased data with label noise via scalable probabilistic implicit differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16601–16610. 8.4

Chengcheng Guo, Bo Zhao, and Yanbing Bai. 2022. Deepcore: A comprehensive library for coreset selection in deep learning. In *Database and Expert Systems Applications: 33rd International Conference, DEXA 2022, Vienna, Austria, August 22–24, 2022, Proceedings, Part I*, pages 181–195. Springer. (document), 8.3

Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. 2020. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33:1023–1035. 3.5, 4.5

Gunshi Gupta, Karmesh Yadav, and Liam Paull. 2020. Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598. 3.5

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*. (document), 8.4.3, 8.1

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. REALM:

Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR. 7.5, 9.2

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4134–4143. 1.1, 3.1, 3.5, 5.5

K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. 1

Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927. 1.1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. 1.1, 5.5

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 1.1, 3.3, A.1

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751. 1.1

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012a. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2. 1.1

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012b. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. 1.1

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural computation*, 9(1):1–42. 2.4.4

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. 6.3.3

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. 1.1, 3.5

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. 4.3.1

Wenpeng Hu, Qi Qin, Mengyu Wang, Jinwen Ma, and Bing Liu. 2021. Continual learning by

using information of each class holistically. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7797–7805. 3.1

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR. 1.1, 5.5

Aman Hussain, Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2021. Towards a robust experimental framework and benchmark for lifelong language learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 3.2.3

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR. 1.1

Gautier Izacard and Édouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880. 7.5

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87. 4.1

Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost Storkey. 2019. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*. 5.4, 5.4.1

Khurram Javed and Martha White. 2019. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32. 3.5

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Transactions of the Association for Computational Linguistics*, 8:423–438. 7.5, 8.2.1, 9.3

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*. 6.5

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. 2.3.2, 4.4.3

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. 1.1, 3.1, 3.4, 3.4.3, 3.4.3, 3.5, A.1

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*. 6.3.3, 9.2

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from

science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*. 13

Najoung Kim, Song Feng, Chulaka Gunasekara, and Luis Lastras. 2020. Implicit discourse relation classification: We need to talk about evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5404–5414. 9, 14

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 1.1, 4.3.1, 6.5, 8.2

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526. 2.4.1, 2.4.1, 3.1, 3.2.3, 3.5, 4.5, 6.3.1, 7.5

A. Krizhevsky and G. Hinton. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*. 2.3.3, 2.3.3

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466. 7.1, 7.2.2

Nathan Lambert, Louis Castricato, Leandro von Werra, and Alex Havrilla. 2022. Illustrating reinforcement learning from human feedback (rlhf). *Hugging Face Blog*. 9.1

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363. 1, 9.2

Yann LeCun. 1998. The mnist database of handwritten digits. 2.3.3

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444. 1.1, 5.5

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*. 1

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. 4.3.1, 4.4.4

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474. 9.2

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. In *First Workshop on Interpolation Regularizers and Beyond at NeurIPS 2022*. 9.2

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947. 3.5

Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, D'Autume Cyprien De Masson, Tim Scholtes, Manzil Zaheer, Susannah Young, et al. 2022. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. In *International Conference on Machine Learning*, pages 13604–13622. PMLR. 4.1

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019a. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*. 8.4

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 1.1, 3.1, 3.3.3, 3.5, 8.1

Zeyu Liu, Yizhong Wang, Jungo Kasai, Hannaneh Hajishirzi, and Noah A Smith. 2021. Probing across time: What does roberta know and when? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 820–842. 1.1, 8.1, 8.2.1

Geoffrey R Loftus. 1985. Evaluating forgetting curves. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(2):397. 8.2.1

Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. 2020. Rehearsal-free continual learning over small non-iid batches. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 989–998. IEEE. 3.5

David Lopez-Paz et al. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476. 2.2, 2.4.2, 3.5, 4.5, 6.1, 6.3.2, 7.2.3, 7.3

Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2023. UNIFIED-IO: A unified model for vision, language, and multi-modal tasks. In *The Eleventh International Conference on Learning Representations*. 1

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82. 3.5

Arun Mallya and Svetlana Lazebnik. 2018. 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. 3.5

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. 4.1, 4.4.4

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32. 2.3.1, 2.4.3, 3.2.2, 3.5, 4.2.1, 4.2.2, 4.2.3, 4.3.2, 4.5, 6.1, 6.2.2, 6.2.3, 6.4, 7.5

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919. 7.6

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural lan-

guage decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*. 6.2.2

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419. 1.1, 6.1, 6.3.2, 6.4

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier. 1, 7.1

James L McGaugh. 2000. Memory–a century of consolidation. *Science*, 287(5451):248–251. 1, 1.1, 6.3.2

Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. 2023a. DSI++: Updating transformer memory with new documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1.2, 9, 9.2

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. 2023b. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50. 1.2, 1.2, 4.5, 7.5, 8.3.2, 9, 9.3

Sanket Vaibhav Mehta, Jinfeng Rao, Yi Tay, Mihir Kale, Ankur Parikh, and Emma Strubell. 2022. Improving compositional generalization with self-training for data-to-text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4205–4219. 1.2, 7.5, 7.6

Sanket Vaibhav Mehta, Zirui Wang, Barnabás Póczos, and Jaime G Carbonell. 2020. Efficient meta lifelong-learning with limited memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 535–548. (document), 1.2, 3.5, 3.5, 4.2.3, 4.1, 4.3.2, 4.5, 7.5, 8.4, 9, 9.2

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35. 7.5, 9.3

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. 2022. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*. 9.1

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. 2013. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. 1, 1.1

Luke Metz, James Harrison, C Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, et al. 2022. Velo: Training versatile learned optimizers by scaling up. *arXiv preprint arXiv:2211.09760*. 8.4

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. 2019. Understanding and correcting pathologies in the training of learned optimizers. In

International Conference on Machine Learning, pages 4556–4565. PMLR. 8.4

David Meunier, Renaud Lambiotte, Alex Fornito, Karen Ersche, and Edward T Bullmore. 2009. Hierarchical modularity in human brain functional networks. *Frontiers in neuroinformatics*, 3:37. 1.1, 4.1

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Nonparametric masked language modeling. *arXiv preprint arXiv:2212.01349*. 9.2

Seyed Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. 2020a. Dropout as an implicit gating mechanism for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 232–233. 1.1

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. 2021. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*. 2.4.4, 3.2.3, 3.5, 3.6, 5.3

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. 2020b. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320. 1.1, 2.4.4, 2.4.4, 2.4.4, 3.1, 3.2.3, 3.4, 3.5, 5.1, 5.3, 7.5

Dmytro Mishkin and Jiri Matas. 2015. All you need is a good init. *arXiv preprint arXiv:1511.06422*. 1.1, 5.5

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*. 7.5, 9.3

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115. 1, 1.1

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics. 11

Clara Na, Sanket Vaibhav Mehta, and Emma Strubell. 2022. Train flat, then compress: Sharpness-aware minimization learns more compressible models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4909–4936. 5.6

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*. 9.1

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2.3.3

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in

transfer learning? *Advances in neural information processing systems*, 33:512–523. 1.1, 3.1, 3.5, 5.5

Helen Ngo, Cooper Raterink, João GM Araújo, Ivan Zhang, Carol Chen, Adrien Morisot, and Nicholas Frosst. 2021. Mitigating harm in language models with conditional-likelihood filtration. *arXiv preprint arXiv:2108.07790*. 9.1

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016*. 7.1, 7.2.2

Randall C O'Reilly and Kenneth A Norman. 2002. Hippocampal and neocortical contributions to memory: Advances in the complementary learning systems framework. *Trends in cognitive sciences*, 6(12):505–510. 1.1

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744. 1, 9.1

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359. 1, 2, 3.5

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71. 1

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. 6.5

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607. 8.4, 8.4.2

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. 2017. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30. 1.1, 5.5

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237. 1.1, 3.5

Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14. 3.1

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473. 7.5

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81. 7

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pages 524–540. Springer. 3.2.3

Rashmi Prasad, Bonnie Webber, Alan Lee, and Aravind Joshi. 2019. Penn discourse treebank version 3.0. In *LDC2019T05*. Philadelphia: Linguistic Data Consortium. 9, 14

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics. 1.1

Chengwei Qin and Shafiq Joty. 2022. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*. 4.1, 4.5

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR. 8.4

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI*. 1.1, 3.5

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8). 6.3.1, 8.1

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*. 8.1

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551. 1.1, 1.1, 3.1, 3.5, 4.1, 4.3.1, 5.3, 6.2.2, 6.3.1, 7.2.1, 7.3, 7.4, 7.5, 8.1

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. 2019. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32. 8.4, 8.4.1

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. 2.3.2

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331. 9.2

Tiago Ramalho and Marta Garnelo. 2019. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*. (document), 6.4, 6.3, 6.5.3

Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. 2021. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *International Conference on Learning Representations*. 3.3.2

Roger Ratcliff. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285. 1, 1.1

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*. 9.2

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010. 3.5, 7.5

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*. 2.2, 3.5, 7.2.3

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*. 7.4

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426. 7.5

Anthony Robins. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146. 1.1

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358. 6.1

Stefan Ruping. 2001. Incremental learning with support vector machines. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 641–642. IEEE. 1

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*. 3.5, 4.1, 6.3.3

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled

version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. 3.1, 3.3, 3.3.3, A.1

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics. 15

Patrick Schramowski, Cigdem Turan, Nico Andersen, Constantin A Rothkopf, and Kristian Kersting. 2022. Large pre-trained language models contain human-like biases of what is right and wrong to do. *Nature Machine Intelligence*, 4(3):258–268. 9.3

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 8.4

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63. 1

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4535–4544. 2.4.1, 4.2.3, 6.2.3, 6.3.1, 6.5

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR. 3.5

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*. 4.1

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR. 1.1, 8.3.1

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in Neural Information Processing Systems*, 30. 3.5, 4.1, 4.5, 6.3.2, 7.5

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32. 8.4, 8.4.1, 8.4.2, 8.4.2

Daniel L Silver. 2011. Machine lifelong learning: Challenges and benefits for artificial general intelligence. In *International conference on artificial general intelligence*, pages 370–375. Springer. 1

Herbert A Simon. 1962. The architecture of complexity. *Proceedings of the American Philosoph-*

ical Society, 106(6):467–482. 1.1, 4.1

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. 2023. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11909–11919. 4.5

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642. 3

Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. 2020. Toward training recurrent neural networks for lifelong learning. *Neural computation*, 32(1):1–35. 3.5, 6.3.2

Shagun Sodhani, Mojtaba Faramarzi, Sanket Vaibhav Mehta, Pranshu Malviya, Mohamed Abdelsalam, Janarthanan Janarthanan, and Sarath Chandar. 2022. An introduction to lifelong supervised learning. *arXiv preprint arXiv:2207.04354*. (document), 2.1, 2.4, 3.2.3, 4.1, 4.5, 6.2.3, 9.1

Ray J Solomonoff et al. 1989. A system for incremental learning based on algorithmic probability. In *Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition*, pages 515–527. 1

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536. 8.4, 9.1

Pablo Sprechmann, Siddhant Jayakumar, Jack Rae, Alexander Pritzel, Adria Puigdomenech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. 2018. Memory-based parameter adaptation. In *International Conference on Learning Representations*. 2.4.3, 6.1, 6.3.2, 6.3.3, 7.5

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. 1.1, 6.5

Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3664–3674. 8

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021. 9.1

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650. 1

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. LAMAL: LAnguage modeling is all you need for lifelong language learning. In *International Conference on Learning Representations*.

(document), 3.5, 4.1, 4.5, 6.3.1, 6.3.2, 6.1, 7.5

Nadeem Ahmed Syed, Huan Liu, and Kah Kay Sung. 1999. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 317–321. 1

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems*, volume 35. 1.1, 7.1, 7.2.1, 7.2.2, 7.4, 7.4.5, 7.5, 8.3, 8.3.2, 8.5, 9.2

Sebastian Thrun. 1995. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems*, 8. 1

Sebastian Thrun and Tom M Mitchell. 1995. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46. 1

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290. 8.2.1, 8.2.1

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*. (document), 1, 6.3, 6.5.3, 8.1, 8.2, 8.3.1, 8.4, 8.4.1, 8.4.2, 8.5

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*. 1.1, 8.1, 9.1

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633. 1.1

Vladimir Vapnik. 1991. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4. 2.1, 2.1

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30. 1.1, 7.1

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*. 1, 2.1, 4.5

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*. 10

Jiaxing Wang, Yong Li, Jingwei Zhuo, Xupeng Shi, WEIZHONG ZHANG, Lixing Gong, Tong Tao, Pengzhang Liu, Yongjun Bao, and Weipeng Yan. 2023. DynaMS: Dyanmic margin selection for efficient deep learning. In *The Eleventh International Conference on Learning Representations*. (document), 8.4, 8.4.2, 8.3, 9.1

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. 2022a. S-prompts learning with pre-trained

transformers: An occam's razor for domain incremental learning. In *Advances in Neural Information Processing Systems*. 4.1, 4.5, 9.2

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. 2022b. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer. 4.5, 9.2

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022c. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149. 4.5, 9.2

Zirui Wang and Jaime Carbonell. 2018. Towards more reliable transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 794–810. Springer. 6.5.4

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641. 1

Colin Wei, Sham Kakade, and Tengyu Ma. 2020. The implicit and explicit regularization effects of dropout. In *International conference on machine learning*, pages 10181–10192. PMLR. 1.1

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data*, 3(1):9. 1, 6.3.1

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. 12

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771. 6.5

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*. 9.1

Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. 2.3.3

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. 2018. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR. 1.1, 5.5

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. *arXiv preprint arXiv:2302.03169*. 8.4

Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S Yu. 2023. Machine unlearning: A survey. *ACM Computing Surveys*, 56(1):1–36. 9.3

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*. 1, 6.3.1

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373. 9.2

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2018. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*. 3.5, 6.3.3

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org. 3.5, 4.5, 6.3.1

Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. 2021. idarts: Differentiable architecture search with stochastic implicit gradients. In *International Conference on Machine Learning*, pages 12557–12566. PMLR. 8.4

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*. 1.1, 8.1

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 649–657. 2.3.1, 2.3.1, 5, 6

Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W Ronny Huang, and Tom Goldstein. 2021. Gradinit: Learning to initialize neural networks for stable and efficient training. *Advances in Neural Information Processing Systems*, 34:16410–16422. 8.5

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*. 7.5, 9.3

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27. 8.2

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76. 1.1, 3.5

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*. 7.5

Barret Zoph. 2022. Designing effective sparse expert models. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1044–1044. 1.1