Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Guangji Bai Date

Harnessing Function Space of Machine Learning Models for Interpretability, Generalizability, and Efficiency

By

Guangji Bai
Doctor of Philosophy

Computer Science and Informatics

Liang Zhao, Ph.D.
Advisor

Carl Yang, Ph.D.
Committee Member

Joyce C. Ho, Ph.D.
Committee Member

Yue Cheng, Ph.D.
Committee Member

Accepted:

Kimberly R.J. Arriola, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Harnessing Function Space of Machine Learning Models for Interpretability, Generalizability, and Efficiency

By

Guangji Bai
B.Sc., Fudan University, Shanghai, China, 2018
M.Sc., George Washington University, D.C., 2020

Advisor: Liang Zhao, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2025

Abstract

Harnessing Function Space of Machine Learning Models for Interpretability, Generalizability, and Efficiency
By Guangji Bai

Modern deep learning has achieved impressive results across a wide range of domains, yet it remains fundamentally constrained by its reliance on parameter-space representations. In deep neural networks, multiple parameter configurations can represent the same function, leading to redundancy, limited interpretability, and poor generalization. This disconnect between parameter space and function space presents a critical bottleneck for building scalable, adaptive, and efficient AI systems — particularly as we pursue the broader goal of Artificial General Intelligence (AGI).

This dissertation proposes a unifying perspective centered on the function space of machine learning models — that is, reasoning about models based on the input-output functions they represent rather than their specific parameter values. By shifting the focus to functional behavior, we uncover new principles for enhancing interpretability, generalizability, and efficiency — three foundational pillars for robust and scalable AI.

For interpretability, we introduce Saliency-Regularized Multi-Task Learning (SRDML), which enforces structured task relationships via input gradients in function space, resulting in more coherent and explainable multi-task models. Building on this, SHARC combines saliency with associative memory replay to improve interpretability and mitigate catastrophic forgetting in continual learning.

For generalizability, we propose Drift-Aware Dynamic Neural Networks (DRAIN), which leverage temporal functional interpolation to handle evolving data distributions and enable robust domain generalization over time. We further extend the function-space perspective to multi-source domain adaptation with Prompt-Based Domain Discrimination (POND), employing prompt-tuning to disentangle invariant and domain-specific behavior across domains.

For efficiency, we develop SparseLLM, a global pruning framework that formulates pruning as a sparse functional optimization problem, enabling significant parameter reduction in large language models (LLMs) while maintaining performance. This direction is further advanced through FedSpaLLM, which incorporates sparsity-aware aggregation and layer-wise sampling in federated learning to address both model redundancy and system heterogeneity in decentralized environments.

Collectively, these contributions lay the foundation for a function-space-based framework for designing machine learning models that are interpretable, generalizable, and efficient — essential traits for the next generation of adaptive and resource-aware AI systems.

Harnessing Function Space of Machine Learning Models for Interpretability, Generalizability, and Efficiency

By

Guangji Bai
B.Sc., Fudan University, Shanghai, China, 2018
M.Sc., George Washington University, D.C., 2020

Advisor: Liang Zhao, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2025

Acknowledgments

Completing my Ph.D. has been one of the most challenging yet rewarding journeys of my life. Coming from a background in mathematics and statistics, I began this path with little research experience and only a budding interest in artificial intelligence. Over the past few years, that curiosity has grown into a deep passion, and I feel incredibly fortunate to have transformed from an academic novice into an independent researcher.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Liang Zhao. His dedication, patience, and guidance have been instrumental in every step of my growth. From helping me navigate my very first research project to encouraging me to think independently and critically, Prof. Zhao has played a pivotal role in shaping the researcher I am today.

I am also sincerely grateful to all my collaborators and peers throughout my Ph.D. journey — labmates, internship mentors, and external collaborators. In particular, I want to thank all my labmates, including but not limited to Chen Ling, Yifei Zhang, Bo Pan, Zheng Zhang, Shiyu Wang, etc. Their support, insights, and friendship — both academically and personally — have meant a great deal to me. I also want to thank Dr. Yijiang Li and Dr. Kibaek Kim, with whom I collaborated during my internship at Argonne National Laboratory. Our work on large language models was a truly memorable and formative experience.

I extend my sincere thanks to my thesis committee members: Dr. Yue Cheng, with whom I've had the pleasure of collaborating for five years. His expertise in machine learning systems has enriched my work immensely. Dr. Joyce Ho, one of the kindest and most supportive faculty members I've met at Emory, whose encouragement has helped me through countless moments of uncertainty. Dr. Carl Yang, who was among the first faculty I worked with at Emory. I've always admired his sharp thinking and energy, and it has been a privilege to learn from him.

Finally, I want to thank my girlfriend, Mengdan Zhu — the most precious gift of my Ph.D. years. Your love, support, and companionship have made this journey unforgettable. Thank you for being there through every day and every milestone.

To my parents, thank you for your unconditional love, encouragement, and belief in me. Your quiet strength and lifelong support have been the foundation of everything I've accomplished.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Recent years have seen tremendous breakthroughs in artificial intelligence (AI), particularly in the form of deep learning. From generating realistic images and human-like text to enabling autonomous driving and decision-making, deep neural networks (DNNs) have delivered remarkable results across a wide range of domains. These successes have sparked renewed interest in the long-standing vision of Artificial General Intelligence (AGI) — the development of machines that can flexibly reason, learn, and adapt across diverse tasks and environments, much like humans do.

Despite these advances, modern deep learning systems still fall short of the generalization, interpretability, and efficiency required for AGI. A fundamental reason lies in how models are currently designed, optimized, and analyzed — specifically, their heavy reliance on *parameter space*.

From Parameters to Functions. At its core, supervised deep learning trains a parameterized model $f_\theta \colon \mathcal{X} \to \mathcal{Y}$ to approximate an unknown target function g. The model learns from data to map inputs to outputs, such as translating a sentence, classifying an image, or predicting an action. But while our goal is to recover a *function*, our entire training and evaluation pipeline focuses on the *parameters* θ.

This creates a subtle but significant issue: deep models are often *overparameter-*

ized, meaning that many different parameter configurations can produce the same or similar functions. The result is a many-to-one mapping from parameter space to function space, introducing redundancy and ambiguity that affect generalization, optimization, and understanding.

One example of this redundancy arises in the self-attention mechanism [144], a foundational component in modern transformer models. The output of self-attention is computed as:

$$y = \sigma(QK^\top)V, \tag{1.1}$$

where $Q \in \mathbb{R}^{n\times d}$ is the query matrix, $K \in \mathbb{R}^{n\times d}$ is the key matrix, $V \in \mathbb{R}^{n\times d}$ is the value matrix, and σ denotes the softmax operation applied row-wise. Consider a simple case with two queries and three keys/values:

$$Q = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}, \quad K = \begin{pmatrix} k_1 & k_2 & k_3 \end{pmatrix}, \quad V = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

The attention output is computed as:

$$y = \sigma\left(\begin{pmatrix} q_1k_1 & q_1k_2 & q_1k_3 \\ q_2k_1 & q_2k_2 & q_2k_3 \end{pmatrix}\right)\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

Now, if we permute the rows of both K and V, say swapping $k_1 \leftrightarrow k_3$ and $v_1 \leftrightarrow v_3$, we obtain:

$$K' = \begin{pmatrix} k_3 & k_2 & k_1 \end{pmatrix}, \quad V' = \begin{pmatrix} v_3 \\ v_2 \\ v_1 \end{pmatrix}.$$

Then the new attention output becomes:

$$y' = \sigma\left(\begin{pmatrix} q_1k_3 & q_1k_2 & q_1k_1 \\ q_2k_3 & q_2k_2 & q_2k_1 \end{pmatrix}\right)\begin{pmatrix} v_3 \\ v_2 \\ v_1 \end{pmatrix}.$$

It is straightforward to verify that $y' = y$, even though the parameters K and V have changed. This shows that the self-attention function is invariant to such permutations — i.e., different parameter configurations yield the same output function.

This example illustrates the concept of *permutation symmetry* in attention layers, revealing a fundamental redundancy in the parameterization. Such invariances make it difficult to reason about model behavior purely from parameter space, reinforcing the need to operate in function space.

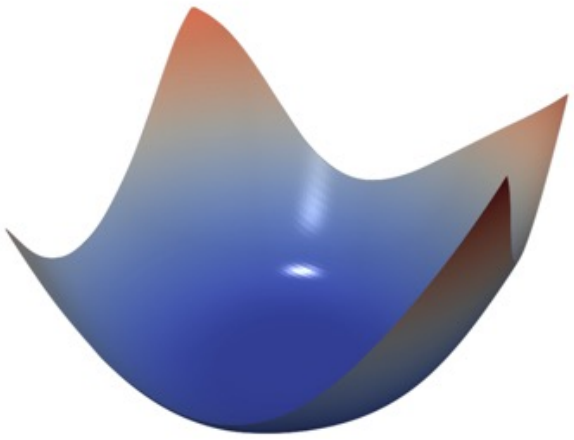


Figure 1.1: Loss surface of a 121-layer DenseNet on CIFAR-10. Wide flat minima suggest many parameter configurations represent similar functions. (Adapted from [72])

Actually, this gap is not only theoretical, but also visual. Empirical studies of neural loss landscapes (e.g., Figure 1.1) show that overparameterized models often converge to large flat regions in the loss surface — effectively wide basins of equivalent solutions. These observations further underscore the *disconnect between parameters and actual functional behavior*.

The Function-Space Perspective. This gap motivates a shift in perspective: instead of reasoning about models in parameter space, we advocate for directly studying and optimizing in *function space* — i.e., the space of mappings $f : \mathcal{X} \to \mathcal{Y}$.

By operating in function space, we can:

- Capture true behavioral similarity between models, independent of redundant parameterizations.
- Regularize learning based on how models behave with inputs, rather than how their weights are configured.
- Build more generalizable and efficient representations by focusing on what functions do, not how they are encoded.

In this dissertation, I develop a function-space-guided learning framework to tackle three core challenges that hinder progress toward AGI:

- **Interpretability:** Understanding how models make predictions and ensuring their reasoning is transparent and meaningful.
- **Generalizability:** Enabling models to adapt to distribution shifts, unseen environments, and evolving domains.
- **Efficiency:** Reducing the computational and memory burden of large-scale models without sacrificing performance.

1.1 Research Issues

This section presents three core research areas that address critical challenges in Functional-Space Guided Learning: alignment in multi-task learning, generalization across domains and temporal dynamics, and optimization for efficiency in large-scale systems. Each area is motivated by a unique perspective within the functional-space

framework and is explored through specific methodological advancements. By tackling these areas, this research lays the foundation for achieving scalable, efficient, and generalizable AI systems.

1.1.1 Harnessing Function Space of Machine Learning Models for Interpretability via Saliency Methods

Saliency-Based Regularization for Functional-Space Knowledge Transfer leverages saliency as a guiding principle to address challenges in multi-task learning (MTL) and continual learning (CL). The *Saliency-Regularized Deep Multi-task Learning* (SRDML) framework redefines task relationships by utilizing saliency maps—gradients of predictive functions—to regularize deep nonlinear functions across tasks, overcoming the limitations of traditional parameter-based approaches. By aligning tasks in saliency space, SRDML enhances interpretability and task relation learning while providing theoretical guarantees for reduced generalization error. Complementing this, *Saliency-Augmented Memory Completion* (SAMC) tackles memory inefficiency and catastrophic forgetting in CL by storing and recovering task-relevant knowledge through saliency-guided abstractions rather than raw data. Inspired by memory pattern completion in neuroscience, SAMC efficiently abstracts learning episodes using saliency maps and reconstructs them via image inpainting, significantly improving memory usage and transparency. Together, these frameworks demonstrate the potential of saliency-based methods to enhance functional-space knowledge transfer, addressing alignment, efficiency, and interpretability challenges in modern AI.

1.1.2 Harnessing Function Space of Machine Learning Models for Generalizability via Saliency Methods via Dynamic Graph Generation

The *Dynamic Regularized Adaptation with Inference Networks* (DRAIN) framework introduces a novel approach for temporal domain generalization (DG) by capturing and predicting the dynamics of both data distributions and model parameters over time. Unlike traditional DG methods that assume fixed domain boundaries or treat time as a simple feature, DRAIN formulates the problem through a Bayesian perspective, jointly modeling the relationship between temporal data shifts and the corresponding model adjustments. At its core, DRAIN represents the model as a dynamic graph, where neurons and parameters form graph-structured entities whose evolution across time is encoded and decoded using recurrent graph generation techniques. This allows DRAIN to holistically track and adapt to data distribution drift, making the model fully time-sensitive and capable of extrapolating its functionality to unseen future domains without requiring future data. Furthermore, DRAIN employs a sequential modeling mechanism to learn and leverage temporal patterns in data and model dynamics, enabling robust prediction of model states in evolving environments. With theoretical guarantees on uncertainty quantification and generalization error, DRAIN addresses key challenges in expressiveness, temporal adaptability, and theoretical rigor, as demonstrated by its superior performance across various temporal DG scenarios.

PrOmpt-based domaiN Discrimination (POND) introduces a novel framework for multi-source time series domain adaptation, leveraging prompt tuning to enhance generalizability across domains. Traditional domain adaptation methods often overlook domain-specific information, focusing instead on learning domain-invariant representations, which fail to capture valuable domain-specific trends and temporal pat-

terns critical for effective adaptation. POND addresses these limitations by extending prompt tuning to time series analysis, enabling the learning of flexible, domain-specific prompts that adapt dynamically to evolving data distributions. To achieve this, POND introduces a neural network-parameterized conditional module that generates prompts tailored to each source domain, ensuring the model can capture both global and local domain-specific information. Furthermore, it incorporates fidelity and distinction criteria to evaluate and optimize the quality of learned prompts, ensuring that they maximize mutual information with labels while minimizing redundancy across domains. By integrating these features into a robust architecture enhanced by the Mixture of Experts (MoE) technique, POND achieves superior performance in selecting the most suitable source domain for adaptation.

1.1.3 Harnessing Function Space of Machine Learning Models for Better Efficiency via Global Pruning

SparseLLM introduces a novel framework for globally optimizing the sparsity of large language models (LLMs) to enhance computational and memory efficiency during inference while maintaining strong performance. Unlike conventional global pruning methods, which are impractical for billion-scale LLMs due to high memory demands, and local pruning methods, which often yield suboptimal solutions by focusing only on layer-wise sparsity, SparseLLM addresses these challenges by decomposing the global pruning objective into manageable subproblems. Specifically, SparseLLM reformulates the global pruning task using auxiliary variables, enabling the coordination of subproblems that correspond to individual layers or modules of the LLM. An alternating optimization algorithm is then employed to solve these subproblems efficiently, ensuring computational feasibility and global optimality through closed-form solutions. SparseLLM consistently outperforms state-of-the-art local pruning techniques such as SparseGPT and Wanda, particularly in high-sparsity regimes,

achieving significant improvements in model perplexity at high sparsity levels. Moreover, its adaptable framework can seamlessly integrate with existing local pruning solvers, enhancing their performance with minimal additional computational overhead. By making global pruning feasible for extremely large models, SparseLLM sets a new standard for resource-efficient LLM inference and serves as a versatile tool for advancing model compression research.

Building on this foundation, FedSpaLLM extends the SparseLLM framework to the federated learning setting, where resource constraints and system heterogeneity present additional challenges. FedSpaLLM enables decentralized global pruning by allowing each client to independently prune its local model based on private data and specified sparsity targets. A novel ℓ_0-norm-based aggregation function is then used to unify the pruned models, ensuring that the global model maintains the desired sparsity while accommodating heterogeneous pruning masks across clients. To further reduce communication and computation costs, FedSpaLLM introduces a *layer sampling strategy* that randomly selects a subset of layers for each client to update in each round. Theoretical analysis guarantees both the global sparsity constraint and unbiasedness of the aggregation process. By bridging the gap between centralized global pruning and federated learning, FedSpaLLM brings the efficiency benefits of SparseLLM to decentralized environments, making it suitable for edge deployment and privacy-sensitive applications.

Together, SparseLLM and FedSpaLLM establish a function-space-driven paradigm for scalable, efficient LLM pruning — both in centralized and federated contexts — setting a new benchmark for resource-aware AI deployment.

1.2 Contribution

The major proposed research contributions that have been addressed up to now can be stated as follows:

Saliency-Regularized Deep Multi-Task Learning (SRDML): This paper makes the following key contributions:

- We propose a new Saliency-Regularized Deep Multitask Learning (SRDML) framework to address the challenges in multitask learning.
- We reconsider the feature weights in traditional linear multitask learning as the input gradient and generalize the feature learning into the non-linear scenario using the concept of saliency detection.
- We reformulate the task relation problem as the similarity among saliency regions across tasks to regularize and infer task relationships.
- We provide theoretical analyses to:
  - Demonstrate the equivalency of our proposed framework with traditional methods.
  - Show how the proposed regularization reduces generalization error.
- We validate the effectiveness and efficiency of our model on synthetic data and multiple large-scale real-world datasets, outperforming various baseline methods.

Saliency-Augmented Memory Completion for Continual Learning (SAMC): This paper makes the following key contributions:

- We propose a novel Saliency-Regularized Deep Multitask Learning (SRDML) framework to address critical challenges in multitask learning.

- We reinterpret feature weights in traditional linear multitask learning as input gradients and extend this concept to non-linear settings by leveraging saliency detection.
- We recast the task relation problem as the similarity among saliency regions across tasks, enabling a new method to regularize and infer task relationships.
- We provide rigorous theoretical analyses to:
  - Establish the equivalency of the proposed framework with traditional approaches.
  - Demonstrate how the proposed regularization effectively reduces generalization error.
- We validate our model's effectiveness and efficiency through experiments on synthetic data and multiple large-scale real-world datasets, achieving superior performance compared to various baselines.

Temporal Domain Generalization with Drift-Aware Dynamic Neural Network (DRAIN): This paper makes the following key contributions:

- We develop a novel and adaptive temporal domain generalization framework that can be trained in an end-to-end manner.
- We introduce an innovative approach that models the system as a dynamic graph and utilizes graph generation techniques to create a fully time-sensitive model.
- We propose using a sequential model to learn temporal drift adaptively and leverage the learned sequential patterns to predict model states in future domains.
- We provide theoretical analyses, including:

  - Uncertainty quantification.
  - Generalization error of the proposed method.

- We demonstrate the efficacy and superiority of our model through extensive experiments.

Towards Global Pruning of Pre-trained Language Models (SparseLLM): This paper makes the following key contributions:

- We propose SparseLLM, a global pruning framework designed to achieve global sparsity in large language models (LLMs) with low memory consumption by decomposing the global pruning objective into manageable subproblems.
- We reformulate LLMs as a composite function and introduce auxiliary variables to facilitate the decomposition and coordination of subproblems, enabling efficient global pruning while maintaining dependencies across model layers.
- We develop an alternating optimization algorithm that efficiently solves the subproblems with closed-form solutions, achieving computational resource efficiency and global optimality.
- We demonstrate that SparseLLM significantly improves the performance of local pruning methods, especially in high-sparsity regimes (sparsity ¿ 60%), reducing perplexity by up to 80% compared to state-of-the-art methods.
- We show that SparseLLM is adaptable and can enhance existing local pruning solvers, such as SparseGPT and Wanda, with minimal additional computational overhead, making it a versatile and practical tool for pruning LLMs.

Multi-Source Time Series Domain Adaptation with Information-Aware Prompt Tuning (POND): This paper makes the following key contributions:

- We propose a flexible prompt generator that extends prompt tuning to time series analysis by incorporating a conditional module to capture evolving domain-specific information, with theoretical analysis demonstrating its superiority.

- We introduce two criteria, fidelity and distinction, to select high-quality prompts, supported by theoretical guarantees on maintaining fidelity and introducing new information.

- We develop an efficient meta-learning-based optimization algorithm and leverage the Mixture of Experts (MoE) technique to enhance the robustness of the proposed POND model.

- We validate the effectiveness of the POND model through comprehensive experiments on multiple benchmark datasets, achieving up to a 66% improvement in F1-score over state-of-the-art methods.

1.3 Thesis Organization

The remainder of this dissertation is organized as follows:

Chapter 2 introduces methodologies for improving interpretability by leveraging saliency-based regularization in function space. Specifically, this chapter presents two frameworks: **SRDML**, which enforces task structure via input gradients, and **SAMC**, which incorporates saliency into associative memory replay to mitigate forgetting in continual learning. Both theoretical foundations and empirical results are discussed.

Chapter 3 explores generalization under distribution shift through functional-space modeling. It introduces **DRAIN**, a framework that enables temporal domain generalization via functional interpolation over time, and **POND**, which leverages prompt-tuning to disentangle invariant and domain-specific behaviors in multi-source

settings. The chapter provides theoretical insights into uncertainty and generalization error, supported by extensive experimental validation.

Chapter 4 focuses on enhancing efficiency through function-space-guided global pruning. It presents **SparseLLM**, a scalable global pruning framework for large language models based on structured sparse optimization, and **FedSpaLLM**, its federated learning extension that incorporates sparsity-aware aggregation and layer sampling to address both resource and system heterogeneity. Experimental results demonstrate the effectiveness of both methods in reducing model redundancy while preserving performance.

Finally, Chapter 5 concludes the dissertation by summarizing key contributions across the three pillars of interpretability, generalizability, and efficiency. It also discusses the broader implications of function-space reasoning and outlines several promising directions for future research toward scalable and adaptable AI systems.

Chapter 2

Harnessing Function Space of Machine Learning Models for Better Interpretability via Saliency-based Methods

Functional space, as an abstract representation of how models encode knowledge and relationships, has emerged as a powerful concept for designing algorithms that enhance generalizability, interpretability, and efficiency. By leveraging insights from functional space, we can design novel methodologies that not only optimize performance but also reveal meaningful patterns in the underlying data and tasks. In this chapter, we explore two key approaches that deploy saliency-based techniques to achieve functional-space-guided learning: SRDML and SAMC.

The first approach, SRDML (Saliency Regularized Deep Multi-task Learning), focuses on multi-task learning (MTL), where the goal is to improve the generalization of multiple tasks by learning their relationships. SRDML addresses key challenges in MTL, such as the lack of explicit task relation modeling and the inability to adaptively

determine task-sharing structures. By modeling task relations through saliency-based measures—specifically, the similarity of input gradients—it provides a theoretically grounded framework for functional space regularization. This results in improved task relation interpretability and generalization error bounds, as demonstrated through rigorous analysis and experiments.

The second approach, SAMC (Saliency-Augmented Memory Completion), builds on principles from cognitive neuroscience to address the problem of catastrophic forgetting in continual learning (CL). SAMC introduces a saliency-guided memory system that prioritizes the storage of the most task-relevant information in a bounded episodic memory. Leveraging this saliency, SAMC employs an adaptive inpainting mechanism to "complete" memories during training, ensuring storage efficiency, generalizability, and interpretability. This methodology highlights the role of saliency in functional-space-guided learning by emphasizing the critical regions of input data necessary for robust model performance.

By focusing on saliency as a mechanism for functional-space encoding, SRDML and SAMC provide complementary strategies for addressing challenges in multi-task and continual learning, respectively. Together, they lay the groundwork for understanding how functional space can guide learning systems toward more efficient and interpretable solutions. The following sections delve deeper into these methods, their theoretical underpinnings, and their practical implications.

2.1 Task Relation Learning via Saliency (SRDML)

2.1.1 Introduction

Multi-task learning (MTL, [23]) is an important research domain based on the idea that the performance of one task can be improved using related tasks as inductive bias. While traditional shallow MTL methods can fit the models for individual

tasks and learn task relations, they do not focus on generating features from scratch and instead rely on pre-defined and explicit features [165, 140]. More recently, deep representation learning empowers MTL to go "deep" by equipping it with the capacity to generate features while fitting the tasks' predictive models. Deep MTL is usually categorized according to the ways of correlating tasks' models into two major types: *hard-parameter sharing* and *soft-parameter sharing*. Hard-parameter sharing methods [166, 85] essentially hard-code which part of neurons or layers to share and which part does not for different tasks instead of doing it adaptively. Moreover, they usually share the layers for representation learning (e.g., convolutional layers) but not those for decision-making (e.g., fully-connected layers for classification). On the other hand, soft-parameter sharing methods [32, 99] do not require to hard-code the sharing pattern but instead, build individual models for each task and "softly" regularize the relatedness among them. Hence, soft-parameter sharing has better flexibility in learning the task relation, while may not be efficient since its model parameters increase linearly with the number of tasks. Hard-parameter sharing, by contrast, is more "concise" but requires pre-defining which parts are shared or not.

Therefore, although MTL is a long-lasting research domain, it remains a highly challenging and open domain that requires significantly more effort to address challenges such as the trade-off between model flexibility and conciseness of hard- and soft-parameter sharing mentioned above. Although more recently, there have come a few attempts trying to alleviate the dilemma, such as those regularizing task relationships in task-specific layers in hard-parameter sharing to achieve knowledge transfer in unshared layers [85] and those adaptively learning which part to share or not by methods like branching [87] or Neural Architecture Search [134], the research frontiers still suffer from several critical bottlenecks, including **(1) Difficulty in regularizing deep non-linear functions of different tasks.** Adaptively learning task relation requires regularizing different tasks' predictive functions, which, how-

ever, are much harder to achieve for nonlinear-nonparametric functions since they require regularizing in the whole continuous domain of input. To work around it, existing works [85, 130] typically resort to a *reduced problem* which is to regularize the neural network parameters. Notice that this reduction deviates from the original problem and is over-restricted. For example, first, two neural networks with different permutations of latent neurons can represent the same function. Moreover, even if they have different architectures, they can still possibly represent the same function [68]. This gap deteriorates the model's generalizability and effectiveness. **(2) Lack of interpretability in joint feature generation and task relation learning.** Despite the incapability of generating features, shallow MTL enjoys good interpretability since they learn explicit task correlations via how the hand-crafted features are utilized. However, in deep MTL, the generated features do not have explicit meaning and how the black-box models relate to each other is highly obscure. Increasing the interpretability of generated features and task relations is imperative yet challenging. **(3) Difficulty in theoretical analysis.** While there are fruitful theoretical analyses on shallow MTL, such as on generalization error [11] and conditions for regularized MTL algorithms to satisfy representer theorems [4], similar analyses meet strong hurdles to be extended to deep MTL due to the difficulty in reasoning about neural networks whose feature space is given by layer-wise embeddings [154]. It is crucial to enhance the theoretical analyses on the model capacity and theoretical relation among different deep MTL models.

This paper proposes a new **S**aliency-**R**egularized **D**eep **M**ulti-task **L**earning (**SRDML**) framework to solve the challenges mentioned above. First, we reconsider the feature weights in traditional linear multitask learning as the input gradient and then generalize the feature learning into the non-linear situation by borrowing the notion of saliency detection. Second, we recast the task relation problem as the similarity among saliency regions across tasks so as to regularize and infer the task

relation. Third, to validate our hypothesis, we have given a theoretical analysis of their equivalency. Meanwhile, we also provide a theoretical analysis of how the proposed regularization helps reduce the generalization error. Finally, we demonstrate our model's effectiveness and efficiency on synthetic and multiple large-scale real-world datasets under comparison with various baselines.

2.1.2 Problem Formulation

Consider a multi-task learning problem with T tasks such that a dataset $\{\mathbf{X}, \mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_T\}$ is given with i.i.d training samples $\mathbf{X}_t = \{\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}, \cdots, \mathbf{x}_n^{(t)}\}$, $\mathbf{Y}_t = \{\mathbf{y}_1^{(t)}, \mathbf{y}_2^{(t)}, \cdots, \mathbf{y}_n^{(t)}\}$, where n is the sample size and $(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})$ is a pair of input and label such that $\mathbf{x}_i^{(t)} \in \mathcal{X}$ and $\mathbf{y}_i^{(t)} \in \mathbb{R}$, $\forall\, i = 1, 2, \cdots, n$ and $t = 1, 2, \cdots, T$.

Given a predictor g which factorizes as $g = f \circ h$, where "$\circ$" stands for functional composition. The function $h : \mathcal{X} \to \mathbb{R}^K$ is called the feature or representation extraction part and is shared for all tasks, while $f : \mathbb{R}^K \to \mathbb{R}$ is a function defined on $\mathbb{R}^K$, a predictor specialized to each task at hand. K denotes the latent representation or feature-map dimensions. We further assume that each task shares the same input feature $\mathbf{x}$, i.e., $\mathbf{x}^{(1)} = \mathbf{x}^{(1)} = \cdots = \mathbf{x}^{(T)}$, which is very commonly seen in deep MTL problems such as multi-task image classification task in the Computer Vision domain.

Our *goal* is to build a deep architecture for learning multiple tasks $\mathbf{y}_i^{(t)} = g_t(\mathbf{x}_i),\ t = 1, 2, \ldots, T$ which jointly generates semantic features and learns task relation to correlate different tasks with interpretability. This goal poses significant challenges to existing work: **1).** Directly regularizing the prediction function of different tasks is extremely hard. Existing work considered a reduced problem by regularizing the feature weights of different f_t which is over-restricted. **2).** How to learn interpretable task relations with deep/implicit features is still unclear. **3).** Theoretical analysis is rare in deep MTL due to the non-linear and non-parametric functions of h and f. To jointly solve these challenges, we reconsider the feature weights in shallow MTL as

Figure 2.1: Illustrative examples of relation between saliency and task similarity. Left: Two tasks are to detect whether the man is smiling and his mouth is open. The salient regions for two tasks are both around the mouth. Right: Two tasks are to detect the horse and person. The salient regions are close to each other, indicating the potential similarity between the tasks.

input gradient, i.e., $\partial f(x)/\partial x,\ x \in \mathbb{R}^K$, and generalize the feature learning into the deep network by considering the saliency detection methods.

2.1.3 Motivation

To achieve model conciseness and efficiency as well as task relatedness flexibility, we share the representation learning layers and learn task relationships in task-specific layers. This is based on essential neuro-inspirations: human sensory organs and retina are the same for all different tasks (meaning the convolution layers are shared). On the other hand, the working memory will leverage the long-term memory for each task, and related tasks will have related memory (i.e., model), and their relatedness can be considered as the similarities of activation patterns for different tasks, namely the similarity among the saliency maps for different tasks.

Then, the next question is how to regularize the relation among different tasks, namely, how to regularize the (dis)similarity of the predictive functions of different tasks. As mentioned above, it is problematic to directly regularize the neuron network parameters due to their gap with the actual function. For example, neural networks with different architectures or neuron permutations could represent the same function. Therefore, this motivates us to explore an innovative alternative so that we can more easily work towards the space of *functional*. Specifically, we propose to regularize first-order derivatives with respect to the input of different tasks. This new strategy

has two crucial merits: First, it is mathematically equivalent to directly regularizing the function without the gap in existing works mentioned above. Second, it also finds inspiration from the saliency map domain and comes with strong interpretability in how tasks correlate.

Key Merit 1: Regularizing task functions without theoretical gap. Specifically, Theorem 2.1.1 below tells us that enforcing multiple tasks to have similar input gradients is equivalent to encouraging those tasks themselves to be similar.

Theorem 2.1.1. *Define* $\mathcal{F} := \{f \in \boldsymbol{C}^1 : f(0) = 0\}$*, where* $\boldsymbol{C}^k$ *is the family of functions with* k^{th}*-order continuous derivatives for any non-negative integer* k*. Given* $f_1, f_2 \in \mathcal{F}$*, we have:*

$$f_1 = f_2 \quad \textbf{\textit{if and only if}} \quad f_1'(x) = f_2'(x),\ \forall x \in \mathcal{X} \tag{2.1}$$

Proof. Please refer to Appendix A for the formal proof. □

Our analysis above allows us to regularize the prediction functions of different tasks in the *functional* space instead of parameter space. The assumption over function family $\mathcal{F} := \{f \in \mathbf{C}^1 : f(0) = 0\}$ is reasonable in practice since an all-zero input x simply corresponds to a "black" picture, and for any tasks we assume a black picture contains no useful information and should be classified as the negative sample (i.e., ground-truth label should be 0).

Key Merit 2: Inspiration from saliency map and enhancement of interpretability. Evaluating task relation with derivative similarity has justification from a saliency perspective. Saliency is a derivative of the prediction score w.r.t. input features, and it denotes the semantic features that influence the prediction most. In addition, similar tasks tend to have similar saliency, while dissimilar tasks tend to have dissimilar saliency. As shown in Figure 2.1, we enforce higher-level semantic features as saliency.

Many previous works have asserted that deeper representations in a CNN capture higher-level visual constructs [14]. Furthermore, convolutional layers naturally retain spatial information which is lost in fully connected layers, so we expect the last convolutional layers to have the best compromise between high-level semantics and detailed spatial information. By following a recent work called Grad-CAM [123], we use the gradient information flowing into the last convolutional layer of the CNN to capture the saliency map to each neuron for a particular task or class of interest.

2.1.4 Proposed Method

First, we will give a formal definition of saliency. For example, in computer vision, given an input image I, a classification ConvNet f predicts I belongs to class c and produces the class score $f_c(I)$ (*abbrev.* f_c). Let A be the feature map activations of the last convolutional layer. We are curious about the rank of each pixel in A based on its importance, which is referred to as saliency. The relationship between f_c and A is highly non-linear due to the non-linearity in f. In this case, we use the first-order derivatives, i.e., $\partial f_c/\partial A$, to approximate the saliency map, which basically reflects the contributions of different pixels in A to the prediction f_c.

The objective function of SRDML is defined as follows:

$$\begin{aligned} &\min_{h,f_1,\cdots,f_T,\xi} \sum\nolimits_{t=1}^{T} \mathcal{L}_t(f_t(h(\mathbf{X})), \mathbf{Y}_t), \text{ s.t.} \\ &\forall\, i,j,\ dist(\nabla_A f_i, \nabla_A f_j) \leq \xi_{ij}\ ,\ \sum\nolimits_{1\leq i<j\leq T}\ \xi_{ij} \leq \alpha \end{aligned} \tag{2.2}$$

where i, j are task indexes with $1 \leq i < j \leq T$, $A = h(\mathbf{X})$ is the feature map activations from the last convolutional layer of h, and $\nabla_A f_t$ is the first-order derivative of function f_t with respect to A, i.e., $\partial f_t/\partial A$. The $dist(\cdot)$ function here can be any distance measure including commonly-used ones like ℓ_1, ℓ_2, etc, and any potential normalization on the input gradient can also be embeded in $dist(\cdot)$.

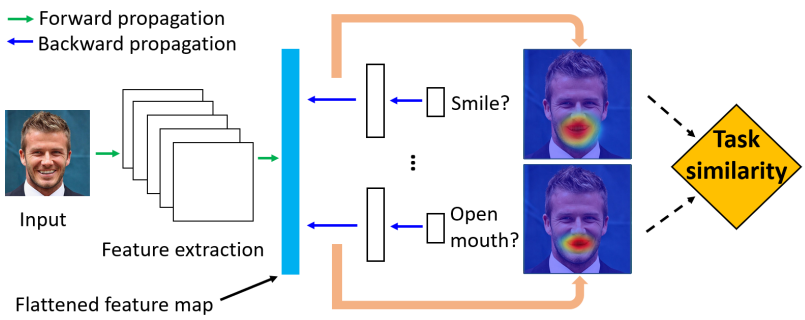


Figure 2.2: A high level overview of SRDML architecture.

To adaptively learn the task relations, we introduce $\{\xi_{ij}\}_{1\leq i<j\leq T}$, which is a set of learnable *slack* variables for each pair of tasks and α is a hyperparameter for controlling the overall level of slacking. Notice each ξ_{ij} can only take non-negative value and this is guaranteed by the inequality constraint and the non-negative norm.

Directly optimizing Eq. 2.2 could be difficult due to the constraint. By utilizing Lagrangian method, we further transform Eq. 2.2 into a regularized form as follow:

$$\begin{aligned}\min_{h,f_1,\cdots,f_T,\omega} &\sum_{t=1}^{T} \mathcal{L}_t(f_t(h(\mathbf{X})),\mathbf{Y}_t) \\ &+\lambda\cdot\sum\nolimits_{1\leq i<j\leq T}\ \omega_{ij}\cdot dist(\nabla_A f_i,\nabla_A f_j) \\ \text{s.t., } \forall\ i,j,\ \omega_{ij}\geq 0 \text{ and } &\sum\nolimits_{1\leq i<j\leq T}\ \omega_{ij}\geq\beta\end{aligned} \tag{2.3}$$

where $\{\omega_{ij}\}_{1\leq i<j\leq T}$ is a set of learnable parameters to explicitly model task relationship during the multi-task training, and λ is the regularization coefficient. Our Eq. 2.3 is motivated by the *graph regularization* [35, 36], where each node corresponds to a specific task and ω_{ij} represents the weight for the edge between task i and task j, so a graph-structure task relationship can be adaptively learned by SRDML. We rearrange the non-negative constraints over ω and apply normalization onto $\{\omega_{ij}\}_{1\leq i<j\leq T}$ to further simplify the constraints as follow:

$$\begin{aligned}\min_{h,f_1,\cdots,f_T,\omega\succ 0} &\sum_{t=1}^{T} \mathcal{L}_t(f_t(h(\mathbf{X})),\mathbf{Y}) \\ &+\lambda\cdot\sum\nolimits_{1\leq i<j\leq T}\ \frac{\omega_{ij}}{W}\cdot dist(\nabla_A f_i,\nabla_A f_j)\end{aligned} \tag{2.4}$$

where $W = \sum_{1 \leq i < j \leq T} \omega_{ij}$. Thanks to our normalization trick, the overall objective of SRDML is differentiable and can be trained in an end-to-end manner. We use standard gradient descent (e.g., Adam [57]) to solve Eq. 2.4, where we aim to learn multiple tasks and the task relationship simultaneously. Although the normalization trick introduced in Eq. 2.4 no longer guarantees that the hard constraint of the lower bound of all ω_{ij} can be strictly satisfied, our empirical results show that our normalization trick works well in practice and SRDML can capture reasonable task relationship by optimizing Eq. 2.4 with finetuned hyperparameters.

A general overview of SRDML architecture can be found in Figure 2.2. First, the input image is fed into a *shared* feature extractor, which is implemented by a sequence of convolutional layers. Right after the feature extraction process, we obtain a set of flattened feature maps (shown as the blue bar in Figure 2.2), which contains high-level semantic information with respect to the original image [123]. On top of the feature map, each task-specific head will first calculate the saliency map with respect to its own prediction. Based on the saliency map for all the tasks, the task similarity can be calculated via some distance measure. Note that our overall framework is differentiable and can be trained in an end-to-end manner.

Last, how to share the convolutional layers is *orthogonal* to the focus of our paper because our SRDML focuses on task-specific layers instead of representation learning layers. This also implies whichever is the best choice for the convolutional layer-sharing strategy can be utilized to work for our model. Our empirical results also demonstrated the reasonableness of the sharing policy that we used in this paper.

2.1.5 Theoretical Analyses

In this section, we present the theoretical analyses of our SRDML model. First, we prove that our proposed regularizer can help reduce the generalization error. Second, we formally analyze the relation between SRDML and other MTL methods. We put

all formal proofs in Appendix A due to the limited space.

Generalization Error Bound

Here we show the generalization bound of our model. Our main contribution here is we proved that **our proposed regularization term can help reduce the generalization error.**

For simpler notation, define

$$\begin{aligned}\mathcal{F}_{\epsilon(\alpha)} :=\big\{\mathbf{f} \in \mathcal{F}^T :\ \forall\, 1 \le i < j \le T,\ x \in \mathcal{X},\\ dist(\nabla_x f_i, \nabla_x f_j) \le \epsilon_{ij},\ \sum\nolimits_{1\le i<j\le T}\ \epsilon_{ij} \le \alpha\big\}\end{aligned} \tag{2.5}$$

where $\mathbf{f} = (f_1, f_2, \cdots, f_T)$ is the vectorization of each task's function, and $\{\epsilon_{ij}\}_{1\le i<j\le T}$ is a set of global slack variables. Hence, the optimization problem of Eq. 2.2 can be simplified as

$$\min_{h\in\mathcal{H},\mathbf{f}\in\mathcal{F}_{\epsilon(\alpha)}} \frac{1}{nT}\sum\nolimits_{t=1}^{T}\sum\nolimits_{i=1}^{n}\mathcal{L}_t(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)}) \tag{2.6}$$

Before introducing the theorem, we make the following standard assumptions over the loss function:

Assumption 2.1.2 ([95])**.** *The loss function $\mathcal{L}$ has values in $[0, 1]$ and has Lipschitz constant 1 in the first argument, i.e.:*

- $\mathcal{L}(y, y') \in [0, 1]$
- $\mathcal{L}(y, y') \le y,\ \forall\ y'$.

Different Lipschitz constants can be absorbed in the scaling of the predictors and different ranges than $[0, 1]$ can be handled by a simple scaling of our results.

Definition 2.1.3 (Expected risk, Empirical risk)**.** *Given any set of function $h, f_1, \cdots, f_T$, we denote the expected risk as:*

$$\mathcal{E}(h, f_1, \cdots, f_T) \coloneqq \frac{1}{T}\sum\nolimits_{t=1}^{T} \mathbb{E}_{(X,Y)\sim\mu_t}[\mathcal{L}_t(f_t(h(X)), Y)] \tag{2.7}$$

Given the data $\boldsymbol{Z} = (\boldsymbol{X}, \boldsymbol{Y})$*, the empirical risk is defined as:*

$$\hat{\mathcal{E}}(h, f_1, \cdots, f_T|\boldsymbol{Z}) \coloneqq \frac{1}{T}\sum\nolimits_{t=1}^{T} \frac{1}{n}\sum\nolimits_{i=1}^{n} \mathcal{L}_t(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)}) \tag{2.8}$$

Definition 2.1.4 (Global optimal solution, Optimized solution)**.** *Denote* $(h^*, \boldsymbol{f}^*)$ *as the global optimal solution of the expected risk:*

$$(h^*, \boldsymbol{f}^*) \coloneqq \underset{h\in\mathcal{H}, \boldsymbol{f}\in\mathcal{F}_{\epsilon(\alpha)}}{\arg\min}\ \mathcal{E}(h, f_1, \cdots, f_T) \tag{2.9}$$

and $(\hat{h}, \hat{\boldsymbol{f}})$ *as the optimized solution by minimizing the empirical risk:*

$$(\hat{h}, \hat{\boldsymbol{f}}) \coloneqq \underset{h\in\mathcal{H}, \boldsymbol{f}\in\mathcal{F}_{\epsilon(\alpha)}}{\arg\min}\ \hat{\mathcal{E}}(h, f_1, \cdots, f_T|\boldsymbol{Z}) \tag{2.10}$$

The following theorem provides theoretical guarantee of our proposed method's generalizability.

Theorem 2.1.5 (Generalization Error)**.** *Let* $\delta > 0$ *and* $\mu_1, \mu_2, \ldots, \mu_T$ *be the probability measure on* $\mathcal{X} \times \mathbb{R}$*. With probability of at least* $1 - \delta$ *in the draw of* $\mathbf{Z} = (\mathbf{X}, \mathbf{Y}) \sim \prod_{t=1}^{T} \mu_t^n$*, we have:*

$$\mathcal{E}(\hat{h}, \hat{\boldsymbol{f}}) - \mathcal{E}(h^*, \boldsymbol{f}^*) \leq c_1 L\frac{G(\mathcal{H}(\boldsymbol{X}))}{nT} + c_2 B\frac{\sqrt{\lambda_{min}^{-1}}\sup_h \|h(\boldsymbol{X})\|}{n\sqrt{nT}} + \sqrt{\frac{8\ln(4/\delta)}{nT}}, \tag{2.11}$$

where c_1*,* c_2 *are universal constants,* $G(\mathcal{H}(\boldsymbol{X}))$ *is the Gaussian average defined as* $G(\mathcal{H}(\boldsymbol{X})) = \mathbb{E}[\sup_{h\in\mathcal{H}} \sum_{kti} \gamma_{kti} h(\mathbf{x}_i^t)|\mathbf{x}_i^t]$*, where* $\{\gamma_{kti}\}$ *is i.i.d standard normal variables.* L *is the Laplacian matrix of graph with* T *vertices and edge-weights* $\{\omega_{ij}\}_{1\leq i<j\leq T}$*, and* λ_{min} *is its smallest non-zero eigenvalue.* B *is any positive value that satisfies the condition* $\sum_{i,j=1}^{T} \omega_{ij} \cdot dist^2(\nabla_A f_i, \nabla_A f_j) \leq B^2$*.*

Some remarks over Theorem 2.1.5: **1).** The first term of the bound can be interpreted as the cost of estimating the shared representation learning function $h \in \mathcal{H}$. This term is typically of order $\frac{1}{n}$. The last term contains the confidence parameter. According to [95] the constant c_1 and c_2 are pretty large, so the last term typically makes a limited contribution in the bound. **2).** The second or middle term contains the cost of estimating task-specific predictors $f \in \mathcal{F}$, and this term is typically of order $\frac{1}{\sqrt{n}}$. Here the positive constant B provides important insights into the relationship between our proposed regularizer and the error bound. **The smaller our regularization term becomes, the smaller values B could take and in turn reduce the second term in the bound.** In general, our generalization error result bounds the gap between the test error of the model trained from finite samples and that trained from infinite data, namely the theoretically optimal model/solution. In other words, Theorem 2.1.5 provides a theoretical guarantee for our performance on the actual test set.

Relation with Other MTL Frameworks

In this section, we mathematically elucidate the relation and difference between our proposed SRDML and other MTL methods, i.e., shallow MTL and deep MTL. Proof can be found in the appendix.

Natural generalization of shallow MTL. Following [165], traditional multi-task learning methods (i.e., linear model based MTL) can be generally classified into two categories: *multi-task feature learning* and *multi-task relation learning*, with objective function $\min_{W,b,\Theta} L(W,b) + \lambda/2 \cdot tr(W^\intercal \Theta^{-1} W)$ and $\min_{W,b,\Sigma} L(W,b) + \lambda/2 \cdot tr(W^\intercal \Sigma^{-1} W)$, where Θ and Σ models the covariance between different features and tasks, respectively. For any regularization-based shallow MTL defined as above, it can be formulated as a *special case* under the general framework of SRDML, with identity feature extraction function h, linear task-specific function f and the corresponding

Table 2.1: Attributes summary in CelebA and COCO.

T.id	CelebA	COCO	T.id	CelebA	COCO
1	ArchedEyebrows	person	11	PaleSkin	couch
2	BagsUnderEyes	cat	12	Sideburns	bed
3	BlackHair	dog	13	Smiling	dining table
4	BrownHair	horse	14	WavyHair	laptop
5	Chubby	car	15	WearingLipstick	tv
6	DoubleChin	truck	16	Young	cell phone
7	Goatee	bus	17		bottle
8	HeavyMakeup	motorcycle	18		cup
9	MouthSlightlyOpen	bicycle	19		bowl
10	Mustache	chair			

regularizer on the input gradients.

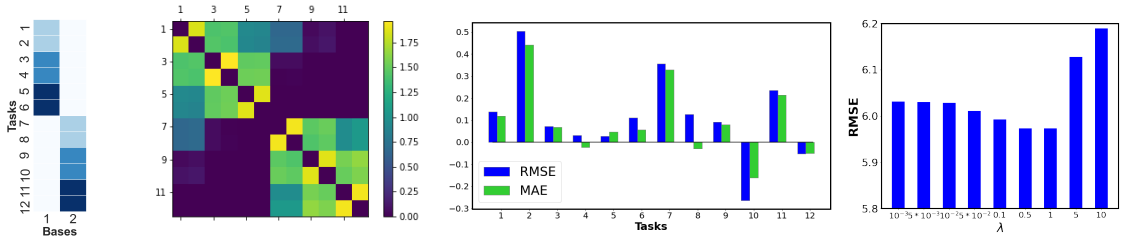
Relation with deep MTL. Define two hyperparameters: **1).** The coefficient of regularizer in SRDML λ, and **2).** the number of layers ℓ before which the model is shared cross tasks. When λ equals 0 and ℓ is greater than 1 and less than L (total number of layers), SRDML degenerates to hard-parameter sharing. On the other hand, when ℓ equals 1 and λ is greater than 0, our SRDML is equivalent to soft-parameter sharing. Hence, both hard-parameter sharing and soft-parameter sharing can be formally formulated as special cases of our proposed SRDML framework.

2.1.6 Experiments

In this section, we validate SRDML on synthetic and real-world datasets against multiple methods, on various aspects including performance, sensitivity, qualitative analysis, and ablation study. The experiments were performed on a 64-bit machine with 4-core Intel Xeon W-2123 @ 3.60GHz, 32GB memory, and NVIDIA Quadro RTX 5000.

Experimental Settings

Controlled Synthetic Dataset. We first check the validity of SRDML on a controlled regression synthetic dataset. We generate T tasks ($T = 12$) and for each task i we generate m samples ($m = 100$). The input data $\mathbf{X}_i \in \mathbb{R}^{m \times d}$ ($d = 20$) for each task i is generated from $\mathbf{X}_i \sim \mathcal{N}(\eta_i, \mathbf{I})$ with mean vector η_i and identity covariance matrix



(a) (b) Learned relation (c) Per-task performance gain. (d) sensitivity analysis.

Figure 2.3: **Experimental results on synthetic dataset. (a)**: Ground-truth of each task's feature weight. **(b)**: Task relation learned by our proposed SRDML. Tasks from different bases show strong independence (as in dark purple), tasks from the same bases show clear similarities (as in light green), and each pair of twin tasks shows very strong similarities (as in yellow). **(c)**: The performance improvement of SRDML over single task learning in RMSE (blue bar) and MAE (green bar). As shown, the SRDML model generally outperforms STL on the synthetic dataset by a large margin. **(d)**: Sensitivity analysis on regularization coefficient.

I. Next, we generate feature weight W by the following steps: **1)** Generate two base feature weights. As shown in Figure 2.3a, the first base feature weight (on the LHS column) corresponds to $\mathbf{w}_1 = (\mathbf{1}; \mathbf{0})^\intercal$ and the second base feature weight (on the RHS column) corresponds to $\mathbf{w}_2 = (\mathbf{0}; \mathbf{1})^\intercal$, where $\mathbf{1}$ and $\mathbf{0}$ each denotes a 10-dimensional all-one and all-zero vector respectively. In this way, $\mathbf{w}_1$ and $\mathbf{w}_2$ can simulate two different regions in the input X since the regions zeroed out by $\mathbf{w}$ will not be helpful in corresponding tasks. **2)** Generate task specific feature weight. Based on $\mathbf{w}_1$ and $\mathbf{w}_2$, we further consider creating different levels of saliency by multiplying the base feature weights by some magnitude parameter. Here we select 3 different magnitude parameters to create different levels of saliency for each base feature weight, and for each level of saliency, we create two tasks which are basically twin tasks. For example, in Figure 2.3a, task 1 and task 2 are twin tasks that share the same level of saliency, and the lightest blue color means they are generated by the lowest magnitude parameter. We denote each generated task-specific feature weight as w_i, $i \in \{1, 2, \cdots, T\}$. The aforementioned logistics are basically symmetric for $\mathbf{w}_1$ and $\mathbf{w}_2$. **3)** Add noise and create labels. We first inject some noise into each task's feature weight by randomly flipping the sign of the value in some positions of each w_i. The proportion of

Table 2.2: Performance (%) on real-world large-scale multi-task learning datasets. Our proposed SRMTL outperforms most comparison methods on all three datasets. Bold and underlined scores are for the best and second-best methods, respectively.

Model	CIFAR-MTL				CelebA				COCO			
	Accuracy	AUC	Precision	Recall	Accuracy	AUC	Precision	Recall	Accuracy	AUC	Precision	Recall
STL	92.65	66.20	71.32	69.83	86.83	90.96	70.53	60.39	79.23	62.91	32.23	27.04
Hard-Share	94.70	95.56	76.30	72.28	89.24	91.38	71.40	58.84	85.11	73.68	**37.43**	19.84
Lasso	91.48	86.64	68.90	24.74	76.55	66.69	37.38	36.62	78.36	64.40	28.53	28.61
L21	91.50	87.58	68.01	29.32	76.09	66.12	37.11	36.13	75.07	65.02	28.95	27.34
RMTL	92.28	85.65	61.54	28.15	75.52	66.99	37.48	36.74	76.87	65.01	29.28	28.43
MRN	94.51	96.67	79.94	76.95	89.35	91.54	71.51	64.64	85.13	75.88	32.73	25.89
MMoE	93.53	93.17	73.42	69.32	77.57	67.84	68.79	58.92	81.20	62.37	33.08	26.14
PLE	94.01	93.32	75.26	70.15	83.21	69.32	70.03	59.72	82.53	63.42	35.27	27.53
MGDA-UB	90.74	84.38	57.80	24.10	90.03	92.92	73.42	62.65	84.51	73.68	36.17	16.08
PCGrad	95.11	96.69	79.03	74.82	90.11	92.87	73.51	62.92	85.42	74.39	34.52	25.26
SRDML	95.82	96.43	81.22	75.93	90.15	92.95	73.87	64.91	85.68	76.77	35.82	28.73
SRDML (w/. PCGrad)	**96.03**	**96.72**	**82.59**	**77.01**	**90.26**	**93.01**	**73.93**	**65.30**	**85.87**	**78.38**	36.14	**30.02**

the flipped positions is controlled to guarantee the overall pattern can be well-kept. Then, we generate the label for each task by $\mathbf{Y}_i = \mathbf{X}_i \cdot w_i + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(\mathbf{0}, 0.1 \cdot \mathbf{I})$ is random normal noise.

Real-world Dataset. We evaluate the proposed method on 3 real-world benchmarks with varying number of tasks and difficulty, including multi-task version of CIFAR-10 [60] (**CIFAR-MTL**), a modified version of **CelebA** [84] and a modified version of **MS-COCO** [80]. To follow our model's assumption, all tasks are image classification ones. For CIFAR-MTL, we follow existing work [122] to create one task for each of the 10 classes in original CIFAR-10 dataset. There are 10 binary classification tasks with 2k training samples and 1k testing samples per task. CelebA has 200 thousand images of celebrity faces and each image is labeled with 40 facial attributes. We follow existing work [170] to select 16 attributes more related to facial appearance and ignore attributes around decoration such as eyeglasses and hats for our experiments. We randomly selected 30k training samples and included the whole validation and test set. For MS-COCO we select 19 types of objects and remove those with too sparse labels. We include all images that contain at least two of the 19 types of objects and randomly split them into training and testing set by half. All results are reported on the test set. For hyperparameter tuning of our method, without further specification, we applied grid search on the range of $\{10^{-3}, 5{*}10^{-3}, \cdots, 0.5, 1\}$ for the regularization coefficient.

Comparison Methods We compare SRDML with various existing methods, including two baselines, three shallow and five deep sate-of-the-art MTL methods:

- **Practical Baselines:** *1). Single Task Learning (STL)* is to train a separate predictor for each task independently. *2) Hard Parameter Sharing (Hard-Share)* considers a shared representation learning backbone (e.g., convolutional layers in CNN) and task-specific prediction head.

- **Shallow MTL Methods:** *1) Lasso* is an ℓ_1-norm regularized method that introduces sparsity into the model to reduce model complexity and feature learning, and that the parameter controlling the sparsity is shared among all tasks. *2) Joint Feature Learning (L_{21})* [34] assumes the tasks share a set of common features that represent the relatedness of multiple tasks. *3) Robust Multi-task Learning (RMTL)* [25] method assumes that some tasks are more relevant than others. It assumes that the model W can be decomposed into a low-rank structure L that captures task-relatedness and a group-sparse structure S that detects outliers.

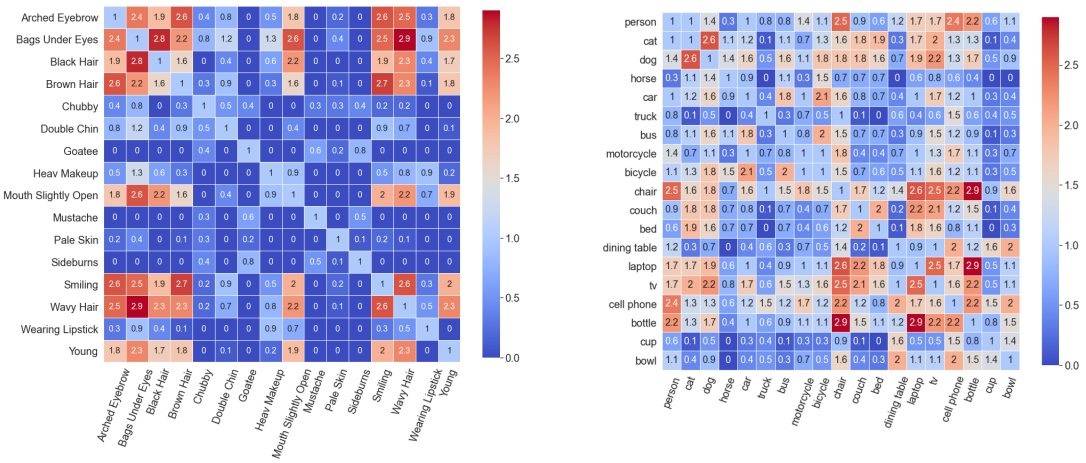
- **Deep MTL Methods:** *Multilinear Relationship Networks (MRNs)* places a tensor normal prior on task-specific layers of the deep multi-task learning model [85]. *2) Multi-gate Mixture-of-Experts (MMoE)* [90] adapt the Mixture-ofExperts (MoE) structure to multi-task learning by sharing the expert submodels across all tasks, while also having a gating network. *3) Progressive layered extraction (PLE)* [136] separates shared components and task-specific components explicitly and adopts a progressive routing mechanism to extract and separate deeper semantic knowledge gradually, improving the efficiency of joint representation learning and information routing across tasks in a general setup. *4) Multi-task Learning as Multi-Objective Optimization (MGDA-UB)* [124] considers multi-task learning from an optimization perspective by using Pareto optimality and Multiple Gradient Descent Algorithm. *5) Gradient Surgery for Multi-task Learning (PCGrad)* [163] aims to solve the

problem of gradient interference by gradient surgery, which is basically by gradient projection to make sure the gradients of different tasks have direction smaller than 90°. Since PCGrad targets gradient interference, it is only applied to the shared layers of each model to avoid the contradiction of each task's gradients. Specifically, PCGrad is applied to the shared convolutional layers.

Implementation Details. All shallow MTL methods are implemented according to the standard package MALSAR [171]. Deep MTL methods and our SRDML are built based on VGG-16 [127], which is a very popular architecture in computer vision. The convolutional layers are followed by one fully connected layer with 128 hidden neurons and one classification layer for our SRDML. Each model is trained by Adam [57]. For PCGrad, due to the fact that it is a gradient surgery method that is model-agnostic and can be applied to any deep MTL method, we report its performance by combining it with the best baseline on each real-world dataset (i.e., Hard-Share on CIFAR-MTL, MGDA-UB on CelebA, MRN on COCO). In addition, we also consider applying PCGrad to our own method SRDML, resulting in two versions of our method, namely SRDML and SRDML with PCGrad.

Experimental Results

Effectiveness on a controlled synthetic dataset. The empirical results on the regression synthetic dataset demonstrate that our model can generally outperform single-task learning and is capable of capturing the ground-truth task relations. Quantitative evaluation in Figure 2.3c shows that SRDML can outperform single-task learning in general, which can be attributed to the effective knowledge sharing between task-specific layers. In addition, the task relationship pattern (i.e., w_{ij} in Eq. 2.4) learned by SRDML as shown in Figure 2.3b is accurate and reasonable, since tasks belong to different bases are well-separated and meanwhile each pair of twin tasks shows very strong correlation (corresponds to those yellow boxes). Within each



(a) **CelebA** (b) **COCO**

Figure 2.4: Visualization of task relation learned by SRDML on real-world dataset. Zoom in for detail.

base, different pairs of twin tasks also show relatively strong relationships due to the fact that they share the same base and only differ in magnitude.

Sensitivity analysis. The sensitivity of hyperparameter λ in SRDML on synthetic dataset is shown in Figure 2.3d. As can be seen, the optimal value for λ is around 0.5 measured by RMSE. The general "U" shape is potentially reasonable because as λ goes to infinity the regularization term would dominate the overall objective function while too small λ will reduce the functionality of the regularizer and finally degenerate to single-task learning.

Effectiveness on real-world datasets.

- **CIFAR-MTL:** Table 2.2 shows the performance results of our proposed SRDML and other baselines on the CIFAR-MTL dataset. We can make the following observations from the results. **1).** Deep multi-task learning models generally outperform shallow ones by a great margin, which confirms the importance of learning the deep representation features as well as the shared policy of feature extraction part which allows knowledge transfer across tasks. **2).** Our proposed SRDML outperforms baselines in the majority of metrics and achieves comparable performance in the rest. **3).** Combining with PCGrad can further improve the performance of SRDML

due to the mitigated negative transfer in the shared convolutional layers by gradient surgery of PCGrad.

- **CelebA:** In this case, we tackle a larger and more challenging benchmark, where we tailored the dataset to contain 16 binary classification tasks with each one corresponding to a certain human's facial feature. As shown in Table 2.2, our model outperforms all comparison methods in the majority of metrics, which is attributed to the potential fact that the salient regions in some tasks are close to those in the related tasks. For example, there are two tasks to classify whether a celebrity's beard is a goatee or mustache, respectively. For both tasks, the salient regions are highly overlapped around the mouth area (as can be seen in Section "Saliency map visualization" in the appendix) so enforcing similar input gradients around the mouth area could improve the knowledge transfer and achieve better performance.

- **COCO:** To evaluate our model under various settings, we consider COCO which contains different types of objects like humans, animals, vehicles, furniture, etc, and each type of object has a varying rate of occurrence. In Table 2.2, we report the task-average classification error with lower values indicating better performance. As shown in Table 2.2, our proposed SRDML outperforms all the baselines by a great margin. This experiment also validates the effectiveness of our model when the number of tasks is relatively large and the image context is complicated. Moreover, MMoE and PLE perform generally not quite well probably due to the fact that these two approaches are designed for multi-task learning under recommender system scenario, which is not similar to that in multi-task image classification, e.g., the number of tasks in our case is much larger and hence more challenging.

Qualitative analysis. Here we demonstrate that SRDML can learn reasonable task relations on challenging real-world datasets by visualizing the task weight matrix

Table 2.3: Sensitivity analysis on regularizer coefficient when tasks are contradicting. Our regularizer coefficient can adaptively reduce to zero and avoid negative transfer.

λ	1	0.1	0.01	0.001	0
RMSE.	2.726	1.550	1.405	1.393	1.392
MAE.	2.198	1.260	1.127	1.127	1.126

(i.e., w_{ij} in Eq. 2.4). As shown in Figure 2.4, many highlighted task relations are intuitive. In CelebA, our proposed SRDML successfully learned the similarity of tasks sharing the same/similar regions around the face, like "Arched Eyebrow" and "Bags Under Eyes"; "Black Hair", "Brown Hair" and "Wavy Hair"; "Goatee", "Sideburns" and "mustache", etc. On the other hand, our model can also learn reasonable task similarities in COCO, including "cat" and "dog"; "car", "bus" and "bicycle"; "couch" and "bed", etc. We also conducted a qualitative analysis experiment on the saliency map generated by our proposed SRDML on similar or related tasks. Please refer to the appendix for the details.

Adaptive regularizer on contradicting tasks. In this section, we conducted another sensitivity analysis when all tasks compete (we generate such a synthetic dataset by following a similar procedure introduced in Section 5.1), and the results in Table 2.3 demonstrate the efficacy of our regularization term, which can adaptively decrease the task-similarity weight to zero and avoid competition.

Ablation study. In this section, we present an ablation study on the task relation learning part in the regularizer. Specifically, we remove the $\{\omega_{ij}\}_{1\leq i<j\leq T}$ in Eq. 2.3 and the coefficient for each term in the regularizer is just the hyperparameter λ. We conducted experiments on all three real-world datasets to see the difference, and the results are shown in Table 2.4.

Table 2.4: Ablation study on adaptive regularizer (Accuracy)

	CIFAR-MTL	CelebA	MS-COCO
SRDML. (**w/o** regularizer)	94.92	89.74	85.18
SRDML. (**w/.** regularizer)	**95.82**	**90.15**	**85.68**

2.2 Saliency-Augmented Memory Completion for Continual Learning

2.2.1 Introduction

An important step toward next-generation Artificial Intelligence (AI) is a promising new domain known as continual learning (CL), where neural networks learn continuously over a sequence of tasks, similar to the way humans learn [108]. Compared with traditional supervised learning, continual learning is still in its very primitive stage. Currently, the primary goal is essentially to avoid *Catastrophic Forgetting* [96] of previously learned tasks when an agent is learning new tasks. Continual learning aims to mitigate forgetting while updating the model over a stream of tasks.

To overcome this issue, researchers have proposed a number of different strategies. Among various approaches, the *replay-based methods* are arguably more effective in terms of performance and bio-inspiration [121] as a way to alleviate the catastrophic forgetting challenge and are thus becoming the preferred approach for continual learning models [118, 2]. However, the performance of these methods highly depends on the size of the episodic memory. Recent work [59] proved that to achieve optimal performance in continual learning, one has to store all previous examples in the memory, which is almost impossible in practice and counters the way how human brain works. Unlike avoiding (catastrophic) forgetting, the attempts on 'how to reasonably forget' is still a highly open question, leading to significant challenges in continual learning, including **1) Memory inefficiency.** The performance of replay-based models depends heavily on the size of the available memory in the replay buffer, which is used to retain as many previous samples as possible. While existing works typically store the entire sample in memory, we humans seldom memorize every detail of our experiences. Thus, compared to biological neural networks, some mechanisms must

still be missing in current models; **2) Insufficient generalization power.** The primary focus of existing works is to avoid catastrophic forgetting by memorizing all the details without taking into account their usefulness for learning tasks. They typically rely on episodic memory for individual tasks without sufficient chaining to make the knowledge they learn truly generalizable to all potential (historical and future unseen) tasks. In contrast, human beings significantly improve generalizability during continual learning; **3) Obscurity of the memory and its importance to learning tasks.** Human being usually has a concise and clear clue on how the relevant memory is useful for learning tasks. Such a clue could even be helpful for telling if the human has the necessary memory to be capable of a specific task at all. However, the majority of existing works in CL pay little attention to pursuing such transparency of continual learning models. For example, most works directly feed the images into their model for training without any explanation generated, which may prevent users from understanding which semantic features in the image are the most decisive ones and how the model is reasoning to make the final prediction. In addition, without an explanation generation mechanism, it is much more challenging for model diagnosing or debugging, let alone understanding how knowledge is stored and refined through the continual learning process.

To jointly address these challenges, this paper proposes **S**aliency-**A**ugmented **M**emory **C**ompletion (**SAMC**) framework for continual learning, which is inspired by the *memory pattern completion* theory in cognitive neuroscience. The memory pattern completion process guides the abstraction of learning episodes (tasks) into semantic knowledge as well as the reverse process in recovering episodes from the memorized abstracts. Specifically, in this paper, instead of memorizing all historical training samples, we memorize their interpretable abstraction in terms of the saliency maps that most determine the prediction outputs for each learning task. **Our contribution includes,** 1). We develop a novel neural-inspired continual learning

framework to handle catastrophic forgetting. 2). We propose two techniques based on saliency map and image inpainting methods for efficient memory storage and recovery. 3). We design a bilevel optimization algorithm with a theoretical guarantee to train our entire framework in an end-to-end manner. 4). We demonstrate our model's efficacy and superiority with extensive experiments.

2.2.2 Problem Formulation

Continual learning is defined as an online supervised learning problem. Following the learning protocol in [86], we consider a training set $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T\}$ consisting of T tasks, where $\mathcal{D}_t = \{(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})\}_{i=1}^{n_t}$ contains n_t input-target pairs $(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}) \in \mathcal{X} \times \mathcal{Y}$. While each learning task arrives sequentially, we make the assumption of *locally i.i.d*, i.e., $\forall\, t, (\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}) \overset{iid}{\sim} P_t$, where P_t denotes the data distribution for task t and *i.i.d* for *independent and identically distributed*.

Given such a stream of tasks, our **goal** is to train a learning agent $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by θ, which can be queried *at any time* to predict the target $\mathbf{y}$ given associated unseen input $\mathbf{x}$ and task id t. Moreover, we require that such a learning agent can only store a small amount of seen samples in an episodic memory $\mathcal{M}$ with fixed budget. Under the goal, we are interested in how to achieve *continual learning without forgetting* problem setting defined as following: Given predictor f_θ, the loss on the episodic memory of task k is defined as

$$\ell(f_\theta, \mathcal{M}_k) \coloneqq |\mathcal{M}_k|^{-1} \sum_{(\mathbf{x}_i, k, \mathbf{y}_i)} \phi(f_\theta(\mathbf{x_i}, k), \mathbf{y}_i), \tag{2.12}$$

$\forall\, k < t$, where ϕ can be *e.g.* cross-entropy or MSE. We consider *constrained optimization* to avoid the losses from increasing, which in turn allows the so called *positive backward transfer* [86]. More specifically, when observing the triplet $(\mathbf{x}, \mathbf{y}, t)$ from the

current task t, we solve the following *inequality-constrained* problem:

$$\begin{aligned} &\min_\theta \ell\big(f_\theta(\mathbf{x}, t), \mathbf{y}\big), \quad \text{s.t.,} \\ &\ell\big(f_\theta, \mathcal{M}_k\big) \leq \ell\big(f_\theta^{t-1}, \mathcal{M}_k\big), \end{aligned} \tag{2.13}$$

where f_θ^{t-1} denotes the predictor state at the end of learning task $t-1$ and $k = 1, 2, \cdots, t-1$. After the training of task t, a subset of training samples will be stored into the episodic memory, i.e., $\mathcal{M} = \mathcal{M} \cup \{(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})\}_{i=1}^{m_t}$, where m_t is the memory buffer size for the current task. During the training of task $t+1$, previously stored samples will serve as $\mathcal{M}_t$ in Eq. 2.13.

Our goal above poses significant challenges to existing work: 1). The performance of replay-based methods highly depends on the memory size. Existing work typically considered storing entire samples into the memory, which was inefficient in practice. 2). Memorizing all the details could be problematic, and how to capture the most important and generalizable knowledge through episodic memory is under-explored. 3). Interpretability and transparency over f_θ are typically ignored in existing work, which results in an obscure model with insufficient explanations on how the model is reasoning towards its predictions and how knowledge is stored and refined through the entire continual learning process.

2.2.3 Proposed Method

In this section, we introduce our proposed Saliency-augmented Memory Completion (SAMC) that addresses all the challenges mentioned earlier and thus narrows the gap between AI and human learning. We innovatively utilize Explainable AI ($\mathcal{XAI}$) methods to select the most salient regions for each image and only store the selected pixels in our episodic memory, thus achieving controllable and better memory efficiency. During the training phase, we leverage learning-based image inpainting techniques for memory completion, where each partially stored image will be restored

by the inpainting model. A bilevel optimization algorithm is designed and allows our entire framework to be trained in an end-to-end manner. Our theoretical analyses show that, as long as we properly select the salient region for each image and the inpainting model is well trained, we can simultaneously achieve better memory efficiency and positive backward transfer with a theoretical guarantee.

Saliency-augmented Episodic Memory.

In this section, we demonstrate how to leverage the saliency map generated by saliency methods to select the "informative" region of each image and how we design the episodic memory structure to store the images efficiently.

Saliency-based methods generate visual explanation via saliency map, which can be regarded as the impact of specific feature map activations on a given prediction. Given an input image $I \in \mathbb{R}^{H\times W\times C}$, a classification ConvNet f_θ predicts I belongs to class c and produces the class score $f_\theta^c(I)$ (short as f_θ^c). The saliency map $M_I \in \mathbb{R}^{H\times W}$ is generated by assigning high intensity values to the relevant image regions that contribute more to the model's prediction, i.e.,

$$M_I = \Phi(I, f_\theta, c), \tag{2.14}$$

where Φ can be any visual explainer that can generate saliency map as Eq. 2.14. In this paper, we consider Grad-CAM as our choice of Φ, due to that it can pass important "sanity checks" regarding their sensitivity to data and model parameters [1], which differentiate Grad-CAM with many other explanation methods [33].

Suppose we are training on task t with predictor state f_θ^{t-1}. Given a mini-batch $\mathcal{B}_t = \{(\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)})\}_{i=1}^{bsz}$ from the current task's data $\mathcal{D}_t$ (bsz denotes mini-batch size), we generate the saliency map M following Eq. 2.14. Intuitively speaking, pixels with higher magnitude in saliency map are more likely to be involved in the region of the target class object while those with lower magnitude are more likely to be the non-

target objects or background regions. Specifically, for any input pair $(x, y) \in \mathcal{B}_t$, the masked image x' is

$$x' = \mathbb{1}\{M_x > \mu\} = \mathbb{1}\{\Phi(x, f_\theta, y) > \mu\}, \tag{2.15}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and μ is the threshold for controlling each image's drop ratio.

We denote the mini-batch after extraction as $\mathcal{B}'_t$. Due to the sparsity in x', it is inefficient to store the entire images in episodic memory. We propose to store extracted images in the format of sparse matrix. Specifically, for each image in $\mathcal{B}'_t$, we first convert it into Coordinate Format (COO) [12], i.e., $[(\text{row}_{x'}, \text{col}_{x'}, \text{value}_{x'})] = \text{COO}(x')$, where the left-hand-side is a list of (rwo,col,value) tuples. After transforming the images into COO format, we store them into the episodic memory as $\mathcal{M}_t = \mathcal{M}_t \cup \text{COO}(\mathcal{B}'_t)$. We consider a simple way for adding samples into memory similar to [86], where the samples are added into memory following *first-in-first-out* (FIFO) and the last group of samples are stored in memory in the end.

Memory Completion with Inpainting.

As shown in Eq. 2.13, when we are training on task t we need to calculate the loss on previous samples stored in episodic memory. However, due to the missing pixels, classification models (e.g., CNN) cannot be operated on such ragged images. There are various existing works on how to retrieve images with missing pixels, and we consider the following two approaches:

Rule-based Inpainting. First approach is to handle the missing pixels by prescribed rules [15, 138]. One advantage of such methods is that they introduce neither extra model parameters nor training costs. Also, one does not need to worry about any potential forgetting of the inpainting method itself during continual learning since it

is purely based on hand-crafted rules. One naive candidate of rule-based inpainting is simply padding all missing pixels with zeros. We have included it as a baseline in this paper for comparison.

Learning-based Inpainting. An alternative is to utilize learning-based inpainting models, and several deep learning models have demonstrated their state-of-the-art performance on various image inpainting tasks [159, 111]. However, the trade-off of their excellent performance is the potential higher memory cost and computational inefficiency. Moreover, it is also challenging to ensure the model generalization for each task during training without severe forgetting.

In this work, we consider autoencoder [14] as our learning-based inpainting model since 1) autoencoder has been utilized as the backbone in many state-of-the-art deep learning-based inpainting models [159, 111], which has proven its capability of learning any useful salient features from the masked image, 2) autoencoder is rather a light-weight model which does not bring heavy burden to our framework. Specifically, the autoencoder $\mathcal{A}_\xi(x')$ can be divided into two parts: an encoder $p_{\xi_e}(z|x')$ that is parameterized by ξ_e, and a decoder $p_{\xi_d}(\tilde{x}|z)$ that is parameterized by ξ_d, where $\xi = \{\xi_e, \xi_d\}$, x' is the masked image, $\tilde{x}$ is the recovered image, and z is a compressed bottleneck (latent variable). Inpainting autoencoder aims at characterizing the conditional probability

$$\mathcal{A}_\xi(x') = p_{\xi_d}(\tilde{x}|z) \cdot p_{\xi_e}(z|x') \tag{2.16}$$

to reconstruct the image $\tilde{x}$ from its masked version x'. Moreover, $\mathcal{A}_\xi(\cdot)$ can be trained with the objective of minimizing the reconstruction loss $\arg\min_\xi ||x^* - \mathcal{A}_\xi(x')||_2^2$, where x^* is the ground truth image.

Furthermore, to maximize the number of samples into the memory $\mathcal{M}_k$, it is imperative to adopt a relatively large drop ratio μ. However, the generalization power of the model would be affected since the details of the original image can hardly be recovered in $\tilde{x}$ under a high drop rate. To ensure the model retains its

generalization power under such circumstance, we leverage a novel image inpatinting framework that incorporates both rule-based inpainting $\mathcal{R}(\cdot)$ and the autoencoder $\mathcal{A}_\xi(x')$ to recover the image from coarse to fine. Specifically, given masked image x' as defined in Eq. 2.15, we propagate it through rule-based inpainting module $\mathcal{R}(\cdot)$ to get a coarse prediction and then feed the output after rule-based inpainting into autoencoder $\mathcal{A}_\xi(\cdot)$ for further refinement, i.e., $\tilde{x} = \mathcal{A}_\xi(\mathcal{R}(x'))$. If we are training on task t, given masked images stored in episodic memory $\mathcal{M}_k, \forall\ k < t$, we convert them from COO format back to standard sparse image tensor and generate the memory with pattern completion $\tilde{\mathcal{M}}_t$ as

$$\tilde{\mathcal{M}}_k = \mathcal{A}_\xi\Big(\mathcal{R}\big(\mathrm{COO}^{-1}(\mathcal{M}_k)\big)\Big), \quad \forall\ k < t, \tag{2.17}$$

where COO^{-1} denotes the decoding process from COO format back to sparse matrix.

Bilevel Optimization.

In each incremental phase, we optimize two groups of learnable parameters: 1) the model parameter of continual learning classifier, θ, and 2) the inpainting model's parameter ξ. Directly optimizing the entire framework could be hard since θ and ξ are coupled. In this paper, we consider formulating the problem as bilevel optimization, i.e.,

$$\begin{aligned} \text{Upper:} \quad & \xi^* = \mathrm{argmin}_\xi \|\mathbf{x} - \mathcal{A}_\xi(\mathcal{R}(\mathbf{x}'))\|_2^2 \\ \text{Lower:} \quad & \text{s.t.} \quad \theta^* = \mathrm{argmin}_\theta\ \ell\big(f_\theta(\mathbf{x}, t), \mathbf{y}\big) \\ & \qquad\qquad \text{s.t. } \ell(f_\theta, \tilde{\mathcal{M}}_k) \leq \ell(f_\theta^{t-1}, \tilde{\mathcal{M}}_k), \end{aligned} \tag{2.18}$$

where $k < t$ and $\mathbf{x}' = \mathbb{1}\{\Phi(\mathbf{x}, f_{\theta^*}, \mathbf{y}) > \mu\}$. The upper-level corresponds to minimizing the image reconstruction loss on the current task, where each raw image is masked by saliency map and then inpainted by the inpainting model. The lower-level corresponds

to the standard objective as Eq. 2.13 for training continual learning classifier f_θ while the constraints are based on inpainted images $\tilde{\mathcal{M}}_k$ instead of actual images $\mathcal{M}_k$.

Optimizing θ: Given ξ, the objective function over θ becomes a constrained optimization problem. To handle the inequality constraints, we follow [86] but with a clearer theoretical explanation.

Definition 2.2.1. *Given loss function ℓ and an input triplet (x, y, t) or episodic memory after completion $\tilde{\mathcal{M}}_k$, define the loss gradient vector as*

$$g \coloneqq \partial \ell(f_\theta(\mathbf{x}, t), \mathbf{y})/\partial\theta, \quad \tilde{g}_k \coloneqq \partial \ell(f_\theta, \tilde{\mathcal{M}}_k)/\partial\theta. \tag{2.19}$$

The first lemma below proves the *sufficiency* of enforcing the constraint over the loss gradient in order to guarantee *positive backward transfer* as in Eq. 2.13.

Lemma 2.2.2. *$\forall\ k < t$, with small enough step size α,*

$$\langle g, \tilde{g}_k \rangle \geq 0 \ \Rightarrow\ \ell(f_\theta, \tilde{\mathcal{M}}_k) \leq \ell(f_\theta^{t-1}), \tilde{\mathcal{M}}_k). \tag{2.20}$$

Guaranteed by the above lemma, we approximate the inequality constraints in Eq. 2.18 by computing the angle between g and $\tilde{g}_k$, i.e., $\langle g, \tilde{g}_k \rangle \geq 0, \forall\ k < t$. Whenever the angle is greater than 90°, the proposed gradient g will be projected to the closest gradient $\tilde{g}$ in ℓ_2-norm that keeps the angle within 90°. For more details please refer to [86] due to limited space.

Optimizing ξ: Following the training procedure of bilevel optimization [38], we take a couple of gradient descent steps over the inpainting model parameter ξ when samples from new task arrive, i.e.,

$$\xi \leftarrow \xi - \beta \cdot \nabla_\xi \|\mathbf{x} - \mathcal{A}_\xi(\mathcal{R}(\mathbf{x}'))\|_2^2, \tag{2.21}$$

where β is the step size for ξ. It has been shown that *early stopping* in SGD can be

regarded as implicit regularization [117], and we adopt this technique over ξ to help mitigate the forgetting of the inpainting model. We only take a few steps of SGD over ξ that helps control any detrimental semantic drift of the autoencoder and thus alleviate potential forgetting.

The next lemma shows that the difference between continual learning model's outputs over the original image and inpainted one can be upper bounded by factors related to Grad-CAM's saliency map and the inpainting model's reconstruction error.

Lemma 2.2.3. *Given input $x \in \mathbb{R}^d$, suppose $\tilde{x}$ is the inpainted sample generated by our proposed method. Denote y_x^c as f_θ's prediction over x that belongs to class c, and $\Delta x \coloneqq \max_{1\leq i\leq d} |x_i - \tilde{x}_i|$, then:*

$$err(x, \tilde{x}) \coloneqq |y_x^c - y_{\tilde{x}}^c| \leq \mu \cdot \Delta x \cdot d^{-2}. \tag{2.22}$$

Our main theorem below guarantees that, with proper choice of the threshold μ for pixel dropping and inpainting model with bounded reconstruction error, enforcing the constraint over the memory after pattern completion $\tilde{\mathcal{M}}$ is equivalent to that over the memory with ground-truth images.

Theorem 2.2.4. *Suppose ℓ is smooth with Lipschitz constant L, and the first order derivative of ℓ, i.e., $\partial\ell/\partial\theta$ is Lipschitz continuous with constant L_θ. We have:*

$$\langle g, \tilde{g}_k \rangle > 0 \;\Rightarrow\; \langle g, g_k \rangle > 0, \quad as \;\; \tilde{x} \to x, \tag{2.23}$$

where g_k is the gradient vector on ground-truth images.

Memory Complexity Analyses.

The complexity to store samples into the memory is $\mathcal{O}(T \cdot (1 - \bar{\mu}) \cdot |\mathcal{M}_t|)$, where T is number of tasks, $\bar{\mu}$ is the average pixel drop ratio determined by μ in Eq. 2.15,

and $|\mathcal{M}_t|$ is the episodic memory size for each task. The greater the $\bar{\mu}$, the lower the memory complexity to store the same amount of samples. If we consider the learning-based inpainting method and denote the number of parameters of the inpainting model as S, the overall memory cost becomes $\mathcal{O}(T \cdot (1 - \bar{\mu}) \cdot |\mathcal{M}_t| + S)$. The cost of storing the inpainting model is often dwarfed by that of storing images since each image is a high-dimensional tensor and the cost for storing the inpainting model is constant w.r.t T.

2.2.4 Experiments

In this section, we compare our proposed method SAMC with several state-of-the-art CL methods on commonly used benchmarks. More details and additional results can be found in the appendix. Code available at `https://github.com/BaiTheBest/SAMC`

Experiment Setups.

Datasets. We perform experiments on three widely-used image classification datasets in continual learning: Split CIFAR-10, Split CIFAR-100, and Split mini-ImageNet [24]. CIFAR-10 consists of 50,000 RGB training images and 10,000 test images belonging to 10 object classes. Similar to CIFAR-10, CIFAR-100, except it has 100 classes containing 600 images each. ImageNet-50 is a smaller subset of the iLSVRC-2012 dataset containing 50 classes with 1300 training images and 50 validation images per class. For Split CIFAR-10, we consider 5 tasks where each task contains two classes. For Split CIFAR-100 and Split mini-ImageNet, we consider 20 tasks where each task includes five classes. The image size for CIFAR is 32×32 and for mini-ImageNet is 84×84.

Architectures. Following [86], we use a reduced ResNet18 with three times fewer feature maps across all layers on all three datasets. Similar to [86, 24], we train and

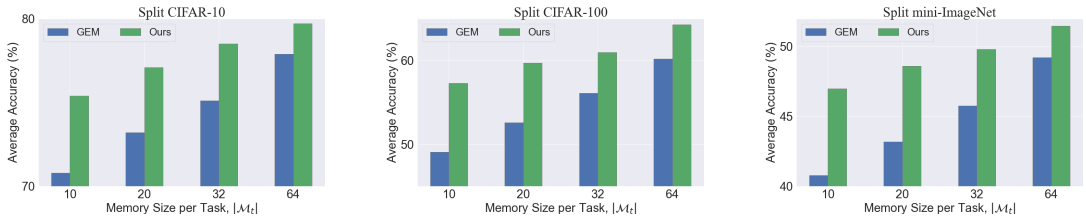


Figure 2.5: **Effects of memory size.** We compare ACC for varying memory size per task on Split CIFAR-10, Split CIFAR-100 and Split mini-ImageNet, respectively. Number of tasks for three datasets are 5, 20, and 20.

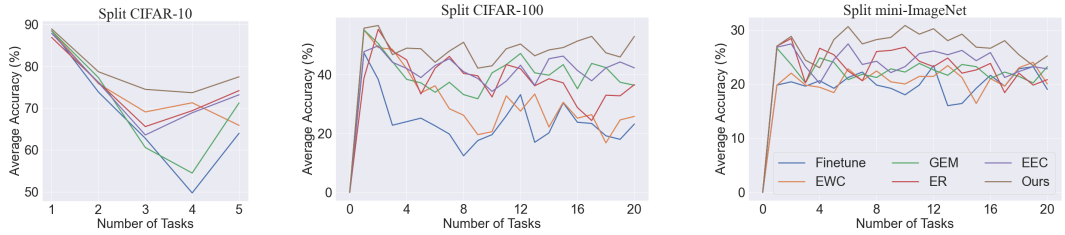


Figure 2.6: **Evolution of test accuracy at the first task, as more tasks are learned.** We compare ACC of the first task over the entire training process on Split CIFAR-10, Split CIFAR-100 and Split mini-ImageNet. When used, episodic memories contain 10 samples per task. The legend is same and shown in the right sub-figure only.

evaluate our algorithm in a 'multi-head' setting where a task id is used to select a task-specific classifier head. For the learning-based inpainting method, we consider the inpainting autoencoder with a three-layer encoder and a three-layer decoder with a convolutional structure, respectively. Please refer to the appendix for more details.

Comparison Methods. We compare our method with several groups of continual learning methods, including:

- **Practical Baselines:** *1) Finetune*, a popular baseline, naively trained on the data stream.

- **Regularization methods:** *2) EWC* [58], a well-known regularization-based method; *3) LwF* [77], another regularization-based method for CNNs.

- **Replay-based Methods:** *4) iCaRL* [118], a class-incremental learner that classifies using a nearest exemplar algorithm; *5) GEM* [86], a replay approach based on an episodic memory of parameter gradients; *6) ER* [24], a simple yet competi-

tive experience method based on reservoir sampling; *7) MER* [120], another replay approach inspired by meta-learning.

- **Pseudo-rehearsal Methods:** *8) EBM* [74], an energy-based method for continual learning; *9) EEC* [6], an autoencoder-based generative method.

Evaluation Metrics. We evaluate the classification performance using the ACC metric, which is the average test classification accuracy of all tasks. We report backward transfer, i.e., BWT [86] to measure the influence of new learning on the past knowledge. Negative BWT indicates forgetting so the bigger the better. We put detailed experimental settings in the appendix.

Experiment Results.

Classification Performance. Table 2.2 shows the overall experimental results, where the memory size per task is set to 10 for all datasets, so the total memory buffer size is 50, 200, 200 on CIFAR-10, CIFAR-100 and mini-ImageNet, respectively. On each dataset, our proposed SAMC outperforms the baselines by significant margins, and the gains in performance are especially substantial on more challenging datasets where number of tasks is large and the sample size for each task is small, e.g., CIFAR-100 and mini-ImageNet. Specifically, our model achieves $\sim$12% and $\sim$15% relative improvement ratio in accuracy, while achieving $\sim$5% and $\sim$6% lower backward transfer compared with the second best method on CIFAR-100 and mini-ImageNet, respectively. This substantial performance improvement can be attributed to better memory efficiency and generalizability provided by SAMC. EWC [58] performs relatively poor without multiple passes over the dataset and only achieves similar accuracy as finetune baseline. GEM is favored most on CIFAR-100 where it outperformed other methods by some margin. Experience replay methods like ER and MER achieved better performance on CIFAR-100. which is consistent with recent studies on tiny size memory [24]. Pseudo-rehearsal methods [74, 6], though resemble

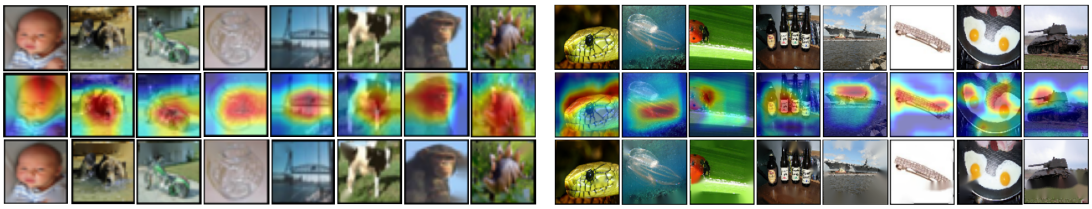


Figure 2.7: **Visualization of saliency map and inpainted images generated by SAMC. Left:** CIFAR-100. **Right:** mini-ImageNet. **First row:** Ground truth; **Second row:** Saliency map; **Third row:** Inpainted image.

to our proposed method in a way, lack interpretability and suffer from forgetting in the generative model, so achieve lower accuracy than SAMC.

Effects of Memory Size. As shown in Figure 2.5, we compare our proposed method SAMC with GEM [86] over varying memory buffer size. Both GEM [86] and SAMC benefit from larger memory size since the accuracy is an increasing function of memory size for both methods. SAMC's improvement in performance over GEM is more substantial in lower data regime, which is possibly due to that sample size is proportional to the memory size. When sample size is small model may easily overfit thus our extra samples can provide maximal benefits, while when sample size is already large enough to learn the model there is no need for extra data. Our model achieves best performance gain in handling few-shot CL which we believe is the most challenging while meaningful case in practice.

Effectiveness in Handling Forgetting. As shown in Figure 2.6, we demonstrate the evolution of first task's test accuracy throughout the entire training process. As can be seen, the proposed method SAMC (red curve) is almost on top of all the curves for the three datasets. Thanks to the memory efficiency of our algorithm and the generalization ability brought by the inpainting model, the proposed method can mitigate catastrophic forgetting to a higher level than other approaches, even with a very small memory buffer.

Memory Usage Analysis. As shown in Table 2.5, we analyze the disc space and training time required by our model on the mini-ImageNet dataset and demonstrate that *our computational bottleneck is negligible compared to our memory saving.* The

Table 2.5: Memory & computational cost analyses. Our method can achieve 60% memory saving with only 7.5% extra computational cost on mini-ImageNet.

	Disc Consupmtion (MB)	Running Time (sec)
GEM	44 (model) + 413 (images)	133
SAMC	44 + 413*$\frac{1}{3}$ (↓) + 0.5 (AE,↑)	133 + 7 (↑) + 3 (↑)
ratio	∼60% (↓)	∼7.5% (↑)

autoencoder (AE) we use is very light-weight and we apply early stopping for its finetuning. By adding quantitative analyses we found our SAMC can achieve **60%** memory saving with only **7.5%** extra computational cost. Specifically, on mini-ImgNet, SAMC requires 44 MB for ResNet18, 137 MB for images (μ=0.6) and less than 1 MB for AE, while GEM baseline requires 44 MB for ResNet and 413 MB for real images. Meanwhile, by measuring the training time for the first two tasks in a single epoch, SAMC takes 7s and 3s for computation induced by Grad-CAM and autoencoder, respectively, in addition to GEM's 130s training time, which leads to a marginal 7.5% (=(7+3)/130) computational cost. Notice that Grad-CAM itself does NOT introduce any extra parameter.

Qualitative Analysis. We visualize the saliency map and inpainted images by SAMC in Figure 2.7. Both the saliency map and inpainted images are visually reasonable. Saliency map localization is more accurate on larger images such as mini-ImageNet. The inpainting model generates high-quality images close to the ground truth though there exist occasional and small obscure regions in some cases.

2.3 Conclusion

In this chapter, we explored saliency-based methods as a cornerstone for functional-space guided learning, focusing on two key approaches: SRDML and SAMC. Both methods leverage saliency to extract meaningful representations from input data, aligning the parameter space with the functional space to achieve better generalization, interpretability, and efficiency.

SRDML demonstrated how task relationships in multi-task learning can be effectively modeled through input gradient similarity. By incorporating a novel saliency-based regularizer, SRDML provides a theoretically grounded framework for improving task relation interpretability and reducing generalization error, as validated through extensive experiments.

SAMC, on the other hand, introduced a saliency-augmented memory mechanism for continual learning, inspired by cognitive neuroscience principles. By prioritizing the storage of task-relevant information and employing an adaptive memory inpainting mechanism, SAMC achieved a balance between storage efficiency and generalizability, addressing the inherent challenges of continual learning.

Together, these methods illustrate the power of saliency in guiding functional-space learning, offering novel solutions to long-standing challenges in multi-task and continual learning. These foundational contributions pave the way for further exploration of functional-space guided methodologies, as we delve into dynamic mapping approaches in the next chapter.

Chapter 3

Harnessing Function Space of Machine Learning Models for Better Generalizability

Dynamic environments and evolving data distributions challenge the adaptability and generalization of machine learning models. A critical step in addressing these challenges is to establish a flexible mapping between parameter and functional spaces, enabling models to adapt dynamically to changes. This chapter explores Dynamic Mapping Methods for Functional-Space Learning, focusing on how mutual mappings between these spaces can enhance the representation and generalization capabilities of models.

We introduce DRAIN, a framework that leverages dynamic graph-based modeling to capture and adapt to temporal changes in data distributions. By facilitating a continuous exchange of information between parameter and functional spaces, DRAIN enables robust learning in dynamic and non-stationary settings. Its novel approach to dynamic mappings paves the way for scalable, interpretable, and generalizable solutions in functional-space-guided learning.

The sections that follow delve into the theoretical underpinnings, design principles, and empirical evaluations of DRAIN, illustrating its contributions to this evolving research area.

3.1 Bridging Parameter and Functional Spaces: A Dynamic Framework for OOD Generalization

3.1.1 Introduction

In machine learning, researchers often assume that training and test data follow the same distribution for the trained model to work on test data with some generalizability. However, in reality, this assumption usually cannot be satisfied, and when we cannot make sure the trained model is always applied in the same domain where it was trained. This motivates Domain Adaptation (DA) which builds the bridge between source and target domains by characterizing the transformation between the data from these domains [13, 40, 143]. However, in more challenging situations when target domain data is unavailable (e.g., no data from an unknown area, no data from the future, etc.), we need a more realistic scenario named Domain Generalization (DG) [125, 5, 31].

Most existing works in DG focus on generalization among domains with categorical indices, such as generalizing the trained model from one dataset (e.g., MNIST [67]) to another (e.g., SVHN [104]), from one task (e.g., image classification [61]) to another (e.g., image segmentation [80]), etc. However, in many real-world applications, the "boundary" among different domains is unavailable and difficult to detect, leading to a *concept drift* across the domains. For example, when a bank leverages a model to predict whether a person will be a "defaulted borrower", features like "annual incoming", "profession type", and "marital status" are considered. However, due to

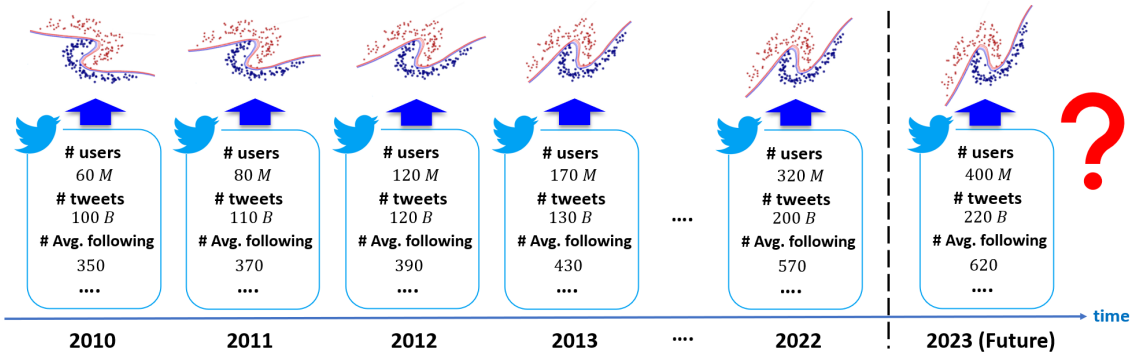


Figure 3.1: **An illustrative example of temporal domain generalization.** Consider training a model for some classification tasks based on the annual Twitter dataset such that the trained model can generalize to the future domains (e.g., 2023). The temporal drift of data distribution can influence the prediction model such as the rotation of the decision boundary in this case.

the temporal change of the society, how these feature values indicate the prediction output should change accordingly following some trends that could be predicted somehow in a range of time. Figure 3.1 shows another example, seasonal flu prediction via Twitter data which evolves each year in many aspects. For example, monthly active users are increasing, new friendships are formed, the age distribution is shifting under some trends, etc. Such temporal change in data distribution gradually outdated the models. Correspondingly, suppose there was an ideal, always update-to-date model, then the model parameters should gradually change correspondingly to counter the trend of data distribution shifting across time. It can also "predict" what the model parameters should look like in an arbitrary (not too far) future time point. This requires the power of *temporal* domain generalization.

However, as an extension of traditional DG, temporal DG is extremely challenging yet promising. Existing DG methods that treat the domain indices as a categorical variable may not be suitable for temporal DG as they require the domain boundary as apriori to learn the mapping from source to target domains [101, 100, 10, 5]. Until now, temporal domain indices have been well explored only in DA [48, 107, 147] but not DG. There are very few existing works in temporal DG due to its big challenges.

One relevant work is *Sequential Learning Domain Generalization* (S-MLDG) [71] that proposed a DG framework over sequential domains via meta-learning [38]. S-MLDG meta-trains the target model on all possible permutations of source domains, with one source domain left for meta-test. However, S-MLDG in fact still treats domain index as a categorical variable, and the method was only tested on categorical DG dataset. A more recent paper called *Gradient Interpolation* (GI) [103] proposes a temporal DG algorithm to encourage a model to learn functions that can extrapolate to the near future by supervising the first-order Taylor expansion of the learned function. However, GI has very limited power in characterizing model dynamics because it can only learn how the activation function changes along time while making all the remaining parameters fixed across time.

The advancement of temporal domain generalization is challenged by several critical bottlenecks, including **1) Difficulty in characterizing the data distribution drift and its influences on models.** Modeling the temporally evolving distributions requires making the model time-sensitive. Intuitive ways include feeding the time as an input feature to the model, which is well deemed simple yet problematic as it discards the other features' dependency on time and dependency on other *confounding* factors changed along time [147]. Another possible way is to make the model parameters a function of time. However, these ways cannot generalize the model to future data as long as the whole model's dynamics and data dynamics are not holistically modeled. **2) Lack of expressiveness in tracking the model dynamics.** Nowadays, complex tasks have witnessed the success of big complex models (e.g., large CNNs [30]), where the neurons and model parameters are connected as a complex graph structure. However, they also significantly challenge tracking their model dynamics in temporal DG. An expressive model dynamics characterization and prediction requires mapping data dynamics to *model dynamics* and hence the graph dynamics of model parameters across time. This is a highly open problem, especially

for the temporal DG area. **3) Difficulty in theoretical guarantee on the performance.** While there are fruitful theoretical analyses on machine learning problems under the independent and identically distributed assumptions [45], similar analyses meet substantial hurdles to be extended to out-of-distribution (OOD) problem due to the distribution drift over temporally evolving domains. Therefore, it is essential to enhance the theoretical analyses on the model capacity and theoretical relation among different temporal domain generalization models.

To address all the above challenges, we propose a Temporal Domain Generalization with **DR**ift-**A**ware dynam**I**c neural **N**etworks (**DRAIN**) framework that solves all challenges above simultaneously. Specifically, we propose a generic framework to formulate temporal domain generalization by a Bayesian treatment that jointly models the relation between data and model dynamics. To instantiate the Bayesian framework, a recurrent graph generation scenario is established to encode and decode the dynamic graph-structured neural networks learned across different timestamps. Such a scenario can achieve a fully time-sensitive model and can be trained in an end-to-end manner. It captures the temporal drift of model parameters and data distributions and can predict the models in the future *without* the presence of future data.

Our contributions include: 1) We develop a novel and adaptive temporal domain generalization framework that can be trained in an end-to-end manner. **2)** We innovatively treat the model as a dynamic graph and leverage graph generation techniques to achieve a fully time-sensitive model. **3)** We propose to use the sequential model to learn the temporal drift adaptively and leverage the learned sequential pattern to predict the model status on the future domain. **4)** We provide theoretical analysis on both uncertainty quantification and generalization error of the proposed method. **5)** We demonstrate our model's efficacy and superiority with extensive experiments.

3.1.2 Problem formulation: Temporal Domain Generalization.

We consider prediction tasks where the data distribution evolves with time. During training, we are given T observed source domains $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$ sampled from distributions on T arbitrary time points $t_1 \leq t_2 \leq \cdots \leq t_T$, with each $\mathcal{D}_s = \left\{(\mathbf{x}_i^{(s)}, y_i^{(s)}) \in \mathcal{X}_s \times \mathcal{Y}_s\right\}_{i=1}^{N_s}$, $s = 1, 2, \cdots, T$ where $\mathbf{x}_i^{(s)}$, $y_i^{(s)}$ and N_s denotes the input feature, label and sample size at timestamp t_s, respectively, and $\mathcal{X}_s$, $\mathcal{Y}_s$ denotes the input feature space and label space at timestamp t_s, respectively. The trained model will only be tested on some target domain *in the future*, i.e., $\mathcal{D}_{T+1}$ where $t_{T+1} \geq t_T$. Our setting further assumes the existence of concept drift across different domains, i.e., the domain distribution is changing across time by following some patterns.

Our *goal* is to build a model that proactively captures the concept drift. Given labeled data from the source domains $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$, we learn the mapping function $g_{\boldsymbol{\omega}_s} : \mathcal{X}_s \to \mathcal{Y}_s$ on each domain $\mathcal{D}_s$, $s = 1, 2, \cdots, T$ where $\boldsymbol{\omega}_s$ denotes the function parameters at timestamp t_s, respectively, and then predict the dynamics across the parameters $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_T$. Finally, we predict the parameters $\boldsymbol{\omega}_{T+1}$ for the mapping function $g_{\boldsymbol{\omega}_{T+1}} : \mathcal{X}_{T+1} \to \mathcal{Y}_{T+1}$ on the unseen future domain. As shown in Figure 3.1, due to the temporal drift in data distribution, e.g. the input features such as Twitter user age distribution and number of tweets increase each year, the prediction model is expected to evolve accordingly, e.g. the magnitude of model parameter weights will decrease annually. Despite the necessity, handling the above problem is an open research area due to several existing challenges: 1) Difficulty in characterizing data distribution drift as well as how it influences the model. 2) Lack of expressiveness in automatically capturing the dynamics of how neural network evolves across time. 3) Theoretical guarantee on model's performance (e.g., generalization error, uncertainty) on future domains is hard to obtain due to the unknown and (potentially) complicated

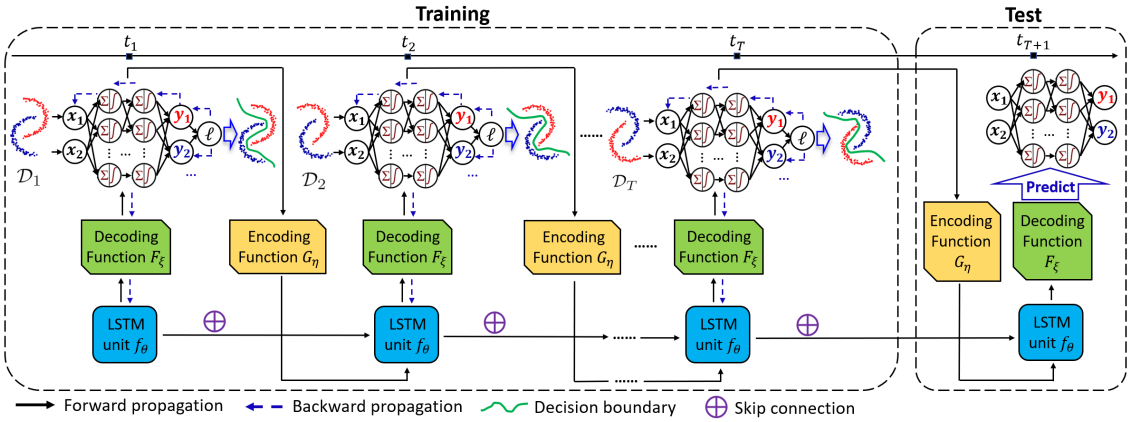


Figure 3.2: A high-level overview of our DRAIN framework. Best viewed in color.

concept drift.

3.1.3 Proposed Method: DRAIN

In this section, we introduce how we address the challenges mentioned above. For the first challenge, we build a systematic Bayesian probability framework to represent the concept drift over the domains, which instantly differentiates our work from all existing methods in DG. For the second challenge, we propose modeling a neural network with changing parameters as a dynamic graph and achieving a temporal DG framework that can be trained end-to-end by graph generation techniques. We further improve the proposed method's generalization ability by introducing a skip connection module over different domains. Finally, to handle the last challenge, we explore theoretical guarantees of model performance under the challenging temporal DG setting and provide theoretical analyses of our proposed method, such as uncertainty quantification and generalization error.

A probabilistic view of concept drift in temporal domain generalization

To perform domain generalization over temporally indexed domains, we need to capture the concept drift within a given time interval. From a probabilistic point

of view, for each domain $\mathcal{D}_s$, $s = 1, 2, \cdots, T$, we can learn a neural network $g_{\boldsymbol{\omega}_s}$ by maximizing the conditional probability $\Pr(\boldsymbol{\omega}_s|\mathcal{D}_s)$, where $\boldsymbol{\omega}_s$ denotes the status of model parameters at timestamp t_s. Due to the evolving distribution of $\mathcal{D}_s$, the conditional probability $\Pr(\boldsymbol{\omega}_s|\mathcal{D}_s)$ will change over time accordingly. Our ultimate goal is to predict $\boldsymbol{\omega}_{T+1}$ given all training data $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$ ($\mathcal{D}_{1:T}$ for short), i.e., $\Pr(\boldsymbol{\omega}_{T+1}|\mathcal{D}_{1:T})$. By the Law of total probability, we have

$$\Pr\left(\boldsymbol{\omega}_{T+1} \mid \mathcal{D}_{1:T}\right) = \int_{\Omega} \underbrace{\Pr\left(\boldsymbol{\omega}_{T+1} \mid \boldsymbol{\omega}_{1:T}, \mathcal{D}_{1:T}\right)}_{\text{inference}} \cdot \underbrace{\Pr\left(\boldsymbol{\omega}_{1:T} \mid \mathcal{D}_{1:T}\right)}_{\text{training}} d\boldsymbol{\omega}_{1:T}, \tag{3.1}$$

where Ω is the space for $\boldsymbol{\omega}_{1:T}$. The first term in the integral represents the inference phase, i.e., how we predict the status of the target neural network in the future (namely, $\boldsymbol{\omega}_{T+1}$) given all history statuses, while the second term denotes the training phase, i.e., how we leverage all source domains' training data $\mathcal{D}_{1:T}$ to obtain the status of the neural network on each source domain, namely $\boldsymbol{\omega}_{1:T}$. By the chain rule of probability, we can further decompose the training phase as follows:

$$\begin{aligned} \Pr\left(\boldsymbol{\omega}_{1:T} \mid \mathcal{D}_{1:T}\right) &= \prod_{s=1}^{T} \Pr\left(\boldsymbol{\omega}_s \mid \boldsymbol{\omega}_{1:s-1}, \mathcal{D}_{1:T}\right) \\ &= \Pr\left(\boldsymbol{\omega}_1 \mid \mathcal{D}_1\right) \cdot \Pr\left(\boldsymbol{\omega}_2 \mid \boldsymbol{\omega}_1, \mathcal{D}_{1:2}\right) \cdots \Pr\left(\boldsymbol{\omega}_T \mid \boldsymbol{\omega}_{1:T-1}, \mathcal{D}_{1:T}\right). \end{aligned} \tag{3.2}$$

Here we assume for each time point t_s, the model parameter $\boldsymbol{\omega}_s$ only depends on the current and previous domains (namely, $\{\mathcal{D}_i : i \leq s\}$), and there is no access to future data (even unlabeled). Now we can break down the whole training process into $T-1$ steps, where each step corresponds to learning the model parameter on the new domain conditional on parameter statuses from the history domains and training data, i.e., $\Pr\left(\boldsymbol{\omega}_{s+1} \mid \boldsymbol{\omega}_{1:s}, \mathcal{D}_{1:s}, \mathcal{D}_{s+1}\right)$, $\forall\ s < T$.

Neural network with dynamic parameters

Since the data distributions change temporally, the parameter $\boldsymbol{\omega}_s$ in $g_{\boldsymbol{\omega}_s}$ needs to be updated accordingly to address the temporal drift across the domains. In this work, we consider leveraging *dynamic graphs* to model the temporally evolving neural networks in order to retain maximal expressiveness.

Intuitively, a neural network $g_{\boldsymbol{\omega}}$ can be represented as an edge-weighted graph $G = (V, E, \psi)$, where each node $v \in V$ represents a neuron of $g_{\boldsymbol{\omega}}$ while each edge $e \in E$ corresponds to a connection between two neurons in $g_{\boldsymbol{\omega}}$. Moreover, given a connection e between neuron u and v, i.e., $e = (u, v) \in E$, function $\psi : E \to \mathbb{R}$ denotes the weight parameter between these two neurons, i.e., $\psi(u, v) = w_{u,v}$, $\forall\ (u, v) \in E$. Essentially, $\boldsymbol{\omega} = \psi(E) = \{w_{u,v} : (u, v) \in E\}$ is a set of parameter values indexed by all edges in E and $\boldsymbol{\omega}$ represents the entire set of parameters for neural network g. Notice that we give a general definition of $g_{\boldsymbol{\omega}}$ so that both shallow models (namely, linear model) and deep neural networks (e.g., MLP, CNN, RNN, GNN) can be treated as special cases here. We aim to characterize the potential drift across domains by optimizing and updating the graph structure (i.e., edge weight) of $g_{\boldsymbol{\omega}}$. [162] have proven that optimizing the graph structure of the neural network could have a smaller search space and a more smooth optimization procedure than exhaustively searching over all possible connectivity patterns.

We consider the case where the architecture or topology of neural network $g_{\boldsymbol{\omega}}$ is given, i.e., V and E are fixed, while the parameter $\boldsymbol{\omega}$ is changing constantly w.r.t time point t_s. In this sense, we can write $\boldsymbol{\omega}_s = \psi(E|s)$ where $\psi(\cdot|s)$ (*abbrev.* ψ_s) depends only on time point t_s. Now the triplet $G = (V, E, \psi_s)$ defines a *dynamic graph* with evolving edge weights.

End-to-end learning of concept drift

Given history statuses $\{\boldsymbol{\omega}_{1:s}\}$ of the neural network learned from $\{\mathcal{D}_{1:s}\}$, we aim at generalizing and extrapolating $\boldsymbol{\omega}_{s+1}$ so that it produces good performance on the new domain $\mathcal{D}_{s+1}$ in an end-to-end manner. In fact, by viewing the neural networks $\{\boldsymbol{\omega}_{1:s}\}$ as dynamically evolving graphs, a natural choice is to characterize the latent graph distribution of $\{\boldsymbol{\omega}_{1:s}\}$ by learning from its evolving trend. Consequently, $\boldsymbol{\omega}$'s can be directly sampled from the distribution for the prediction in future domains.

We characterize the latent distribution of $\{\boldsymbol{\omega}_{1:s}\}$ as a sequential learning process based on a recurrent architecture, and each unit f_θ in the recurrent model is parameterized by θ to generate $\boldsymbol{\omega}_s$ by accounting for previous $\{\boldsymbol{\omega}_i : i < s\}$. Specifically, at each recurrent block (i.e., time step) t_s, f_θ produces two outputs (m_s, h_s), where m_s is the current memory state and h_s is a latent probabilistic distribution (i.e., hidden output of f_θ) denoting the information carried from previous time steps. The latent probabilistic distribution h_t allows us to generate the dynamic graph $\boldsymbol{\omega}_s$ by a decoding function $F_\xi(\cdot)$. Intuitively, different from existing works that train and regularize a neural network on single domain [103], here we focus on directly searching for distribution of networks with "good architectures". Lastly, the sampled $\boldsymbol{\omega}_s$ is encoded by a graph encoding function $G_\eta(\cdot)$, which then serves as the input of next recurrent block. Such a recurrent model is trained on a single domain $\mathcal{D}_s$ to generate $\boldsymbol{\omega}_s$ for prediction by minimizing the empirical loss, i.e., $\min_{\theta,\xi,\eta} \sum_{i=1}^{N_s} \ell\big(g_{\boldsymbol{\omega}_s}(\mathbf{x}_i^{(s)}), y_i^{(s)}\big)$, where $\ell(\cdot,\cdot)$ can be cross-entropy for classification or MSE for regression. The optimal $\boldsymbol{\omega}_s$ on domain $\mathcal{D}_s$ will then be fed into the next domain $\mathcal{D}_{s+1}$ along with the memory state m_s as input to guide the generation of $\boldsymbol{\omega}_{s+1}$ until the entire training phase is done. For the inference phase, we feed the optimal parameters from the last training domain, namely $\boldsymbol{\omega}_T$, into the encoding function and leverage the recurrent block, together with the memory state m_T to predict the latent vector on the future domain $\mathcal{D}_{T+1}$, followed by the decoding function to decode the latent vector and generate the

optimal parameters $\boldsymbol{\omega}_{T+1}$.

Less forgetting and better generalization

During the training of recurrent models, the performance degradation problem is also likely to be encountered. Such a problem can be severe in temporal DG since a more complicated concept correlation exists between each domain. In addition, if the training procedure on each domain $\mathcal{D}_s$ takes a large number of iterations to converge, we may also observe the forgetting phenomenon (i.e., the recurrent model f_θ will gradually focus on the current training domain and have less generalization capability for future domains). To alleviate such a phenomenon, we leverage a straightforward technique - skip connection to bridge the training on $\mathcal{D}_s$ with previous domains $\{\mathcal{D}_{1:s-1}\}$. Specifically,

$$\Phi\Big(\boldsymbol{\omega}_s, \big\{\boldsymbol{\omega}_{s-\tau:s-1}\big\}\Big) \coloneqq \boldsymbol{\omega}_s + \lambda \cdot \sum\nolimits_{i=s-\tau}^{s-1} \boldsymbol{\omega}_i, \tag{3.3}$$

where λ is regularization coefficient and τ denotes the size of the *sliding window*. The skip connection could enforce the generated network parameters $\boldsymbol{\omega}_s$ to contain part of the previous network's information, and the implementation of the fixed-sized sliding window can better alleviate the potential drawback of the computational cost.

3.1.4 Theoretical analyses

In this section, we provide a theoretical analysis of our proposed framework's performance in the target domain. Our analyses include uncertainty quantification and generalization error. Uncertainty characterizes the dispersion or error of an estimate due to the noise in measurements and the finite size of data sets, and smaller uncertainty means less margin of error over the model predictions. On the other hand, generalization error measures how accurate the model's prediction is on unseen data. Our analyses show that our proposed DRAIN achieves **both better prediction ac-**

curacy as well as smaller margin of error on the target domain compared with online and offline DG baselines.

First, we introduce two DG methods, namely online baseline and offline baseline, as defined below:

Definition 3.1.1. *Given timestamp t_{s+1} and domains $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_{s+1}$, and model parameter state from previous timestamp, namely ω_s. Define online model $\mathcal{M}_{on}$ and offline model $\mathcal{M}_{off}$ as $\omega_{s+1} = argmax_{\omega_{s+1}} \Pr(\omega_{s+1}|\omega_s, \mathcal{D}_{s+1})$ and $\omega_{s+1} = argmax_{\omega_{s+1}} \Pr(\omega_{s+1}|\mathcal{D}_{1:s+1})$, respectively.*

Offline method $\mathcal{M}_{\text{off}}$ is trained using ERM over all source domains, while online method $\mathcal{M}_{\text{on}}$ considers one-step finetuning over the model parameter on each new domain's dataset. Both $\mathcal{M}_{\text{off}}$ and $\mathcal{M}_{\text{on}}$ are time-oblivious, i.e., unaware of the concept drift over time.

Assumption 3.1.2. *Consider a parameterized function $q_\theta(\cdot)$ to approximate $P(\omega_{t+1}|\omega_t)$, which is the unknown ground-truth concept drift of the model parameter distribution. It is assumed that the prior over q_θ follows a normal distribution with: $\mathbb{E}[q_{\theta_0}(\omega)] = \omega$, $Var(q_{\theta_0}(\omega)) = \sigma^2_{\theta_0}$, $\forall\, \omega \in \Omega$.*

Definition 3.1.3 (Predictive Distribution)**.** *Given training sample $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$, and input feature from future domain, namely $\mathbf{x}_{T+1}$, the predictive distribution can be defined as*

$$\Pr\left(\hat{y} \mid \mathbf{x}_{T+1}, \mathcal{D}_{1:T}\right) = \int \Pr\left(\hat{y} \mid \mathbf{x}_{T+1}, \omega_{T+1}\right) \Pr\left(\omega_{T+1} \mid \mathcal{D}_{1:T}\right) d\omega_{T+1}. \tag{3.4}$$

Our first theorem below shows that by capturing the concept drift over the sequential domains, our proposed method always achieves the smallest uncertainty in prediction on the future domain.

Theorem 3.1.4 (Uncertainty Quantification). *Given training domains $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T$ where $Var(\mathcal{D}_i)$ is the same, we have the following inequality over each method's predictive uncertainty, i.e., the variance of predictive distribution as defined in Eq. 3.4: $Var(\mathcal{M}_{ours}) < Var(\mathcal{M}_{on}) \leq Var(\mathcal{M}_{off})$.*

Our second theorem shows that, besides uncertainty, our proposed method can also achieves the smallest generalization error thanks to learning the concept drift.

Definition 3.1.5. *Given predictive distribution in Eq. 3.4, as well as ground-truth label y_{T+1} from the future domain, define the predictive or generalization error as $err \coloneqq \ell(\mathbb{E}[P(\hat{y}|\mathbf{x}_{T+1}, \mathcal{D}_{1:T})], y_{T+1})$.*

Theorem 3.1.6 (Generalization Error). *Assume $g_\omega(\cdot)$ has Lipschitz constant with upper bound L_{upper} and lower bound L_{lower} w.r.t ω. We have the following inequality over each method's predictive error defined above: $err(\mathcal{M}_{ours}) < err(\mathcal{M}_{on}) < err(\mathcal{M}_{off})$.*

Complexity Analyses. In our implementation, the encoding and decoding functions are instantiated as MLPs. The total number of parameters of the encoding and decoding functions is $\mathcal{O}(Nd + C)$, which is *linear* in N. Here N is the number of parameters in predictive models (namely $\boldsymbol{\omega}$), d is the width (i.e., number of neurons) of the last hidden layer of the encoding and decoding functions, and C denotes the number of parameters for all the layers before the last for the encoding and decoding functions. Additionally, in many situations, the first few layers of representation learning could be shared. Hence, we do not need to generate all the parameters in $\boldsymbol{\omega}$, but just the last few layers.

3.1.5 Experiments

In this section, we present the performance of DRAIN against other state-of-the-art approaches with both quantitative and qualitative analysis. The experiments in this paper were performed on a 64-bit machine with a 4-core Intel Xeon W-2123

Table 3.1: Performance comparison of all methods in terms of misclassification error (in %) for classification tasks and mean absolute error (MAE) for regression tasks (both smaller the better.) Results of comparison methods on all datasets except "Appliance" are reported from [103]. "-" denotes that the method could not converge on the specific dataset.

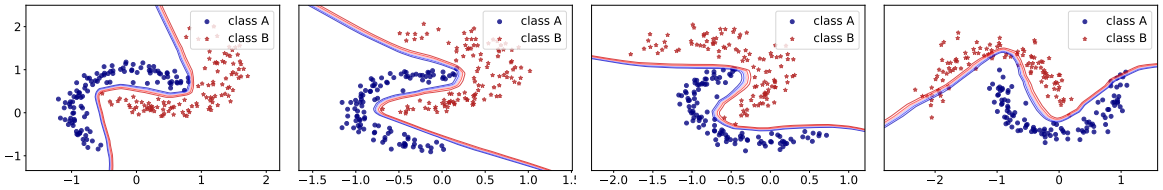
Model	**Classification** (in %)					**Regression**	
	2-Moons	Rot-MNIST	ONP	Shuttle	Elec2	House	Appliance
Offline	22.4 ± 4.6	18.6 ± 4.0	**33.8 ± 0.6**	0.77 ± 0.1	23.0 ± 3.1	11.0 ± 0.36	10.2 ± 1.1
LastDomain	14.9 ± 0.9	17.2 ± 3.1	36.0 ± 0.2	0.91 ± 0.18	25.8 ± 0.6	10.3 ± 0.16	9.1 ± 0.7
IncFinetune	16.7 ± 3.4	10.1 ± 0.8	34.0 ± 0.3	0.83 ± 0.07	27.3 ± 4.2	9.7 ± 0.01	8.9 ± 0.5
CDOT	9.3 ± 1.0	14.2 ± 1.0	34.1 ± 0.0	0.94 ± 0.17	17.8 ± 0.6	-	-
CIDA	10.8 ± 1.6	9.3 ± 0.7	34.7 ± 0.6	-	14.1 ± 0.2	9.7 ± 0.06	8.7 ± 0.2
GI	3.5 ± 1.4	7.7 ± 1.3	36.4 ± 0.8	0.29 ± 0.05	16.9 ± 0.7	9.6 ± 0.02	8.2 ± 0.6
DRAIN	**3.2 ± 1.2**	**7.5 ± 1.1**	38.3 ± 1.2	**0.26 ± 0.05**	**12.7 ± 0.8**	**9.3 ± 0.14**	**6.4 ± 0.4**

@ 3.60GHz, 32GB memory, and NVIDIA Quadro RTX 5000. Additional experiment settings and results (e.g., hyperparameter settings and scalability analysis) are demonstrated in the appendix.

Experiment setting

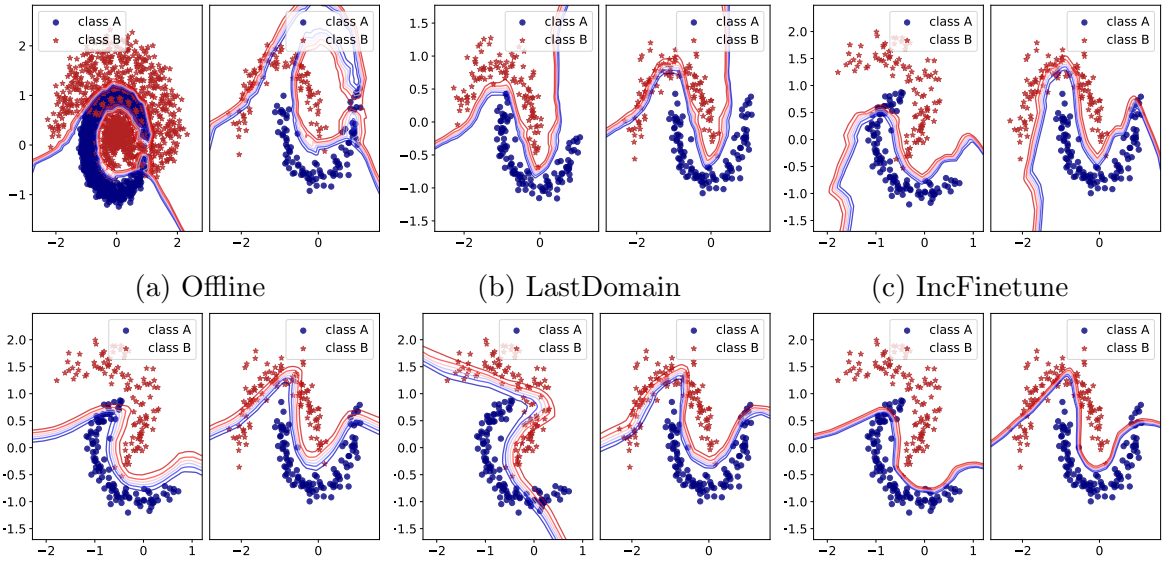
Datasets. We compare with the following classification datasets: Rotated Moons (2-Moons), Rotated MNIST (Rot-MNIST), Online News Popularity (ONP), Electrical Demand (Elec2), and Shuttle; and the following regression datasets: House prices dataset (House), Appliances energy prediction dataset (Appliance). The first two datasets are synthetic, where the rotation angle is used as a proxy for time. The remaining datasets are real-world datasets with temporally evolving characteristics.

Comparison Methods. We adopt three sets of comparison methods: **practical baselines** that do not consider the concept drift, including *1). Offline* that treats all source domains as a single domain, *2). LastDomain* that only employs the last training domain, and *3). IncFinetune* that sequentially trains on each training domain. **Continuous domain adaptation methods** that focus only on DA, including *1). CDOT* [107] that transports most recent labeled examples to the future, and *2). CIDA* [147] that specifically tackles the continuous DA problem; and one **temporal domain generalization method**: *GI* [103].



(a) Training Domain $\mathcal{D}_2$ (b) Training Domain $\mathcal{D}_4$ (c) Training Domain $\mathcal{D}_6$ (d) Testing Domain $\mathcal{D}_9$

Figure 3.3: **Visualization of the decision boundary of DRAIN** (blue dots and red stars represent different data classes). As the distribution of data points is consistently changing, as shown in Figure 3.3a - 3.3c, DRAIN can effectively characterize such a temporal drift and predict accurate decision boundaries on the unseen testing domain in Figure 3.3d.



(a) Offline (b) LastDomain (c) IncFinetune

(d) CDOT (e) CIDA (f) GI

Figure 3.4: **Visualization of decision boundary** (blue dots and red stars represent different data classes), where the right subfigure of comparison methods Figure 3.4a - 3.4f demonstrate the decision boundary predicted for the test domain $\mathcal{D}_{T+1}$, the left subfigure in Figure 3.4a shows the decision boundary learned from the all data points in the concatenated training domain ($[\mathcal{D}_1, \cdots, \mathcal{D}_T]$), the left subfigure in Figure 3.4b shows the decision boundary learned from all samples in the last training domain $\mathcal{D}_T$, and the left subfigures in Figure 3.4c - 3.4f show the decision boundary learned on $\mathcal{D}_4$.

All experiments are repeated 10 times for each method, and we report the average results and the standard deviation in the following quantitative analysis.

Quantitative analysis

We first illustrate the performance of our proposed method against comparison methods. The experiments are conducted in both classification and regression tasks with the domain generalization setting, i.e., models are trained on the training domains and deployed on the unseen testing domain.

As can be seen from Table 3.1, DRAIN consistently achieves competitive results across most datasets. Specifically, DRAIN excels the second-best approaches on Elec2 (CIDA), House (GI), and Appliance (GI) by a great margin.

The only exception is the ONP dataset, where the Offline method achieves the best result and all state-of-the-art methods cannot generalize well on unseen testing domains since the ONP dataset does not exhibit a strong concept drift. Additionally, all time-oblivious baselines perform rather unsatisfactorily since they are not capable of handling the concept drift of the data distribution. Both CDOT and CIDA can generate better results than time-oblivious baselines, yet their generalization ability on the unseen domains is still limited as the maintained time-invariant representation in both methods cannot address the concept drift without any data in the testing domain. As the only method that addresses the temporal domain generalization problem, GI imposes a gradient regularization with a non-parametric activation function to handle the concept drift, which relies too much on the task-specific heuristic. In contrast, DRAIN proposes to sequentially model each domain in an end-to-end manner, which could address the concept drift more inherently.

Qualitative analysis

We compare different methods qualitatively by visualizing the decision boundary on the 2-Moons dataset. As shown in Figure 3.3a - 3.3c, we demonstrate the decision boundary predicted by DRAIN at $\mathcal{D}_2$, $\mathcal{D}_4$, $\mathcal{D}_6$ training domains, and the final predicted decision boundary on the testing domain $\mathcal{D}_9$ (Figure 3.3d). As can be seen, DRAIN

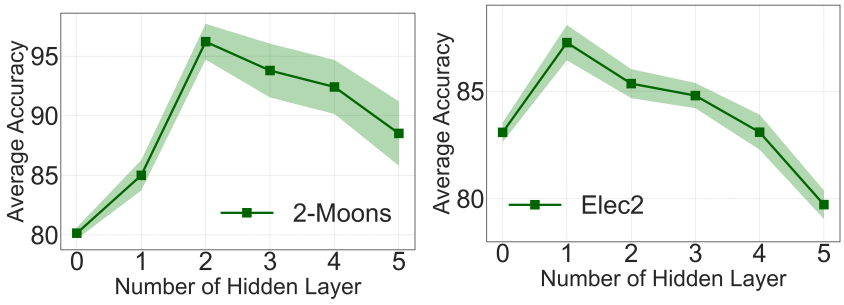


Figure 3.5: Sensitivity analysis on the number of layers of the generated neural network by DRAIN.

can successfully characterize the concept drift by sequentially modeling the $\{\mathcal{D}_T\}$, and the learned decision boundary could rotate correctly along time.

We further visualize the decision boundary learned by other comparison methods in Figure 3.4a - 3.4f. Firstly, the left subfigure in Figure 3.4a shows the decision boundary learned by the Offline method on the concatenated training domains $\{\mathcal{D}_{1:T}\}$, and the learned decision boundary overfits the training data and shows poor performance when generalizing on the unseen testing domain (the right subfigure of 3.4a). Furthermore, as the current state-of-the-art continuous domain adaptation methods, CDOT transports the most recent labeled data points in $\mathcal{D}_T$ to the future, which makes the learned decision boundary almost temporal-invariant (Figure 3.4d) and cannot generalize well in the scenario of domain generalization. CIDA utilizes the adversarial training technique to solve the domain adaptation, yet the predicted decision boundary in Figure 3.4e is less stable than other state-of-the-art methods due to its model complexity. Lastly, even though GI is the only method proposed to tackle the temporal domain generalization problem, the produced decision boundaries, as shown in both the training domain and testing domain (Figure 3.4f), are still less accurate than our proposed method, since they heavily utilize heuristics to regularize the gradient.

Sensitivity analysis

We conduct sensitivity analysis on the depth of the neural network g_{ω_s} for DRAIN. As shown in Figure 3.5, the optimal number of hidden layers for g_{ω_s} is 2 and 1 on 2-Moons and Electric datasets, respectively. The curve on both datasets has an inverse "U" shape, meaning that too few layers may limit the general expressiveness of our model, while too many layers could potentially hurt the generalization ability due to overfitting.

Ablation study

We further conduct an ablation study on three datasets to evaluate the effect of different components in DRAIN, and the results are exhibited in Table 3.2. Specifically, we remove the sequential learning model in DRAIN, and the resulting ablated model ✗ RNN corresponds to the offline baseline model. We also independently remove the skip connection module to let the sequential learning model uniformly acquire information from all previous domains, and the resulting model is named ✗ Skip.C.

Table 3.2: Ablation study. Comparison of performance between our method and two alternatives across two datasets for classification tasks and one dataset for regression tasks.

Ablation	2-Moons	Rot-MNIST	House
✗ RNN	22.4 ± 4.6	19.5 ± 3.4	11.0 ± 0.36
✗ Skip.C	7.1 ± 1.3	10.3 ± 1.7	9.7 ± 0.13
DRAIN	**3.2 ± 1.2**	**7.5 ± 1.1**	**9.3 ± 0.14**

As shown in the table, each component can effectively contribute to the overall model performance, and modeling the temporal correlation between all domains by a sequential model can provide a rather larger performance gain. In addition, removing the skip connection in the sequential learning model would make DRAIN hard to capture the long-range temporal dependency among domains, since long-range domain information could potentially be forgotten during the model learning.

3.2 Domain Adaptation via Prompt Tuning for Time Series Data

3.2.1 Introduction

Due to the prevalence of time series sensor data, time series analysis has found applications in various real-world scenarios, including human activity recognition [3], sleep stage classification [168], and machine fault diagnosis [145, 146, 69]. In these applications, time series data are measured under different subjects, operating conditions, or sensor configurations (*i.e.*, domains). In other words, time series analysis should be conducted across different domains. Unfortunately, the labels of time series data are difficult to collect due to the high costs of the labeling process [151]. To mitigate labeling costs, researchers aim to leverage labeled data from some domains (*i.e.*, source domains) to infer labels for unlabeled data in other domains (*i.e.*, target domains) [149], which is defined as a time series domain adaptation problem. For example, the goal of the transponder fault diagnosis problem is to detect the working statuses of transponders (*i.e.*, normal or abnormal) based on fiber-optic signals. In this problem, the model is trained under certain working modes (*e.g.*, single mode) using labeled time series data, and then this trained model is applied to other working modes (*e.g.*, multi-mode).

However, the time series domain adaptation problem is highly challenging due to complex dynamic time series patterns, distribution shift (*i.e.*, different distributions of inputs among different domains), and possible label shift (*i.e.*, different distributions of labels among different domains) [7, 46, 22]. These challenges have been extensively investigated by researchers, leading to the proposal of various methods to address the domain gap, such as kernel matching [82], context information alignment [65], and temporal-spectral fusion [157]. Most existing methods, however, primarily focus on

domain adaptation from a single source domain. Yet, it is more crucial to investigate it from multiple sources. This is because the more source domains are utilized, the greater potential improvements it can achieve. For instance, the collection of labeled signal data from more modes facilitates a better understanding of transponder statuses. Despite the importance of the multi-source domain adaptation problem, it is rarely explored in previous literature and requires attention and extensive investigations from researchers.

In order to effectively handle the multi-source time series domain adaptation problem, three important challenges need to be overcome: **1. The lack of exploration to utilize domain-specific information for domain adaptation.** Existing domain adaptation methods primarily focus on learning a common feature extractor to encode time series inputs from different source domains into domain-invariant representations, and then apply this feature extractor to the target domain [112, 76, 88, 54, 151]. While this strategy has its rationale, it often overlooks domain-specific information (*i.e.*, information unique to a specific time series domain), such as global trends, local trends, and temporal patterns. Such domain-specific information is valuable to evaluate which source domains are more suitable for adaptation to the target domain. **2. The difficulty to learn domain-specific information that changes over time.** While it is important to capture domain-specific information for better domain adaptation, such information can be dynamically changing, which is extremely difficult to capture. In the example of the transponder fault diagnosis problem, different domains generate different distributions of fiber-optic signals, which are important domain-specific information to capture. However, such distributions can be shifted drastically when the transponder suddenly suffers from a failure. **3. The difficulty to evaluate learned domain-specific information.** Not only is learning domain-specific information difficult, but it is also challenging to evaluate learned domain-specific information. In other words, it is unclear whether learned

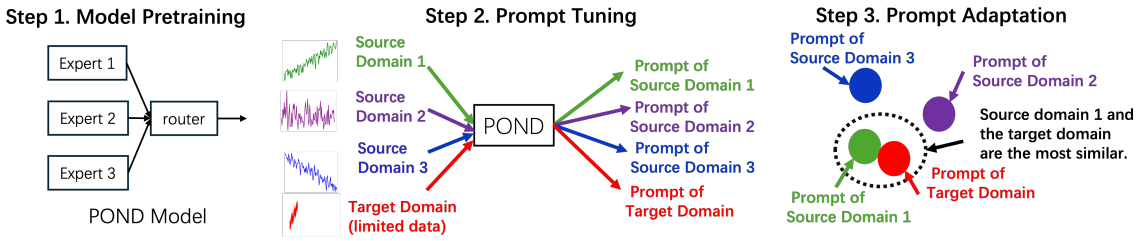


Figure 3.6: Pipeline of our proposed POND model: Step 1 pretrains the proposed POND model; Step 2 learns prompts of all source domains and the target domain; Step 3 utilizes learned prompts to select the most similar source domain to the target domain for domain adaptation.

domain-specific information accurately reflects the true one. This ambiguity arises because domain-specific information is often associated with unique but inexplicable underlying patterns. Unlike images and languages with human-recognizable features, such time series patterns are difficult for humans to understand [89]. Consequently, it becomes challenging, if not impossible, for humans to evaluate whether learned domain-specific information matches such time series patterns.

In order to tackle these three challenges simultaneously, we propose PrOmpt-based domaiN Discrimination (POND), the first framework to utilize prompts for time series domain adaptation to our knowledge. Its pipeline is shown in Figure 3.6, which consists of three steps: model pertaining, prompt tuning, and prompt adaptation. Specifically, to address Challenge 1, we extend the idea of prompt tuning to time series analysis and learn prompts to capture common and domain-specific information. To handle Challenge 2, we introduce a conditional module for each source domain to generate prompts from time series input data. For Challenge 3, we propose two criteria to choose good prompts, which are used to select the most suitable source domain for domain adaptation (*i.e.*, prompt adaptation). Our contributions can be summarized as follows:

- **Propose a flexible prompt generator to learn domain-specific information.** We extend the idea of prompt tuning to time series analysis to capture

information specific to source domains. However, traditional prompts have limited flexibility in learning domain-specific information that evolves over time. To address this limitation, we introduce a conditional module that generates prompts parameterized by a neural network to capture domain-specific information. Theoretical analysis also demonstrates the superiority of our proposed prompt generator over traditional prompt tuning.

- **Develop two criteria for selecting good prompts.** We propose two criteria, fidelity and distinction, to ensure that prompts accurately capture domain-specific information from all source domains. Fidelity is achieved by maximizing the mutual information between prompts and labels, while distinction is achieved by minimizing the mutual information between prompts from different source domains. Theoretical guarantees establish that our generated prompts maintain fidelity and introduce new information.

- **Present an efficient algorithm with a robust architecture.** We introduce a simple yet effective optimization algorithm based on meta-learning to efficiently learn the objective. Additionally, we leverage the Mixture of Experts (MoE) technique to enhance the robustness of our proposed POND model.

- **Conduct comprehensive experiments on multiple benchmark datasets.** Extensive experiments across 50 scenarios on four benchmark datasets demonstrate the effectiveness and robustness of our proposed POND model. Experimental results indicate that our proposed POND model outperforms all state-of-the-art comparison methods by up to 66% on the F1-score.

3.2.2 Problem Setup

In this section, we mathematically formulate the multi-source time series domain adaptation problem. Important notations are shown in Table 3.3. Given M source

Table 3.3: Important notations and Descriptions.

Notations	Descriptions
S_i	The i-th source domain
T	Target domain
C	Class set
$(X_j^{(S_i)}, Y_j^{(S_i)})$	The j-th time series pair for S_i
$(X_j^{(T)}, Y_j^{(T)})$	The j-th time series pair for T
$Y^{(S_i)}, Y^{(T)}$	Label sets for S_i and T
P	Common prompt
$\Delta P^{(S_i)}$	Domain-level prompt for S_i
$\Delta P_j^{(S_i)}$	Instance-level prompt generated by $X_j^{(S_i)}$ for S_i

time series domains $S_i (i = 1, \cdots, M)$ and a target domain T, their j-th time series inputs are denoted as $X_j^{(S_i)} \sim p(X|Y_j^{(S_i)})$ and $X_j^{(T)} \sim p(X|Y_j^{(T)})$, respectively, where $Y_j^{(S_i)}$ and $Y_j^{(T)}$ are corresponding labels of $X_j^{(S_i)}$ and $X_j^{(T)}$, respectively. Here, $X_j^{(S_i)}, X_j^{(T)} \in \mathbb{R}^{n \times L}$, where n is the number of channels and L is the sequence length. The labels $Y_j^{(S_i)}, Y_j^{(T)} \in C = \{c_1, c_2, \cdots, c_K\}$, where $c_i (i = 1, \cdots, |C|)$ represents a label class, and the number of classes is $|C|$. $Y^{(S_i)} = \{Y_j^{(S_i)}\}$ and $Y^{(T)} = \{Y_j^{(T)}\}$ are the label sets for the source domain S_i and the target domain T, respectively. Sets $X^{(S_i)} = \{X_j^{(S_i)}\}$ and $X^{(T)} = \{X_j^{(T)}\}$ represent the input sets for the source domain S_i and the target domain T, respectively. We assume that the labeled time series of all source domains $S_i (i = 1, \cdots, M)$ are abundant, but the labeled time series are limited in the target domain T. Then the multi-source time series domain adaptation problem is formulated as follows:

Problem Formulation: Given the time series input sets $X^{(S_i)}$ and label sets $Y^{(S_i)} (i = 1, 2, \ldots, M)$ of M source domains, and the time series input set $X^{(T)}$ of the target domain T, the goal of the problem is to predict the label set $Y^{(T)}$ by learning the mapping F:

$$F : X_i^{(T)} \rightarrow Y_i^{(T)}.$$

Our problem formulation is very flexible: the time series input can be either univariate (*i.e.*, $N = 1$) or multivariate (*i.e.*, $N > 1$); the time series domain adaptation can be from a single source (*i.e.*, $M = 1$) or multiple sources (*i.e.*, $M > 1$); the classification problem can be either binary (*i.e.*, $K = 2$) or multi-class (*i.e.*, $K > 2$).

3.2.3 Prompt-based Domain Discrimination

In this section, we present our POND model to address the multi-source time series domain adaptation problem.

The Flexible Prompt Generator

The goal of this section is to explore methods for learning information that changes over time from different source domains for domain adaptation (*i.e.*, tackling Challenges 1 and 2). Most existing papers propose various strategies to extract domain-invariant representations from all source domains by making different domains indistinguishable [112, 76, 54, 151, 169]. However, this idea may discard domain-specific information from multiple source domains, which indicates which source domain is most similar to the target domain. To address this, a natural solution is to directly learn domain-specific information from the labeled time series pair $(X_j^{(S_i)}, Y_j^{(S_i)})$. This motivates us to utilize prompt tuning to learn domain-specific information, which was first introduced by the NLP community and demonstrated impressive success in many NLP tasks [18, 81, 70]. Compared with other domain adaptation techniques, prompt tuning has three advantages: firstly, prompts are adjusted via gradients by labeled data from multiple source domains, which offer domain-specific information; secondly, prompt tuning leverages small amounts of labeled data effectively for adaptation, which is suitable for the target domain with limited labeled data [70]; thirdly, prompts can be utilized as a heuristic to select the most similar source domain to the target domain for adaptation.

The prompt, which is extended from NLP to time series, is defined as a learnable vector that prepends to the time series input to learn domain-specific information by the labeled pair $(X_j^{(S_i)}, Y_j^{(S_i)})$. Mathematically, let $P^{(S_i)} \in \mathbb{R}^{n \times m}$ be the prompt of the source domain S_i, where m is the prompt length. Then, for the j-th time series input $X_j^{(S_i)}$, any time series model takes $[P^{(S_i)}, X_j^{(S_i)}]$ (*i.e.*, the concatenation of $P^{(S_i)}$ and $X_j^{(S_i)}$) as its model input. We decompose $P^{(S_i)}$ into two components:

$$P^{(S_i)} = P + \Delta P^{(S_i)}$$

where $P \in \mathbb{R}^{n \times m}$ is a common prompt to learn the common characteristics of all source domains, which can also be directly applied to the target domain T, and $\Delta P^{(S_i)} \in \mathbb{R}^{n \times m}$ is a prompt to learn domain-specific information (*i.e.*, information unique to the source domain S_i), which will be utilized to select the most similar source domain to the target domain T.

While the domain-specific prompt $\Delta P^{(S_i)}$ is potentially effective in learning domain-specific information about the source domain S_i (*i.e.*, address Challenge 1), it cannot directly address Challenge 2. This is because $\Delta P^{(S_i)}$ is time-independent and has little freedom to capture time-dependent domain-specific information (*e.g.*, distribution shifts of fiber-optic signals). To tackle this, instead of using a fixed prompt, we learn such domain-specific information by prompts generated from the time series input. This is because the time series input usually contains rich time-dependent information (*e.g.*, time series distributions and trends). Specifically, we introduce a conditional module $g^{(S_i)}$, parameterized by a neural network, to generate instance-level prompts based on time series instances:

$$\Delta P_j^{(S_i)} = g^{(S_i)}(X_j^{(S_i)}; \zeta) \in \mathbb{R}^{m \times n}$$

where $\Delta P_j^{(S_i)}$ is the instance-level prompt generated by the time series input $X_j^{(S_i)}$

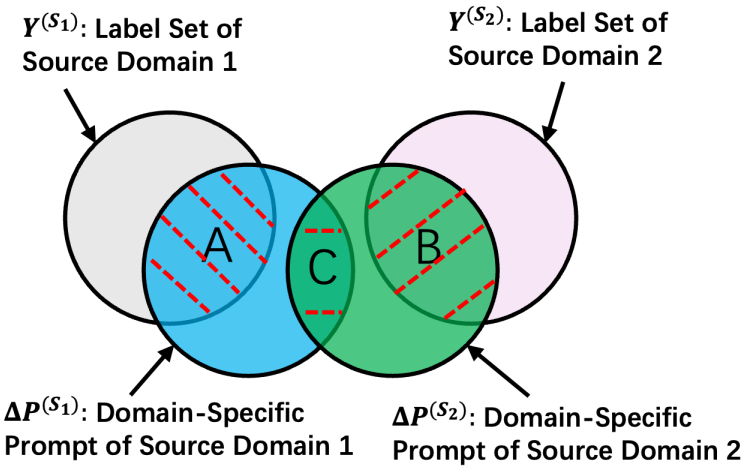


Figure 3.7: Illustration of two criteria: high fidelity and high distinction. Fidelity and distinction are represented as areas of $A + B$ and C, respectively.

and a random variable ζ, and the domain-level prompt $\Delta P^{(S_i)}$ is the aggregation of all instance-level prompts $\Delta P_j^{(S_i)}$ (*e.g.*, $\Delta P^{(S_i)} = \frac{1}{|S_i|}\sum_{j=1}^{|S_i|} \Delta P_j^{(S_i)}$). For any time series input $X_j^{(S_i)}$, its corresponding prompt is formulated as $P + \Delta P_j^{(S_i)}$.

Our proposed prompt generator $g^{(S_i)}$ conditionally generates instance-level prompts for specific time series inputs, which intuitively has more freedom of expression to learn domain-specific information than the traditional prompt tuning.

Two Important Criteria for Good Prompts

In the previous section, we extended prompt tuning to capture information on specific time series domains. While prompts are easy to recognize in computer vision and natural language fields, the learned prompts of time series data are not recognizable to humans, making it hard, if not impossible, to evaluate whether prompts are good enough to learn information for time series data. For example, a hard prompt consists of natural language that clearly describes the task at hand, explicitly asks the model for some result or action, and makes it easy to understand why the prompt elicited such behavior from the model [70]. In contrast, the learned prompts of specific time series domains are visualized as extra time segments, which are difficult to

understand by humans. Moreover, there is a lack of exploration on what constitutes a good prompt that captures domain-specific information without human-engineering priors. From our perspective, ideal prompts to capture domain-specific information should maintain high fidelity and high distinction, as illustrated in Figure 3.7: high fidelity suggests large overlaps between the learned domain-specific prompts and label information (*i.e.*, large $A + B$ in Figure 3.7), and high distinction implies small overlaps among domain-specific prompts of different source domains (*i.e.*, small C in Figure 3.7). They are introduced in details as follows:

High Fidelity. One important criterion for the prompt generator $g^{(S_i)}$ is fidelity (*i.e.*, the generated prompt $\Delta P_j^{(S_i)}$ preserves the domain-specific information of the source domain S_i). Motivated by the theory of information bottleneck [139], high fidelity is defined as the large mutual information between $\Delta P_j^{(S_i)}$ and $Y_j^{(S_i)}$, which should be maximized:

$$\max \sum_{i=1}^{M} \sum_{j=1}^{|S_i|} MI(\Delta P_j^{(S_i)}, Y_j^{(S_i)}), \tag{3.5}$$

where $MI(\bullet, \bullet)$ denotes the operator of mutual information. Based on the definition of mutual information, we have:

$$MI(\Delta P_j^{(S_i)}, Y_j^{(S_i)}) = H(Y_j^{(S_i)}) - H(Y_j^{(S_i)}|\Delta P_j^{(S_i)}),$$

where $H(Y_j^{(S_i)})$ represents the entropy of $Y_j^{(S_i)}$ and $H(Y_j^{(S_i)}|\Delta P_j^{(S_i)})$ is the entropy of $Y_j^{(S_i)}$ conditioned on $\Delta P_j^{(S_i)}$. Since $H(Y_j^{(S_i)})$ is constant, Equation (3.5) is equivalent to minimizing the conditional entropy $H(Y_j^{(S_i)}|\Delta P_j^{(S_i)})$, which can be expressed as:

$$\min \sum_{i=1}^{M} \sum_{j=1}^{|S_i|} H(Y_j^{(S_i)}|\Delta P_j^{(S_i)}).$$

Due to the computational complexity of the conditional entropy $H(Y_j^{(S_i)}|\Delta P_j^{(S_i)})$,

it can be approximated by the cross-entropy between $f([\Delta P_j^{(S_i)}, X_j^{(S_i)}])$ and $Y_j^{(S_i)}$ [161, 89], where $f([\Delta P_j^{(S_i)}, X_j^{(S_i)}])$ is the prediction obtained by concatenating $\Delta P_j^{(S_i)}$ and $X_j^{(S_i)}$ as an input to our proposed POND model. The fidelity loss is then expressed as:

$$\ell_F = \sum_{i=1}^{M} \sum_{j=1}^{|S_i|} Y_j^{(S_i)} \log f([\Delta P_j^{(S_i)}, X_j^{(S_i)}]). \tag{3.6}$$

Now, we theoretically show that the learned prompt $\Delta P_j^{(S_i)}$, which minimizes the fidelity loss (*i.e.*, Equation (3.6)), possesses the following properties:

Property 3.2.1 (Preserving Fidelity)**.** *If $\Delta P_j^{(S_i)}$ minimizes Equation* (3.6)*, the mutual information between $\Delta P_j^{(S_i)}$ and the label $Y_j^{(S_i)}$ is equivalent to that between the time series input $X_j^{(S_i)}$ and the label $Y_j^{(S_i)}$, i.e., $MI(\Delta P_j^{(S_i)}, Y_j^{(S_i)}) = MI(X_j^{(S_i)}, Y_j^{(S_i)})$.*

Property 3.2.2 (Adding New Information)**.** *By minimizing Equation* (3.6)*, the generated prompt $\Delta P_j^{(S_i)}$ contains new information compared to the time series input $X_j^{(S_i)}$, i.e., $H(\Delta P_j^{(S_i)}) \geq H(X_j^{(S_i)})$.*

For the formal proofs of this work, one may refer to the original paper. These properties demonstrate that minimizing Equation (3.6) ensures that the generated prompts will not decrease fidelity and may add new information to the time series input.

High Distinction. In addition to high fidelity, it is essential that the generated domain-specific prompt $\Delta P^{(S_i)}$ distinguishes the unique information of the source domain S_i from other source domains. This unique information not only aids in understanding the differences between multiple time series source domains but also provides valuable insights for selecting suitable sources for domain adaptation. To achieve this, from the perspective of information theory, we define the objective to

maintain high distinction as minimizing the mutual information of domain-specific prompts between different source domains, which should be minimized as follows:

$$\min \sum_{i_1 \neq i_2} MI(\Delta P^{(S_{i_1})}, \Delta P^{(S_{i_2})}), \tag{3.7}$$

where $\Delta P^{(S_{i_1})}$ and $\Delta P^{(S_{i_2})}$ represent the domain-specific prompts of any two source domains S_{i_1} and S_{i_2}. Equation (3.7) is computationally infeasible to minimize directly, but it can be achieved by minimizing the leave-one-out upper bound [113, 89]. Other mutual information upper bounds, such as the contrastive log-ratio bound [26], can also conveniently be incorporated into our framework. Therefore, the objective to encourage high distinction is formulated as minimizing the leave-one-out bound (*i.e.*, discrimination loss):

$$\ell_D = \sum_{i_1 \neq i_2} \mathbb{E} \log \frac{\exp(\text{sim}(\Delta P^{(S_{i_1})}, \Delta P^{(S_{i_2})}))}{\sum_{i \neq i_1, i \neq i_2} \exp(\text{sim}(\Delta P^{(S_{i_1})}, \Delta P^{(S_i)}))}, \tag{3.8}$$

where $\text{sim}(\Delta P^{(S_{i_1})}, \Delta P^{(S_{i_2})}) = tr((\Delta P^{(S_{i_1})})^T \Delta P^{(S_{i_2})})$ denotes the inner product of the two domain-specific prompts $\Delta P^{(S_{i_1})}$ and $\Delta P^{(S_{i_2})}$, and $tr(A)$ represents the trace of any matrix A.

The Learning Objective

After introducing two criteria for selecting good prompts, we present our learning objective in this section.

Combining the fidelity loss ℓ_F in Equation (3.6) and the discrimination loss ℓ_D in Equation (3.8), our learning objective is expressed as follows:

$$\min_{P, g^{(S_i)}} G(P, g^{(S_i)}) = \ell_R + \lambda_1 \ell_D + \lambda_2 \ell_F, \tag{3.9}$$

where $\ell_R = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} R(f([P + \Delta P_j^{(S_i)}, X_j^{(S_i)}]), Y_j^{(S_i)})$ is the training loss that

measures the performance of prompt tuning. Here, $R(\cdot,\cdot)$ is the loss function, and $[P+\Delta P_j^{(S_i)}, X_j^{(S_i)}]$ is the concatenation of the overall prompt $P+\Delta P_j^{(S_i)}$ and the time series input $X_j^{(S_i)}$. Two tuning parameters $\lambda_1, \lambda_2 > 0$ control the trade-off among the training loss, the fidelity loss, and the discrimination loss.

To optimize Equation (3.9), we need to enumerate all source domains, which may be inefficient and unscalable [89]. To address this, we propose a simple yet effective learning algorithm based on the classic Reptile meta-learning framework [105], which randomly picks a source domain each time and conducts standard steps of gradient descent without the need for calculating second derivatives.

After learning the common prompt P and the prompt generator $g^{(S_i)}$, they can be utilized for target domain transfer. Specifically, the prompt generator $g^{(T)}$ is optimized by the labeled time series pairs $(X_i^{(T)}, Y_i^{(T)})$ in the target domain T as follows:

$$\min_{g^{(T)}} \frac{1}{|T|} \sum_{i=1}^{|T|} R(f([P + \Delta P_i^{(T)}, X_i^{(T)}]), Y_i^{(T)}), \tag{3.10}$$

where $\Delta P_i^{(T)} = g^{(T)}(X_i^{(T)}) \in \mathbb{R}^{m \times n}$ is the instance-level domain-specific prompt of the time series input $X_i^{(T)}$, and the domain-level domain-specific prompt of the target domain T is $\Delta P^{(T)} = \frac{1}{|T|} \sum_{j=1}^{|T|} \Delta P_j^{(T)}$. However, $g^{(T)}$ may not be reliable for prediction due to the limited labeled data involved. To handle this, $\Delta P^{(T)}$ is utilized as a heuristic to find the most similar source domain by the simple nearest neighbor rule (*i.e.*, prompt adaptation):

$$S_i = \arg\max_{S_i} \text{sim}(\Delta P^{(S_i)}, \Delta P^{(T)}), \tag{3.11}$$

where $\text{sim}(\Delta P^{(S_i)}, \Delta P^{(T)})$ is a similarity function (*e.g.*, cosine similarity) between the domain-specific prompts $\Delta P^{(S_i)}$ and $\Delta P^{(T)}$. Then, we utilize the prompt generator $g^{(S_i)}$ for prediction in the target domain T: $f([P + g^{(S_i)}(X_j^{(T)}), X_j^{(T)}])$.

Table 3.4: Statistics of four datasets.

Dataset	# Domain	# Channel	# Class	Seq Len	# Train	# Test
HAR	30	9	6	128	2300	990
WISDM	36	3	6	128	1350	720
HHAR	9	3	6	128	12716	5218
SSC	20	1	5	3000	14280	6130

3.2.4 Experiments

In this section, we employ four benchmark datasets to evaluate our proposed POND model in comparison with six state-of-the-art methods. All experiments were conducted on a Linux server equipped with an Intel(R) Xeon(R) Silver 4214 CPU and an NVIDIA GPU running version 510. More experiments are included in the supplementary materials.

Experimental Settings

Benchmark Dataset: We evaluated the performance of all methods on four benchmark datasets, HAR, WISDM, HHAR and SSC [115]. The statistics of all benchmark datasets are shown in Table 3.4, which are introduced as follows:

1. HAR [3]: The Human Activity Recognition (HAR) dataset incorporates data collected from three sensors—accelerometer, gyroscope, and body sensors—deployed on 30 subjects (*i.e.*, domains) engaged in six distinct activities.

2. WISDM [63]: The WIreless Sensor Data Mining (WISDM) dataset, using accelerometer sensors, involves 36 subjects participating in activities similar to the HAR dataset, with additional challenges due to class distribution imbalances among different subjects.

3. HHAR [129]: The Heterogeneity Human Activity Recognition (HHAR) dataset was collected from 9 subjects using sensor readings from smartphones and smartwatches.

4. SSC [42]: The Sleep Stage Classification (SSC) problem aims to categorize

electroencephalography (EEG) signals into five stages. We utilize the Sleep-EDF dataset [42], including EEG recordings from 20 healthy subjects.

Comparison Methods: We compared our proposed POND method with six state-of-the-art time series domain adaptation approaches: Raincoat [46], CoDATs [151], Deep Coral [131], MMDA [116], DIRT-T [126] and DSAN [174]. All comparison methods are introduced as follows:

1. Raincoat [46]: it is an unsupervised domain adaptation method addressing both feature and label shifts.

2. CoDATs [151]: it is the first method to handle multi-source domain adaptation through adversarial training with weak supervision.

3. Deep Coral [131]: it minimizes domain shift by aligning second-order statistics of source and target distributions.

4. MMDA [116]: it integrates Maximum Mean Discrepancy (MMD) and CORrelation ALignment (CORAL) along with conditional entropy minimization to address domain shift.

5. DIRT-T [126]: it utilizes adversarial training, conditional entropy, and a teacher model to align source and target domains.

6. DSAN [174]: it minimizes the discrepancy between source and target domains via a Local Maximum Mean Discrepancy (LMMD) that aligns relevant subdomain distributions.

Metrics: Two performance metrics were employed: Macro-F1 score and Accuracy. Macro-F1 is the unweighted mean of per-class F1 scores, treating all classes equally. Accuracy is the ratio of accurately predicted samples to all samples.

Hyperparameter Settings: We adapted the setting of supervised domain adaptation, where ten samples in the target domain were used for domain transfer. All source-target scenarios were selected randomly to ensure the fairness of the performance evaluation. Single-source domain adaptation methods (e.g. Raincoat) were

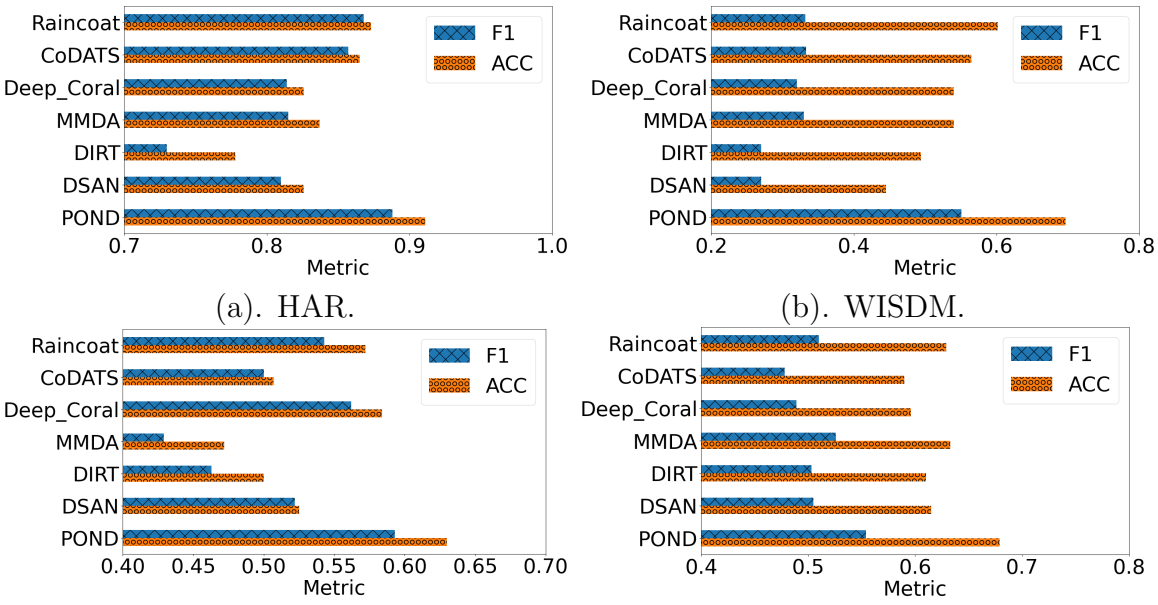
trained by combining all source domains. For the training set of all time series source domains, 60% was used for pretraining our POND model, 20% for prompt tuning, and 20% for validation sets. The batch size was set to 16. The number of global steps N, global learning rate δ and the local learning rate η were set to 50, 0.01 and 0.001, respectively. The number of experts was set to three. The prompt generator is a two-layer Multi-Layer Perceptron (MLP) with Tanh activation. For the transformer model, the numbers of encoder layers, decoder layers, and heads in the multi-head attention were set to 2, 1, and 4, respectively. The dimensions of the multi-head attention and the feed-forward layer were set to 16 and 128, respectively. The hyper-parameters λ_1 and λ_2 were chosen based on performance on the validation set. λ_1 and λ_2, along with other hyperparameters such as the number of epochs, are provided in Table 3.5. All methods were averaged by ten times.

Table 3.5: Hyperparameters of all datasets.

Dataset	#Epochs	Prompt Length	λ_1	λ_2
HAR	50	5	1	1
WISDM	200	3	1	1
HHAR	200	5	1	1
SSC	100	10	0.1	0.1

3.2.5 Experimental Results

Performance Evaluation: We conducted a comprehensive performance evaluation to test all methods across approximately 50 scenarios on four datasets. Figure 3.8 displays the F1-score and accuracy of all methods on these datasets. Our proposed POND method consistently outperforms others across all four datasets. Specifically, on the HAR dataset, the F1-score of POND is approximately 0.9, only 2% lower than the top-performing comparison method, Raincoat. The F1-score gaps on the HHAR, and SSC datasets are 5% and 4.4%, respectively. The largest gap is observed in the WISDM dataset, where the F1-score and accuracy of POND hover around 0.6 and 0.7, while all comparison methods score below 0.35 and 0.6, respectively. Considering



(a). HAR. (b). WISDM.

(c). HHAR. (d). SSC.

Figure 3.8: The F1-score and accuracy of all methods on four benchmark datasets: the proposed POND outperforms comparison methods consistently.

the inherent difficulty of training on the WISDM dataset due to class imbalance, this highlights the effectiveness of our proposed POND, especially on challenging datasets.

Among the comparison methods, Raincoat emerges as the best overall. In terms of F1-score, Raincoat outperforms MMDA by 5% on the HAR dataset and shows an 8% superiority over CoDATs on the HHAR dataset. For accuracy, Raincoat performs 7% better than DIRT on the HHAR dataset and surpasses Deep Coral by 3% on the SSC dataset. CoDATs and Deep Coral also demonstrate competitive performance, achieving around 55% accuracy on the WISDM dataset, while DSAN lags behind at 45%. On the other hand, MMDA, DIRT, and DSAN exhibit varying performance across datasets. For instance, DSAN performs comparably to Raincoat on the SSC dataset but ranks the lowest on the WISDM dataset.

Table 3.6 presents the performance of all methods across various scenarios in four datasets, including the upper bound achieved by training and testing on the target domain. The reported values include means and standard deviations from ten imple-

Table 3.6: F1-score on different scenarios of four datasets: the proposed POND model outperforms all comparison methods.

Scenario	Raincoat	CoDATs	Deep_Coral	DIRT	DSAN	POND	Target Only
HAR 1-15 → 16	0.823 ± 0.094	0.767 ± 0.093	0.773 ± 0.082	0.612 ± 0.135	0.738 ± 0.095	**0.849 ± 0.021**	0.856 ± 0.027
HAR 1-15 → 20	0.872 ± 0.142	0.932 ± 0.025	0.923 ± 0.023	0.848 ± 0.101	0.929 ± 0.033	**0.968 ± 0.021**	0.983 ± 0.018
HAR 1-15 → 21	0.867 ± 0.141	0.903 ± 0.070	0.882 ± 0.028	0.921 ± 0.090	0.909 ± 0.110	0.972 ± 0.021	1.000 ± 0.000
HAR 1-15 → 28	0.766 ± 0.107	0.775 ± 0.166	**0.852 ± 0.044**	0.671 ± 0.175	0.783 ± 0.046	0.829 ± 0.018	0.853 ± 0.019
HAR 16-20 → 1	0.792 ± 0.072	0.744 ± 0.053	0.667 ± 0.077	0.546 ± 0.060	0.698 ± 0.037	**0.883 ± 0.017**	0.986 ± 0.010
HAR 16-20 → 2	0.825 ± 0.048	0.821 ± 0.151	0.796 ± 0.055	0.509 ± 0.050	0.652 ± 0.057	**0.936 ± 0.017**	0.943 ± 0.024
HAR 16-20 → 3	0.814 ± 0.028	0.746 ± 0.078	0.741 ± 0.058	0.605 ± 0.056	0.565 ± 0.043	**0.878 ± 0.018**	0.978 ±0.013
HAR 16-20 → 4	0.679 ± 0.084	0.605 ± 0.082	0.479 ± 0.110	0.336 ± 0.110	0.436 ± 0.032	**0.754 ± 0.033**	0.921 ± 0.018
WISDM 0-17 → 18	0.379 ± 0.061	0.384 ± 0.049	0.346 ± 0.023	0.300 ± 0.041	0.287 ± 0.045	**0.606 ± 0.020**	0.705 ± 0.046
WISDM 0-17 → 20	0.354 ± 0.040	0.368 ± 0.039	0.376 ± 0.031	0.347 ± 0.071	0.269 ± 0.064	**0.570 ± 0.023**	0.704 ± 0.051
WISDM 0-17 → 21	0.355 ± 0.057	0.310 ± 0.088	0.259 ± 0.018	0.276 ± 0.055	0.245 ± 0.046	**0.450 ± 0.026**	0.636 ± 0.095
WISDM 0-17 → 23	0.306 ± 0.015	0.327 ± 0.075	0.318 ± 0.031	0.271 ± 0.016	0.277 ± 0.044	**0.482 ± 0.017**	0.538 ± 0.034
WISDM 0-17 → 25	0.365 ± 0.030	0.540 ± 0.125	0.435 ± 0.043	0.314 ± 0.107	0.353 ± 0.120	**0.559 ± 0.050**	0.672 ± 0.039
WISDM 0-17 → 28	0.399 ± 0.028	0.431 ± 0.033	0.418 ± 0.032	0.304 ± 0.044	0.339 ± 0.030	**0.656 ± 0.046**	0.689 ± 0.048
WISDM 0-17 → 30	0.314 ± 0.020	0.305 ± 0.028	0.298 ± 0.023	0.266 ± 0.035	0.246 ± 0.076	**0.670 ± 0.039**	0.791 ± 0.028
WISDM 18-23 → 5	0.648 ± 0.001	0.558 ± 0.129	0.534 ± 0.102	0.549 ± 0.097	0.484 ± 0.055	**0.652 ± 0.035**	0.734 ± 0.095
WISDM 18-23 → 6	0.544 ± 0.074	0.565 ± 0.143	0.437 ± 0.078	0.405 ± 0.089	0.454 ± 0.112	**0.628 ± 0.033**	0.872 ± 0.049
WISDM 18-23 → 7	0.588 ± 0.070	0.404 ± 0.117	0.530 ± 0.094	0.518 ± 0.120	0.476 ± 0.127	**0.672 ± 0.029**	0.888 ± 0.035
HHAR 0-6 → 7	0.765 ± 0.142	0.652 ± 0.108	0.815 ± 0.105	0.649 ± 0.005	0.730 ± 0.164	**0.834 ± 0.014**	0.861 ± 0.016
HHAR 5-8 → 2	0.321 ± 0.023	0.347 ± 0.082	0.309 ± 0.032	0.276 ± 0.021	0.314 ± 0.095	**0.352 ± 0.014**	0.881 ± 0.018
SSC 0-9 → 16	0.578 ± 0.028	0.510 ± 0.044	0.537 ± 0.024	0.523 ± 0.019	0.515 ± 0.044	**0.568 ± 0.012**	0.601 ± 0.018
SSC 0-9 → 17	0.511 ± 0.024	0.413 ± 0.118	0.452 ± 0.077	0.530 ± 0.053	0.463 ± 0.081	**0.559 ± 0.006**	0.602 ± 0.014
SSC 0-9 → 18	**0.605 ± 0.016**	0.548 ± 0.037	0.544 ± 0.046	0.574 ± 0.021	0.569 ± 0.046	0.604 ± 0.014	0.602 ± 0.013
SSC 0-9 → 19	0.562 ± 0.024	0.540 ± 0.052	0.531 ± 0.055	0.565 ± 0.028	0.568 ± 0.080	**0.570 ± 0.010**	0.613 ± 0.019
SSC 10-12 → 8	0.294 ± 0.028	0.380 ± 0.066	0.379 ± 0.076	0.322 ± 0.048	0.411 ± 0.046	**0.470 ± 0.010**	0.531 ± 0.019

mentations, with the best results highlighted in bold. Overall, our proposed POND model consistently outperforms all methods, aligning with the observations in Figure 3.8. Notably, POND exhibits superior performance on the challenging WISDM dataset, as indicated by Figure 3.8. For instance, POND outperforms all comparison methods by at least 23% when transferring from domains 0-17 to domain 18. While POND excels overall, there are instances where comparison methods outperform it. For example, Deep Coral performs better than POND by 2% when transferring domains 1-15 to domain 28 on the HAR dataset, and MMDA marginally outperforms POND when transferring domains 1-15 to domain 21 on the HAR dataset.

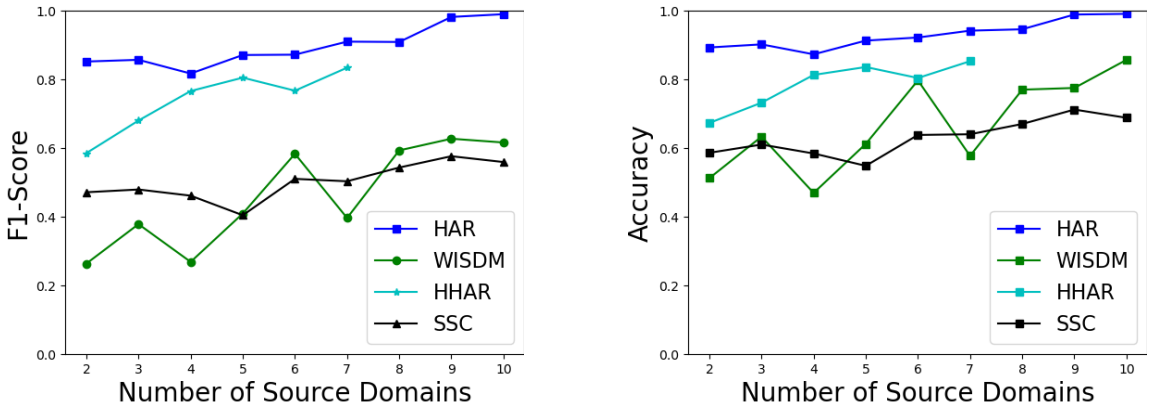
In addition to superior performance, our proposed POND model demonstrates greater stability compared to all comparison methods, as indicated by lower standard deviations. For instance, the standard deviation of POND is 0.006 when transferring domains 0-9 to domain 17 on the SSC dataset, while the standard deviations of all comparison methods range between 0.024 and 0.118, being at least 3 times larger than that of POND. Importantly, POND achieves results close to the upper bound in many scenarios, such as "HAR 1-15 → 16", "SSC 0-9 → 18", and "HHAR 0-6 → 7".

Table 3.7: Ablation study on the WISDM dataset: all components of our proposed POND model contribute to the outstanding performance.

MoE	0-17→ 22	0-17→ 23	0-17→ 24	0-17→ 25	18-23→ 5	18-23→ 6	Overall
✗	0.622±0.057	0.415±0.015	0.510±0.030	0.581±0.036	0.623±0.058	0.516±0.038	0.545±0.039
✗	0.646±0.064	0.396±0.048	0.527±0.030	0.573±0.034	0.628±0.051	0.512±0.057	0.547±0.047
✗	0.632±0.069	0.384±0.041	0.498±0.032	0.572±0.045	0.611±0.055	0.514±0.025	0.535±0.045
✓	0.575±0.043	0.349±0.029	0.517±0.032	0.584±0.030	0.621±0.056	0.578±0.035	0.537±0.038
✓	0.719±0.062	0.405±0.052	0.529±0.042	0.588±0.034	0.616±0.050	0.565±0.049	0.570±0.048
✓	**0.725±0.031**	**0.482±0.017**	**0.559±0.050**	**0.695±0.035**	**0.652±0.035**	**0.628±0.033**	**0.624±0.034**

Ablation Study: Next, we demonstrate the ablation study of the proposed POND method, whose goal is to identify whether all components of our proposed POND model contribute to the performance. Specifically, we explore the necessity of the MoE technique, common prompt, and prompt generator. The challenging WISDM dataset was utilized to test the performance. Table 3.7 illustrates the performance of different scenarios, all of which were averaged by 10 times. The first two rows show the performance with the common prompt, and the prompt generator available only, respectively. The fourth to sixth rows demonstrate the performance without the MoE, common prompt, and prompt generator, respectively, and the last row shows the performance of the complete POND model. Overall, our proposed POND model performs best when the MoE, common prompt, and prompt generator are all available, which suggests that all components are necessary for the outstanding performance of our proposed POND model. For example, in the scenario of "18-23→ 6", the best performance without any component only achieves a performance no more than 0.58, whereas that of the complete POND model is 5% better. The gap is widened to 7% for the scenario "0-17→ 25".

Sensitivity Analysis: In this section, we explore how source domains influence performance on the target domain. Figure 3.9 illustrates the relationship between performance metrics (F1-score and accuracy) and the number of source domains, averaged over 10 implementations. Generally, our proposed POND model demonstrates improved performance with an increasing number of source domains. For instance, POND achieves 50% accuracy with two source domains for training, but this figure

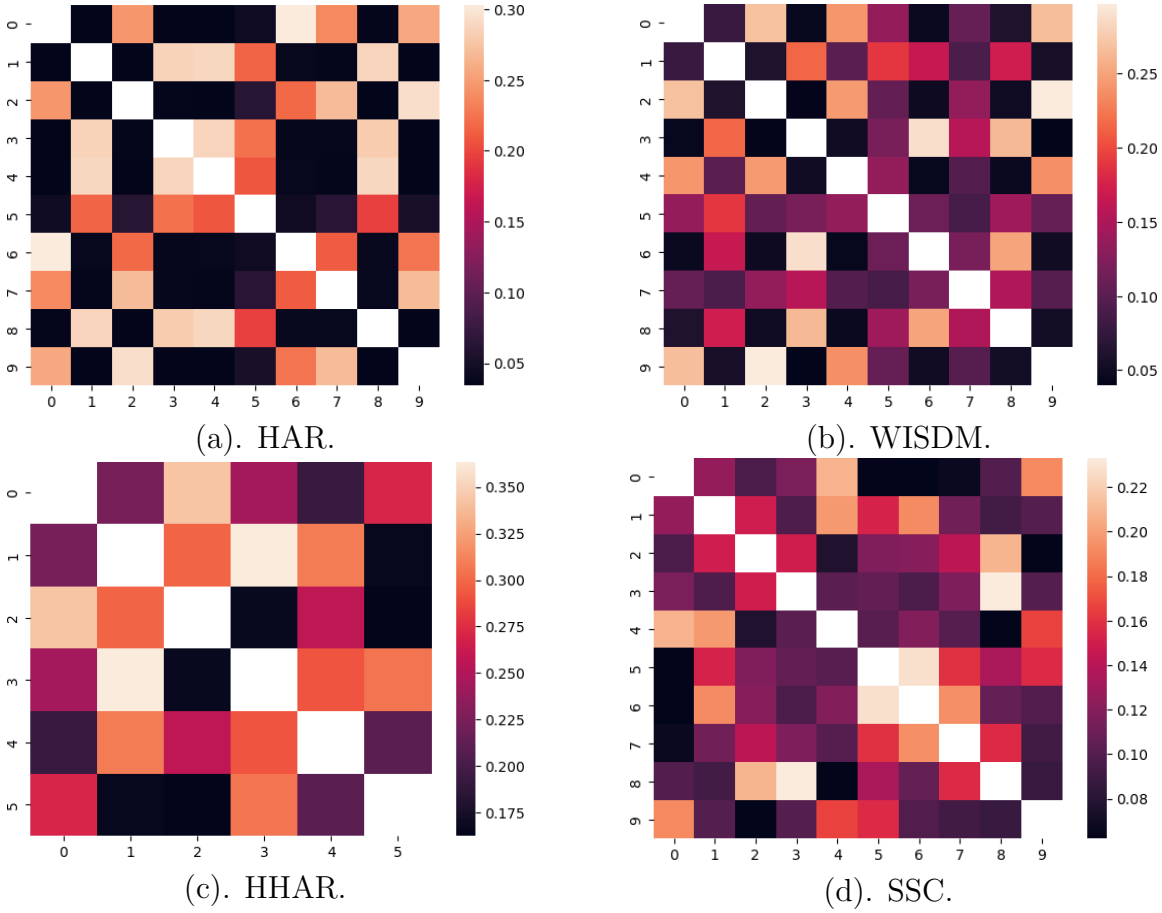


(a). F1-score. (b). Accuracy.

Figure 3.9: The F1-score and accuracy of the proposed POND model with different source domains: the performance grows with the increase of source domains. (The HHAR dataset has less than 10 domains.)

rises by 30% when an additional 8 source domains are included. Similarly, the F1-score of POND increases by 20% when the number of source domains changes from 2 to 6. However, some exceptions exist. For example, there is a notable 25% drop in F1-score when increasing the number of source domains from 6 to 7 on the WISDM dataset. Another instance involves a 5% performance drop when increasing the source domains from 4 to 5 on the SSC dataset.

Visualization of Discrimination Loss: Finally, we present a visualization of the discrimination loss ℓ_D for pairwise source domains. Figure 3.10 illustrates the exponents of discrimination losses for all pairs of source domains across four datasets. Both the X-axis and Y-axis represent the indices of source domains. Darker colors indicate smaller discrimination losses, reflecting better domain discrimination. The diagonals are left blank. Overall, our proposed POND model effectively discriminates most source domains, as evidenced by the predominance of dark squares. For instance, domains 3-5 and domains 6-7 exhibit clear discrimination with losses below 0.05. Similar effective discrimination is observed for domain pairs 6 and 0 on the WISDM dataset, domain pairs 1 and 5 on the HHAR dataset, and domains 5-7 and 0 in the SSC dataset. However, discrimination losses for some domain pairs are larger than



(a). HAR. (b). WISDM.

(c). HHAR. (d). SSC.

Figure 3.10: The visualization of the exponent of discrimination loss: most pairs of source domains are well discriminated.

others. For instance, on the HAR dataset, the discrimination loss between domains 0 and 6 is the largest, approximately 0.30, but still within an acceptable range. It's worth noting that domain discrimination may not adhere to the transitive property. For example, domains 3 and 9, as well as domains 4 and 9, are well-discriminated, but domains 3 and 4 are relatively poorly-discriminated.

3.3 Conclusion

This chapter introduced dynamic mapping methods for functional-space guided learning, with a focus on the DRAIN framework. By bridging parameter space and functional space through a novel encoder-decoder mechanism, DRAIN addresses key challenges in continual learning, including the need for adaptability, generalizability, and efficient knowledge retention.

The encoder-decoder approach in DRAIN enables bidirectional mapping between parameter and functional spaces, facilitating the encoding of task-specific knowledge into a shared functional representation and its subsequent retrieval for learning. This dynamic mapping enhances the model's ability to generalize across tasks, adapt to new information, and provide insights into how functional representations evolve over time.

By formalizing the interaction between parameter and functional spaces, DRAIN sets the stage for a deeper understanding of how functional-space principles can be applied to address dynamic and non-stationary learning scenarios. The framework highlights the potential of functional-space guided learning to improve the efficiency and robustness of machine learning systems, serving as a stepping stone toward broader applications.

The next chapter explores these applications, showcasing how functional-space guided learning principles can address practical challenges in efficiency and domain

adaptation.

Chapter 4

Harnessing Function Space of Machine Learning Models for Better Efficiency via Global Pruning

Optimizing the efficiency of large-scale machine learning models is a critical step toward making advanced AI systems scalable and accessible. This chapter explores a cutting-edge application of function space optimization: SparseLLM, a globally optimized pruning strategy for large language models (LLMs). SparseLLM addresses the challenges of computational and memory constraints by decomposing the global pruning problem into smaller, manageable subproblems, enabling the efficient and scalable pruning of billion-parameter LLMs.

This chapter begins by examining the limitations of traditional pruning techniques, which either focus on layer-wise pruning or require large computational resources for global pruning. SparseLLM bridges this gap by leveraging auxiliary variables and a flexible decomposition strategy to balance computational efficiency with

global optimality. The resulting pruning framework not only improves the scalability of pruning methods but also demonstrates superior performance in high-sparsity regimes compared to state-of-the-art approaches.

The following sections detail the mathematical foundation, algorithmic design, and empirical evaluation of SparseLLM, showcasing its adaptability to different model architectures and its ability to significantly reduce perplexity while maintaining competitive zero-shot performance. SparseLLM's versatility makes it a valuable tool for researchers and practitioners aiming to optimize LLMs for real-world applications.

4.1 Global Pruning of Pre-trained Language Models

4.1.1 Introduction

Large language models (LLMs) [141, 106] have recently transformed the field of natural language processing (NLP) by delivering exceptional results across a variety of intricate language benchmarks [150, 17, 20]. Nonetheless, these models, with billions of parameters, generally necessitate significant computational resources. To make LLMs more accessible, extensive efforts have been devoted to model compression of LLMs [155, 8], including pruning, quantization, knowledge distillation, and low-rank factorization. *Pruning*, by introducing *sparsity*, jointly enhances memory and computational efficiency and offers unparalleled flexibility, seamlessly integrating with any LLMs, thus standing out as a highly effective and widely adopted compression strategy.

Model pruning has a long history [66] and has proven effective in applications related to vision and smaller language models [47]. However, conventional pruning techniques, which rely on global pruning and require loading the entire model into

the same GPU [92, 128], become impractical for today's LLMs due to their vast size. Recently, several *local pruning* methods have been proposed for billion-scale LLMs. These methods compress each layer separately, and the overall compressed model is then obtained by "stitching together" the individually compressed layers. SparseGPT [39], an efficient unstructured pruning method for LLMs with hundreds of billions of parameters, achieves parameter reduction of up to 60% with minimal performance loss. Another approach, Wanda [133], introduces a novel pruning criterion that evaluates weights by considering both magnitude and related input activations. Despite its efficiency gains, local pruning only aims to minimize the local error for each specific layer under sparsity constraints, resulting in a *suboptimal* solution for the overall model. This is because local pruning *over-aligns* the intermediate layers' activations, leading to suboptimal performance, especially in high-sparsity regimes [128, 135].

To address these challenges and achieve global pruning with low memory consumption, we propose *SparseLLM* that decomposes the global pruning objective into multiple subproblems, each of which can be solved with low resources and coordinate to achieve the global pruning objective. More specifically, we first formulate LLMs as a composite function where the output of one module is the input of the next. Based on this formulation, we reformulate the global pruning goal into an equivalent form with auxiliary variables that facilitate its decomposition and coordination of the subproblems. Then we propose an alternating optimization algorithm to efficiently solve the subproblems, achieving computational

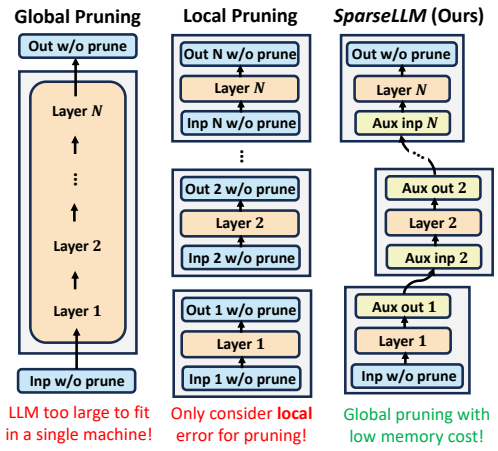


Figure 4.1: *SparseLLM* decomposes the global pruning of LLMs into manageable subproblems by leveraging the chain of modules and auxiliary variables while maintaining dependencies.

resource efficiency and global optimality, due to the close-form solution of each sub-problem. Empirically, we find that *SparseLLM* can consistently improve the performance of local pruning methods, particularly in high sparsity regimes (¿ 60%), where the perplexity can be significantly decreased by up to around 80% as compared to the state-of-the-art methods.

Furthermore, our SparseLLM framework can be readily applicable to enhance the performance of most existing local pruning solvers, such as SparseGPT and Wanda, with marginal additional computational overhead. This adaptability ensures that our framework can be seamlessly integrated into a wide range of LLMs and pruning methods, making it a versatile tool and useful baseline for future research exploiting the sparsity of LLMs.

4.1.2 Background and notation

Global pruning

Given a pre-trained neural network f with parameter $\mathbf{W}$ and inputs $\mathbf{X}$, global pruning aims to find a global sparsity mask $\mathbf{M}$ and possibly updated weights $\widehat{\mathbf{W}}$ to minimize the *global loss* $\mathcal{L}$ between the final outputs of the uncompressed and compressed model:

$$\min_{\mathbf{M},\widehat{\mathbf{W}}} \ \mathcal{L}\big(f(\mathbf{X};\mathbf{M}\odot\widehat{\mathbf{W}}), f(\mathbf{X};\mathbf{W})\big), \tag{4.1}$$

where $\odot$ denotes the *element-wise* multiplication. In addition to NP-hardness [16], however, a critical challenge in solving Eq. 4.1 is the huge memory cost, as one needs to store the entire model in a single GPU, rendering this method impractical for modern billion-scale LLMs.

Local pruning

Local pruning circumvents the memory issue mentioned above by dividing the full model compression into subproblems for each layer and constructing a *local loss* to measure the ℓ_2-error between the outputs of the uncompressed and compressed layers. Hence, the local pruning can be formulated by

$$\min_{\mathbf{M}_\ell, \widehat{\mathbf{W}}_\ell} \|\mathbf{W}_\ell \cdot \mathbf{X}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell) \cdot \mathbf{X}_\ell\|_2^2. \tag{4.2}$$

Although smaller than the global pruning, the local pruning still needs to optimize both the mask $\mathbf{M}_\ell$ and the remaining weights $\widehat{\mathbf{W}}_\ell$ and thus remains NP-hard. Therefore, exactly solving it for larger layers is unrealistic, leading all existing methods to resort to approximations.

Mask selection & weight reconstruction. A particularly popular approach is to separate the problem into *mask selection* and *weight reconstruction* [51, 64]. Concretely, this means first choosing a pruning mask $\mathbf{M}$ according to some salient criterion, like the weight magnitude [172], and then optimizing the remaining unpruned weights while keeping the mask unchanged. Importantly, once the mask is fixed, Eq. 4.2 turns into a *linear regression* problem that can be easily optimized.

Existing solvers. Early work [56] applied iterated linear regression to small networks. Recently, the AdaPrune approach [51] has shown good results for this problem on modern models via magnitude-based weight selection, followed by applying SGD steps to reconstruct the remaining weights. Follow-up works demonstrate that pruning accuracy can be further improved by removing the strict separation between mask selection and weight reconstruction. More recently, [39] developed SparseGPT, an efficient unstructured pruning method for LLMs with hundreds of billions of parameters, achieving up to 60% parameter reduction with minimal performance loss. [133] in-

troduced a novel pruning criterion in Wanda, which evaluates weights by considering both magnitude and related input activations.

What is wrong with local pruning?

As shown in Eq. 4.14, local pruning focuses on minimizing the error for each specific layer ℓ subject to sparsity constraints. This results in a suboptimal solution with respect to the global pruning problem. While the primary goal of pruning is to ensure that the input and output of the pruned model align closely with those of the original models, the local pruning overly constrains the activations of all the intermediate layers between the two models, leading to performance degradation.

4.1.3 Proposed Method

We present our proposed method SparseLLM, that can address the drawbacks of existing pruning methods by achieving a global pruning with low memory consumption. SparseLLM decomposes the global pruning objective into many subproblems, each of which can be solved using low resources and can coordinate with each other toward the global pruning objective. An overview of SparseLLM on the OPT and LlaMA configurations are shown in Figure 4.2.

Motivation

The development of SparseLLM is motivated by the observation: LLMs can be formulated as a composite function such that the output of one module is the input of the next. This allows us to reformulate the global pruning goal into its equivalent form with auxiliary variables that enable the decomposition into multiple subproblems, as detailed in Sec. 4.1.3. Then we develop a resource-efficient algorithm that achieves the alternating optimization of the subproblems with global optimality, thanks to the close-form solution of each subproblem, as illustrated in Sec. 4.1.4.

A unified formulation of pruning

In this section, we present the reformulation of the global pruning problem into an equivalent one by introducing auxiliary variables. This reformulation provides a more flexible form and enables the decomposition of the problem into many manageable subproblems.

The key idea behind our formulation is to decouple the densely parametric parts (linear layers) from non-parametric parts (activation function, self-attention, layer norm, etc) using a splitting technique. Rather than feeding the output of the dense linear layer $\mathbf{W}_\ell$ directly into the non-parametric and potentially nonlinear layer ϕ_ℓ, we store the output of layer ℓ in a new variable $\mathbf{z}_\ell = \mathbf{W}_\ell \boldsymbol{a}_{\ell-1}$ [1]. We also represent the output of the non-parametric layer as a vector of activations $\boldsymbol{a}_\ell = \phi_\ell(\mathbf{z}_\ell)$. We then solve the following problem:

$$\begin{aligned} \min_{\{\widehat{\mathbf{W}}_\ell\},\{\mathbf{M}_\ell\},\{\boldsymbol{a}_\ell\},\{\mathbf{z}_\ell\}} \quad & \mathcal{L}(\mathbf{z}_L, \mathbf{y}), \\ \text{s.t.} \quad & \mathbf{z}_\ell = (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}, \ \forall\, \ell \in [L], \\ & \boldsymbol{a}_\ell = \phi_\ell(\mathbf{z}_\ell), \ \forall\, \ell \in \Omega, \\ & \boldsymbol{a}_\ell, \mathbf{z}_\ell = \boldsymbol{a}_\ell^{pre}, \mathbf{z}_\ell^{pre}, \ \forall\, \ell \in [L-1]\backslash\Omega, \end{aligned} \tag{4.3}$$

where L represents the total number of dense (linear) layers and $[L] = \{1, 2, \cdots, L\}$. $[L-1]\backslash\Omega$ denotes the complement set of Ω. We use $\boldsymbol{a}_\ell^{pre}$, $\mathbf{z}_\ell^{pre}$ to denote the corresponding intermediate variables' values of the original dense (i.e., *without* pruning) pre-trained model. y denotes the ground-truth final output of the dense pre-trained model.

In our proposed formulation above, its unified nature lies in the interpretation and application of the set Ω, which denotes the indices of layers subject to the pruning

[1] For the sake of simplicity and clearer presentation, the bias term is omitted in the following equations where its exclusion does not lead to confusion.

process. Intuitively, Ω measures how "global" the pruning is. The bigger the set of Ω is, the more layers are connected via the second constraint, and the pruning is more towards the global extreme, and vice versa. The generality and versatility of our formulation is illustrated in the following remark:

Remark 4.1.1 (Generality and flexibility of Eq. 4.3)**.** *Given an LLM formulated as a composite function with dense layers $l \in \{1, 2, \ldots, L-1\}$, where L is the total number of dense layers and Ω denotes the set of layers subject to the pruning process. Our formulation can seamlessly treat both global and local pruning as special cases under certain conditions. Specifically:*

- *When $\Omega = \{1, 2, \ldots, L-1\}$, solving our pruning formulation is equivalent to global pruning, accounting for inter-layer dependencies across the entire network.*
- *When $\Omega = \emptyset$, the formulation simplifies to local pruning, considering each layer independently (the last constraint dominates and "cuts" all layer dependencies with pre-trained values.)*

The ability to shift between these two extremes, and potentially any intermediate configurations, demonstrates the flexibility and comprehensiveness of our formulation. By adjusting Ω, one can seamlessly transition from a global perspective to a local perspective. This flexibility not only caters to a wide range of pruning strategies but also provides a unified framework to compare and contrast the effectiveness of different pruning methods under a consistent mathematical lens.

4.1.4 Algorithm design

In this section, we introduce the algorithm design of *SparseLLM*, which alternatively optimizes the subproblems associated with the corresponding variables. This approach is resource-efficient and achieves global optimality, attributed to the closed-form solutions that each subproblem yields.

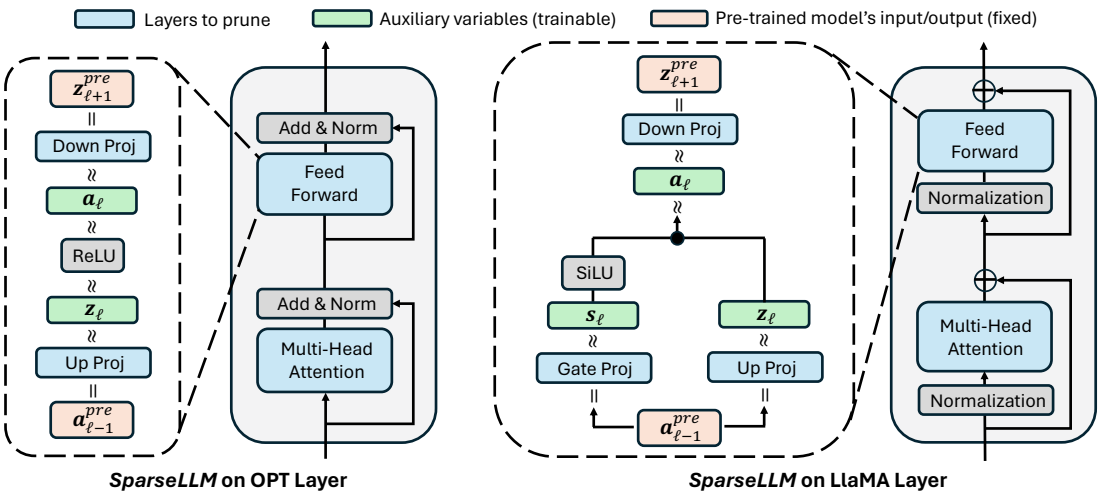


Figure 4.2: Illustration of *SparseLLM* on OPT and LlaMA. The auxiliary variables and soft constraints (i.e., $\approx$) allow *SparseLLM* to decompose the global pruning into manageable subproblems while maintaining the dependencies. Subproblems are *analytically* solvable and enjoy fast convergence.

The key idea of our algorithm lies behind the flexibility of Ω in our Eq. 4.3, as we want to find a better trade-off between completely global (memory bottleneck) and completely local (suboptimal performance) pruning. Naively applying SparseLLM to prune all layers globally is impractical. On the other hand, recent work shows that the feed-forward network (FFN) module in each decoder layer accounts for more than *two-thirds* of the total parameters in an LLM [83]. Therefore, our SparseLLM prioritizes the global pruning of the FFN module, while still adhering to a local pruning strategy for the multi-head attention (MHA) module (see Figure 4.2). This strategy strikes a balance between the computational feasibility of pruning large-scale models and the effectiveness of the pruning process, adhering to the limitations and practices of state-of-the-art LLM pruning frameworks.

Formally speaking, rather than trying to solve Eq. 4.3 directly, we first relax the constraints by adding an ℓ_2-penalty function to the objective and attack the

unconstrained problem:

$$\mathcal{L}(\mathbf{z}_L, \mathbf{y}) + \alpha \sum\nolimits_{\ell \in [L]} \|\mathbf{z}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}\|_2^2 + \beta \sum\nolimits_{\ell \in \Omega_{\text{FFN}}} \|\boldsymbol{a}_\ell - \phi_\ell(\mathbf{z}_\ell)\|_2^2, \quad (4.4)$$

where α, β are hyperparameters for controlling the weight of each constraint. Ω_{FFN} denotes the set of indexes for the linear layers in the FFN module of each decoder layer, i.e., linear layers from the same FFN module are pruned globally. For simplicity, the superscript "pre" of $\boldsymbol{a}_\ell$ and $\mathbf{z}_\ell$ in the third constraint in Eq. 4.3 is omitted here, i.e., for $\ell \notin \Omega_{\text{FFN}}$ the $\boldsymbol{a}_\ell$ and $\mathbf{z}_\ell$ are fixed and equal to the pre-trained model's intermediate value in the second term of Eq. 4.4. In the following subsections, we illustrate how we approach the pruning of FFN and MHA modules, respectively.

SparseLLM on OPT models

For each decoder layer in a pre-trained LLM, our Eq. 4.4 instantly simplifies to globally pruning the corresponding FFN module within that decoder layer as:

$$\alpha\|\mathbf{z}_{\ell+1}^{pre} - (\mathbf{M}_{\ell+1} \odot \widehat{\mathbf{W}}_{\ell+1})\boldsymbol{a}_\ell\|_2^2 + \beta\|\boldsymbol{a}_\ell - \phi_\ell(\mathbf{z}_\ell)\|_2^2 + \alpha\|\mathbf{z}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}^{pre}\|_2^2, \quad (4.5)$$

where layers ℓ and $\ell+1$ correspond to the up-projection and down-projection linear layers.

In this work, we consider the *alternating* method to optimize our Eq. 4.5, i.e., optimize each variable while keeping the rest fixed. The careful and elaborate design of our Eq. 4.5 allows us to derive a *closed-form* solution to every subproblem as shown below.

Pruning weight. First consider optimizing Eq. 4.5 with respect to $\mathbf{M}_\ell$ and $\widehat{\mathbf{W}}_\ell$. For each linear layer ℓ in a FFN module, the optimal solution minimizes $\|\mathbf{z}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}\|_2^2$. To solve it, the first step is to decompose $\mathbf{z}_\ell$ to $\mathbf{W}_\ell\boldsymbol{a}_{\ell-1}$, where $\mathbf{W}_\ell = \mathbf{z}_\ell\boldsymbol{a}_{\ell-1}^\dagger$ († denotes the pseudo-inverse.) Plug decomposed $\mathbf{z}_\ell$ back in original loss and

we get $\|\mathbf{W}_\ell \boldsymbol{a}_{\ell-1} - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}\|_2^2$, which aligns with the pruning objective of Eq. 4.2 and can be analytically solved by existing pruning solver e.g., SparseGPT. The superscript of "*pre*" for $\boldsymbol{a}_{\ell-1}$ is omitted in this section for simpler notation.

Updating activation. Minimization for $\boldsymbol{a}_\ell$ is a simple least-squares problem similar to weight pruning. However, in this case, the matrix $\boldsymbol{a}_{\ell-1}$ appears in two penalty terms in Eq. 4.5, so we must minimize $\alpha\|\mathbf{z}_{\ell+1}^{pre} - (\mathbf{M}_{\ell+1} \odot \widehat{\mathbf{W}}_{\ell+1})\boldsymbol{a}_\ell\|_2^2 + \beta\|\boldsymbol{a}_\ell - \phi_\ell(\mathbf{z}_\ell)\|_2^2$ for $\boldsymbol{a}_\ell$, holding all other variables fixed. By following a very similar idea to Ridge regression, the new value of $\boldsymbol{a}_\ell$ is given by:

$$(\alpha \mathbf{W}_{\ell+1}^\intercal \mathbf{W}_{\ell+1} + \beta \mathbf{I})^{-1}(\alpha \mathbf{W}_{\ell+1}^\intercal \mathbf{z}_{\ell+1}^{pre} + \beta \cdot \mathrm{ReLU}(\mathbf{z}_\ell)), \tag{4.6}$$

where $\mathbf{W}_\ell$ denotes the updated weight matrix after pruning, i.e., $\mathbf{W}_\ell \coloneqq \mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell$.

Updating output. The update for $\mathbf{z}_\ell$ requires minimizing the following loss:

$$\beta\|\boldsymbol{a}_\ell - \mathrm{ReLU}(\mathbf{z}_\ell)\|_2^2 + \alpha\|\mathbf{z}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}^{pre}\|_2^2. \tag{4.7}$$

This problem is non-convex and non-quadratic (because of the non-linear function ReLU). Fortunately, because the ReLU function works entry-wise on its argument, the entries in $\mathbf{z}_\ell$ are de-coupled. Solving Eq. 4.7 is particularly easy for the case of ReLU, as it can be solved in closed form followed by a simple if-then logic. Specifically, one only needs to compute two solutions of a quadratic equation:

$$\mathbf{z}_\ell^{(1)} = (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}^{pre}, \quad \mathbf{z}_\ell^{(2)} = (\alpha+\beta)^{-1} \cdot \left(\beta \boldsymbol{a}_\ell + \alpha \mathbf{z}_\ell^{(1)}\right), \tag{4.8}$$

where the first solution corresponds to those entries of $\mathbf{z}_\ell$ that are negative (reduced to zero by ReLU), and the second solution corresponds to those entries of $\mathbf{z}_\ell$ that are non-negative.

***SparseLLM* on LlaMA models**

In this section, we introduce how *SparseLLM* decomposes global pruning into subproblems and solves them iteratively on LlaMA model families. The model architecture of LlaMA can be found in Figure 4.2. Overall, *SparseLLM* operates similarly on both LlaMA and OPT models, with the main difference being that LlaMA includes an additional dense linear layer, known as the gate projection layer, and uses the SiLU activation function instead of ReLU.

Pruning weight. In this part, *SparseLLM* functions almost identically to its operation on OPTs.

Updating activation $\mathbf{a}_\ell$. Similarly, for updating $\boldsymbol{a}_\ell$, *SparseLLM* works nearly the same as on OPT. The minimization for $\boldsymbol{a}_\ell$ is a simple least-squares problem, akin to weight pruning. However, in this case, the matrix $\boldsymbol{a}_{\ell-1}$ appears in two penalty terms in Eq. 4.5, necessitating the minimization of:

$$\alpha\|\mathbf{z}_{\ell+1}^{pre} - (\mathbf{M}_{\ell+1} \odot \widehat{\mathbf{W}}_{\ell+1})\boldsymbol{a}_\ell\|_2^2 + \beta\|\boldsymbol{a}_\ell - \mathrm{SiLU}(\mathbf{s}_\ell) \odot \mathbf{z}_\ell\|_2^2, \tag{4.9}$$

for $\boldsymbol{a}_\ell$, with all other variables held fixed. Following a concept similar to Ridge regression, the updated value of $\boldsymbol{a}_\ell$ is:

$$\big(\alpha\mathbf{W}_{\ell+1}^\intercal\mathbf{W}_{\ell+1} + \beta\mathbf{I}\big)^{-1}\big(\alpha\mathbf{W}_{\ell+1}^\intercal\mathbf{z}_{\ell+1}^{pre} + \beta \cdot \mathrm{SiLU}(\mathbf{s}_\ell) \odot \mathbf{z}_\ell\big), \tag{4.10}$$

where $\mathbf{W}_\ell$ denotes the updated weight matrix after pruning, i.e., $\mathbf{W}_\ell \coloneqq \mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell$.

Updating output $\mathbf{z}_\ell$. Updating $\mathbf{z}_\ell$ is somewhat simpler in LlaMA since the activation function applies over the gate projection layer. The update requires minimizing the loss:

$$\beta\|\boldsymbol{a}_\ell - \mathrm{SiLU}(\mathbf{s}_\ell) \odot \mathbf{z}_\ell\|_2^2 + \alpha\|\mathbf{z}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}^{pre}\|_2^2. \tag{4.11}$$

This problem is quadratic when solving for $\mathbf{z}_\ell$ with other variables fixed. Through

mathematical manipulations, the analytical solution for $\mathbf{z}_\ell$ is found by solving a quadratic equation:

$$\mathbf{z}_\ell^* = \frac{(\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell)\boldsymbol{a}_{\ell-1}^{pre} + \mathrm{SiLU}(\mathbf{s}_\ell) \odot \boldsymbol{a}_\ell}{\mathrm{SiLU}(\mathbf{s}_\ell)^2 + \mathbf{1}}, \tag{4.12}$$

where the division is element-wise and $\mathbf{1}$ denotes the all-one matrix.

Updating gate projection output $\mathbf{s}_\ell$. Updating $\mathbf{s}_\ell$ involves minimizing:

$$\beta\|\boldsymbol{a}_\ell - \mathrm{SiLU}(\mathbf{s}_\ell) \odot \mathbf{z}_\ell\|_2^2 + \alpha\|\mathbf{s}_\ell - (\mathbf{M}_s \odot \widehat{\mathbf{W}}_s)\boldsymbol{a}_{\ell-1}^{pre}\|_2^2, \tag{4.13}$$

where $\mathbf{M}_s$ and $\widehat{\mathbf{W}}_s$ denote the mask and layer weights for the gate projection layer. This problem is non-convex and non-quadratic due to the non-linear SiLU function. However, since SiLU operates entry-wise, the entries in $\mathbf{s}_\ell$ are decoupled. Despite LlaMA lacking a simple closed-form solution as in OPT (which uses ReLU), the problem can still be solved quickly and analytically using a lookup table of pre-computed solutions, since each element in $\mathbf{s}_\ell$ depends on only three variables.

Remark 4.1.2 (Global convergence of SparseLLM)**.** *Consider the objective function given by Eq. 4.5, under the condition that the activation function ϕ is ReLU. Notice that (1) the objective function is convex with respect to each variable when all others are fixed, and (2) given that closed-form solutions exist for the subproblems in the alternating optimization scheme, the proposed algorithm resembles multiblock ADMM which has been shown to converge to in many applications.*

Pruning of MHAs

SparseLLM also prunes other linear layers besides those in FFNs. By following Eq. 4.4, for each linear layer out of FFN modules, the pruning objective simplifies to $\alpha\|\mathbf{z}_{\ell+1}^{pre} - (\mathbf{M}_{\ell+1} \odot \widehat{\mathbf{W}}_{\ell+1})\boldsymbol{a}_\ell^{pre}\|_2^2$, which is equivalent (with some simple math) to that of completely local pruning as shown in Eq. 4.2. Existing LLM pruning solvers such

as SparseGPT and Wanda are applicable here.

Time complexity analyses

The proposed *SparseLLM* consists of three main steps, with the overall time complexity being the sum of the complexities of these steps. In the weights pruning step, the complexity is dominated by the pseudo-inverse computation of matrix $\boldsymbol{a}_\ell$ (dimensions $n \times h$), which is $O(nh^2)$. Using SparseGPT as the solver, the exact pruning step has a complexity of $O(h^3)$. The second step, updating activations, involves matrix inversion of the weight matrix $\mathbf{W}_\ell$ (size $h \times h$) with a complexity of $O(h^3)$. The third step, updating outputs, has a lower complexity. Thus, the overall algorithm complexity is bounded by $O(h^3)$, therefore making our method's per-epoch time complexity comparable to SparseGPT.

4.1.5 Experiments

Experiments setup. We implemented *SparseLLM* in PyTorch [110] and use the HuggingFace Transformers library [152] for handling models and datasets. All pruning experiments are conducted on NVIDIA A100 GPUs. For calibration data, we follow [39] and use 128 2048-token segments, randomly chosen from the first shard of the C4 [114] dataset. This represents generic text data crawled from the internet and ensures our experiments are zero-shot as no task-specific data is seen during pruning. *We followed existing work [39, 133] and pruned all linear layers (in FFN and MHA) to the target sparsity.*

Models, datasets & evaluation. We consider the OPT model family [164] and LlaMA-2 model family [141] in our experiments as well as the most recent LlaMA-3 model. We show results on different sizes of models to provide a broader picture for the performances of *SparseLLM*. In terms of metrics, we mainly focus on perplexity, which is known to be a challenging and stable metric that is well-suited for evaluating

the accuracy of compression methods [158, 27]. We consider the test sets of raw-WikiText2 [98] (WT2) and PTB [93] as well as a subset of the C4 validation data, all popular benchmarks in LLM compression literature [158, 109, 39, 133]. For additional interpretability, we also provide zero-shot accuracy results following the same setup of [133], which is based on the popular EleutherAI-eval harness [41].

Comparison methods. We compare against three baselines, magnitude pruning [172] applied locally, and two other state-of-the-art local pruning methods, SparseGPT [39] and Wanda [133].

Results and analyses

Pruning vs. model sizes. We begin by exploring the pruning capabilities of *SparseLLM* across various model sizes in comparison to baseline methods. For each model, we consider unstructured sparsity ranging from 70% to 90% with a 10% increment, as well as a 3:4 semi-structured sparsity. The 3:4 semi-structured sparsity is inspired by our preliminary results that suggest good performance *SparseLLM* at high sparsity regimes. However, note that two of our baselines, Magnitude and Wanda, are unable to be configured to this sparsity out-of-box. We conduct a sensitivity study on the calibration sample sizes (see Appendix D.2) and use calibration sample sizes between 32 and 64 for all experiments. Moreover, we prune the first 50% of the Transformer decoder layers in each model to achieve a balance between the computation resources and the performances. Detailed results can be found in Table 4.1 and Table 4.2 as well as Table D.3 in Appendix D.4. Note that in Table 4.2 for LlaMA-3 model, we only compare SparseGPT to the proposed *SparseLLM*. The perplexity results of the dense models are reported next to the names of the models.

From the tables, it shows a general trend of increasing perplexity with increasing sparsity. Moreover, we observe a trend of decreasing perplexity for SparseGPT and *SparseLLM* at the same sparsity with increasing model sizes. However, such a

Table 4.1: Perplexity of OPT models for sparsity $\geq 70\%$; the lower the perplexity, the better.

OPT-1.3b (WikiText2 (WT2): 14.62; PTB: 20.29; C4: 16.07)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	6420.80	4828.13	3435.99	9998.71	1.1e4	5347.89	8209.13	1.0e4	4917.02	-	-	-
Wanda	21.56	34.77	25.78	142.20	146.76	142.24	5692.65	4751.69	4501.73	-	-	-
SparseGPT	18.04	28.19	21.45	69.67	93.36	60.83	2596.70	2361.86	1363.08	252.81	238.41	146.21
SparseLLM	17.82	27.72	20.99	58.92	85.33	58.36	1350.31	1192.36	655.76	128.83	144.48	106.01
OPT-2.7b (WikiText2 (WT2): 12.47; PTB: 17.97; C4: 14.32)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	1691.74	1237.08	1415.02	1.0e4	7916.69	6050.07	7.9e5	5.3e5	4.7e5	-	-	-
Wanda	88.61	140.09	90.06	6140.81	4746.96	5678.66	3.0e4	3.5e4	2.4e4	-	-	-
SparseGPT	13.79	21.18	16.18	24.32	37.82	25.92	2662.74	2285.01	1776.08	91.02	91.79	64.95
SparseLLM	13.82	21.07	16.14	23.87	37.09	24.90	1200.12	759.11	527.70	56.90	77.14	52.77
OPT-13b (WikiText2 (WT2): 10.13; PTB: 14.52; C4: 12.06)												
Sparsity	70%			80%			85%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	9037.12	7734.58	5909.47	1.1e4	9140.88	6340.22	1.3e4	1.3e4	9087.50	-	-	-
Wanda	30.94	39.26	33.31	4216.04	2894.77	2450.57	1.1e4	1.1e4	7244.96	-	-	-
SparseGPT	10.89	16.35	13.39	21.42	33.62	21.01	8408.03	6380.30	3416.23	4715.16	7454.37	2.11e4
SparseLLM	10.96	16.57	13.38	19.07	28.77	19.29	2052.27	1536.51	538.61	289.17	687.48	677.13
OPT-30b (WikiText2 (WT2): 9.56; PTB: 14.04; C4: 11.45)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	8691.40	4769.89	4732.66	8941.81	5292.98	5092.26	3.8e7	3.0e7	1.4e7	-	-	-
Wanda	7766.61	5547.45	5741.74	8770.33	6020.70	7132.20	6354.15	4296.37	4654.27	-	-	-
SparseGPT	9.58	14.41	11.93	16.49	22.01	17.67	5747.87	5169.50	3555.24	441.35	464.73	209.44
SparseLLM	9.56	14.40	11.94	15.61	19.64	16.61	3050.63	2712.39	1758.63	51.28	73.61	37.99
OPT-66b (WikiText2 (WT2): 9.34; PTB: 13.36; C4: 10.99)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	-	-	-
Wanda	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	-	-	-
SparseGPT	9.45	13.64	11.37	28.27	57.41	26.26	7803.10	6594.88	4433.35	6594.37	6329.59	3799.87
SparseLLM	9.37	13.66	11.37	16.45	21.00	17.70	7504.17	5644.65	3683.91	4641.8	5296.93	1618.43

trend is not obvious for Magnitude and Wanda. We also observe that SparseGPT and *SparseLLM* consistently outperform Magnitude and Wanda by a significant margin. For smaller sparsity, *SparseLLM* achieves comparable perplexity to SparseGPT. As we increase the sparsity, *SparseLLM* starts to demonstrate noticeable improvements over SparseGPT. In numerous instances for the OPT model family, *SparseLLM* achieves perplexity reductions of more than 50% compared to SparseGPT. We also

Table 4.2: Perplexity of LlaMA models for sparsity $\geq 70\%$; the lower the perplexity, the better.

LlaMA-2 7b (WikiText2 (WT2): 5.47; PTB: 37.91; C4: 7.26)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	1058.00	544.43	889.46	6380.27	NaN	4162.92	9498.91	1.02e4	7539.65	-	-	-
Wanda	2644.22	4040.95	1630.09	1814.01	3376.35	1124.26	5206.93	4607.30	2780.45	-	-	-
SparseGPT	15.98	302.15	18.58	53.20	803.02	52.57	344.97	2503.82	279.77	68.28	784.79	60.45
SparseLLM	16.15	274.35	18.23	49.96	664.39	47.39	225.23	2233.52	181.56	64.17	667.27	54.56
LlaMA-2 13b (WikiText2 (WT2): 4.88; PTB: 50.94; C4: 6.73)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	30.34	2317.39	28.48	4133.98	4706.65	4112.69	5580.71	5514.22	5090.63	-	-	-
Wanda	23.42	502.53	32.65	295.29	2340.13	261.15	3003.49	3804.69	1738.73	-	-	-
SparseGPT	12.98	267.63	15.95	45.59	550.59	45.20	825.99	1410.46	673.33	63.48	660.70	56.29
SparseLLM	12.95	277.76	15.77	36.36	578.35	38.63	646.15	1078.94	466.98	53.71	632.11	50.40
LlaMA-3 8b (WikiText2 (WT2): 6.14; PTB: 11.18; C4: 9.45)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
SparseGPT	22.37	36.56	30.53	72.87	113.95	79.86	214.68	261.18	198.34	96.75	107.52	102.11
SparseLLM	20.98	33.78	28.94	57.83	85.98	72.18	197.47	241.68	181.69	76.33	99.54	93.68

Table 4.3: Perplexity of 2:4 sparsity; the lower the perplexity, the better.

	OPT-1.3b			OPT-2.7b			OPT-6.7b			OPT-13b		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	96.68	133.92	48.08	272.34	308.55	267.70	64.11	92.23	82.67	67.07	110.77	52.61
Wanda	15.63	24.04	18.23	13.66	21.67	16.10	11.86	18.54	14.77	10.33	15.35	12.54
SparseGPT	15.11	23.71	17.88	12.62	19.28	15.12	11.30	16.90	13.51	10.20	15.14	12.48
SparseLLM	14.97	23.40	17.67	12.62	19.28	15.12	11.07	16.73	13.42	10.20	15.14	12.41

see that performance improvements from *SparseLLM* over SparseGPT are more significant for the OPT model family than the LlaMA-2 model family.

We provide additional set of perplexity results for a 2:4 semi-structured sparsity for a few OPT models in Table 4.3. We see that *SparseLLM* and SparseGPT generally outperform Magnitude and Wanda while *SparseLLM* has comparable if not slightly better performances compared to SparseGPT with the 2:4 semi-structured sparsity. Note that a 2:4 semi-structure sparsity is considered to be in low sparsity regime.

Zero-shot experiments. To further conclude the evaluations and discussions, we show results for several zero-shot tasks in Table 4.4 and Table 4.5 as well as Table D.4 in Appendix D.4, comparing SparseGPT and *SparseLLM*. These evaluations

are known to be relatively noisy [28], but more interpretable. We also report the results for zero-shot tasks from the dense models in the "Dense" row. We see that the accuracy of both methods decreases with increasing sparsity, which is expected, as more parameters are pruned. A similar trend of increasing accuracy with increasing model size is observed too. Across all the tasks, OBQA and ARC-c remain the most challenging ones as the accuracy for both methods is 30% or below 30% while both methods perform well for BoolQ, RTE, WinoGrande, and ARC-e. In general, *SparseLLM* is able to achieve higher accuracy in the majority of tasks across the models of different sizes in both OPT and LlaMA-2 model families.

Table 4.4: Accuracy (%) of zero-shot tasks for OPT models; the higher the accuracy, the better.

OPT-13b									
Sparsity	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
Dense		65.87	57.76	52.44	66.02	67.82	33.46	28.62	53.14
70%	SparseGPT	63.03	54.87	50.89	65.43	67.47	32.85	26.40	51.56
	SparseLLM	63.85	55.23	50.73	65.67	66.46	31.83	27.20	51.57
80%	SparseGPT	59.72	52.35	46.82	61.48	62.50	31.23	21.80	47.99
	SparseLLM	60.89	53.07	46.19	62.12	62.21	30.38	23.00	48.27
90%	SparseGPT	47.49	52.71	33.17	51.54	39.98	21.33	17.80	37.72
	SparseLLM	53.43	52.71	38.19	52.96	46.68	25.26	17.40	40.95
3:4	SparseGPT	47.55	53.43	31.30	50.20	37.63	22.53	17.60	37.18
	SparseLLM	51.13	52.35	38.51	55.96	49.24	24.83	21.40	41.92
OPT-30b									
Sparsity	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
Dense		70.46	61.82	54.27	69.02	70.47	35.49	30.20	55.96
70%	SparseGPT	68.78	58.48	53.83	67.64	69.15	34.30	29.60	54.54
	SparseLLM	69.11	61.73	53.97	68.43	69.78	34.73	29.80	55.36
80%	SparseGPT	64.86	60.65	49.73	61.40	61.91	31.74	24.20	50.64
	SparseLLM	65.41	59.57	50.65	61.96	62.71	32.25	26.50	51.29
90%	SparseGPT	37.83	53.79	25.96	49.88	26.47	20.22	12.60	32.39
	SparseLLM	43.55	52.35	26.32	50.04	27.31	20.56	14.00	33.45
3:4	SparseGPT	55.81	51.26	33.64	54.54	42.05	21.33	21.00	39.95
	SparseLLM	60.83	54.15	39.35	55.41	45.24	24.06	22.20	43.03

Table 4.5: Accuracy (%) of zero-shot tasks for LlaMA models; the higher the accuracy, the better.

LlaMA-2 7b									
Sparsity	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
Dense		75.05	66.43	56.92	69.93	75.34	41.89	34.40	59.99
70%	SparseGPT	68.26	57.04	39.67	59.04	60.9	28.58	20.60	47.73
	SparseLLM	67.61	57.31	40.12	61.39	59.39	28.76	21.40	48.13
80%	SparseGPT	59.36	52.71	28.83	48.7	34.22	18.34	14.40	36.65
	SparseLLM	60.12	53.07	28.62	50.59	34.55	18.69	14.30	37.13
90%	SparseGPT	39.02	52.34	26.66	47.80	28.32	17.37	12.40	31.99
	SparseLLM	39.45	52.71	26.79	51.17	28.32	19.52	12.50	32.92
3:4	SparseGPT	53.94	54.15	28.09	49.17	31.57	17.41	14.80	35.59
	SparseLLM	57.34	53.43	28.26	48.86	32.45	18.17	14.4	36.13
LlaMA-2 13b									
Sparsity	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
Dense		77.89	70.40	59.94	72.77	77.40	46.50	33.20	62.59
70%	SparseGPT	70.03	53.43	42.20	66.54	64.94	31.66	25.40	50.60
	SparseLLM	69.87	54.15	42.50	68.64	64.97	31.40	25.80	51.05
80%	SparseGPT	62.69	52.71	28.94	50.91	36.24	18.17	14.00	37.67
	SparseLLM	64.39	52.86	29.19	51.46	35.69	18.77	14.20	38.08
90%	SparseGPT	50.21	51.35	26.71	49.14	26.68	19.71	13.2	33.86
	SparseLLM	55.35	52.05	26.89	51.34	27.35	19.62	14.20	35.26
3:4	SparseGPT	61.28	53.71	28.40	47.99	33.21	18.26	14.00	36.69
	SparseLLM	61.71	55.71	28.56	51.62	32.11	18.49	13.8	37.43

4.2 Federated Pruning of Large Language Models

4.2.1 Introduction

Large Language Models (LLMs) such as GPT [106] and LlaMA [141] have recently transformed the field of Natural Language Processing (NLP) due to their ability to perform exceptionally well across a variety of complex language benchmarks. However, the increasing scale of these models, which can contain billions of parameters, also brings significant computational and storage costs. The high memory and inference costs make it challenging to deploy LLMs in real-world applications where resources are constrained [8]. Consequently, there has been an increasing interest in *model compression* techniques such as pruning, quantization, and knowledge distillation, which aim to reduce the computational load while maintaining model per-

formance [173]. Among these techniques, *pruning* has emerged as a highly effective approach for reducing the size and complexity of LLMs by introducing sparsity into the models.

Despite the recent success of LLM pruning methods, existing approaches predominantly assume that the calibration data used for pruning is publicly available [39, 133]. However, in many real-world scenarios, especially when dealing with sensitive applications like medical agents or financial systems, the data used for pruning might be private and cannot be shared openly [119]. On the other hand, Federated Learning (FL), a distributed machine learning technique that enables multiple clients to collaboratively train models without sharing their private data, has gained significant popularity in traditional machine learning [165]. However, most works on LLMs in FL settings have focused on fine-tuning. Due to the intrinsic differences between fine-tuning and pruning, existing FL-based fine-tuning methods cannot handle the problem of pruning LLMs with private data.

To address the challenges posed by pruning LLMs in federated settings, where private data cannot be shared and heterogeneity exists among clients, we propose a novel method called **FedSpaLLM** (Federated Sparse LLM). FedSpaLLM is the first framework that allows pruning LLMs under a federated learning setting with resource heterogeneity. Our method allows each client to prune its local model based on its data while maintaining privacy and accommodating diverse computational resources.

Our contributions are summarized as follows:

- **Federated pruning framework for LLMs**: We present the first framework for pruning LLMs in FL, allowing collaborative pruning with private local data.
- **ℓ_0-norm aggregation**: We introduce a novel aggregation function that preserves important weights by averaging only non-zero elements across client models.
- **Adaptive mask expansion**: We propose a mask expansion technique to meet global sparsity targets while accounting for client-specific pruning.

- **Layer sampling**: We develop a resource-aware layer sampling strategy, enabling personalized pruning and reducing communication costs.
- **Extensive evaluation**: We conduct comprehensive experiments, showing that FedSpaLLM improves both pruning efficiency and model performance in heterogeneous federated environments.

4.2.2 Related Work

Pruning of LLMs. *Pruning* regained prominence in the late 2010s for reducing inference costs [44]. LLM pruning can be categorized into *structured* and *unstructured* pruning.

Unstructured pruning removes individual parameters without regard to model structure, often using thresholds to nullify smaller weights. SparseGPT [39] achieves up to 60% parameter reduction in LLMs with minimal performance loss. Wanda [133] introduces a pruning criterion based on both weight magnitude and activations, particularly effective in linear layers. DynaTran [142] dynamically prunes activations at runtime, enhanced by a custom ASIC architecture.

Structured pruning removes groups of parameters such as filters or attention heads. LLM-Pruner [91] combines first-order data and Hessian information for structured pruning, while LoSparse [75] uses low-rank and sparse approximations to balance pruning and model expressiveness. Structured pruning of hidden dimensions, as shown by [137], extends to embeddings and attention heads. ZipLM [62] optimizes structured compression for accuracy and hardware efficiency.

Federated Learning with LLMs. FL on LLMs has primarily focused on LLM *fine-tuning* and has gained attention for enabling private and efficient model updates. FedPrompt [167] introduces prompt-tuning in FL, reducing communication costs by updating only soft prompts. Fang et al. [37] and Li et al. [78] leverage low-rank adapters for parameter-efficient fine-tuning, improving training speed and accuracy.

FedNLP [79] provides a benchmarking framework for evaluating FL methods on NLP tasks. FedAdapter [21] uses adapters to accelerate model convergence in federated settings. FeDeRA [156] employs singular value decomposition to further improve LoRA-based fine-tuning efficiency. C2A [55] introduces a hypernetwork-based framework for generating client-specific adapters to handle client heterogeneity. FedBPT [132] enables efficient prompt-tuning with a gradient-free approach, reducing memory and communication costs. PrE-Text [49] generates differentially private synthetic data to enable central training, reducing on-device computation.

Federated Pruning on DNNs. Model pruning in FL improves efficiency by reducing communication and computation costs. FedP3 [160] and HeteroFL [29] address client model heterogeneity, enabling smaller, personalized models. FedTiny [50] and PruneFL [53] implement progressive pruning to fit models within resource constraints. FedPrune [102] improves performance by pruning global models based on client capabilities, while Complement Sparsification [52] reduces communication overhead using sparsity techniques. However, all the work above only applies to smaller DNNs, and cannot trivially scale to massive LLMs.

Federated Distillation. In addition to model pruning, knowledge distillation (KD) has been explored as a resource-efficient approach to training NNs in edge environments [73]. In this paradigm, a smaller proxy model is used to facilitate learning, either by mimicking the behavior of a larger model or by aggregating knowledge from multiple clients. For instance, FedBiOT [153] introduces a method for locally fine-tuning LLMs without requiring full-model transmission, allowing clients to distill and learn from a proxy model. Similarly, DaFKD [148], a domain-aware FL knowledge distillation framework, incorporates domain-specific adaptations to improve the generalization of models across heterogeneous clients.

Despite the merits of federated distillation, our proposed approach offers several advantages. Proxy models and FL distillation methods require training additional

models, increasing computational overhead. Additionally, they assume an accessible public dataset for generating the soft labels, which usually does not hold in practice. Our framework directly prunes the global LLM without auxiliary models and any additional public data. It achieves target sparsity across heterogeneous clients while maintaining privacy through local pruning and aggregation.

4.2.3 Preliminaries

Pruning of LLMs

In this work, we focus on *unstructured* pruning, [39, 133, 9] which typically utilizes local pruning. Local pruning circumvents the memory issue mentioned above by dividing the full model compression into sub-problems for each layer and constructing a *local loss* to measure the ℓ_2-error between the outputs of the uncompressed and compressed layers. Hence, the local pruning can be formulated by:

$$\min_{\mathbf{M}_\ell, \widehat{\mathbf{W}}_\ell} \|\mathbf{W}_\ell \cdot \mathbf{X}_\ell - (\mathbf{M}_\ell \odot \widehat{\mathbf{W}}_\ell) \cdot \mathbf{X}_\ell\|_2^2, \tag{4.14}$$

where $\odot$ denotes element-wise multiplication. Although smaller than the global pruning, the local pruning still needs to optimize both the mask $\mathbf{M}_\ell$ and the remaining weights $\widehat{\mathbf{W}}_\ell$ and thus remains NP-hard. Therefore, exactly solving it for larger layers is unrealistic, leading all existing methods to resort to approximations.

Mask Selection & Weight Reconstruction. A particularly popular approach is to separate the problem into *mask selection* and *weight reconstruction* [51, 64]. Concretely, this means first choosing a pruning mask $\mathbf{M}$ according to some salient criterion, like the weight magnitude [172], and then optimizing the remaining unpruned weights while keeping the mask unchanged. Importantly, once the mask is fixed, Eq. 4.14 turns into a *linear regression* problem that can be easily optimized.

Background of Federated Learning

FL is a distributed machine learning paradigm designed to enable collaborative training of a single global model without exchanging private data between clients [43]. In this setup, multiple clients, each with their own local dataset, train their local models independently. The central server coordinates the process by sending the global model to a selected group of clients. These clients then perform local optimization using their respective datasets. After local training, the updated models are sent back to the server, where they are aggregated to update the global model.

In each communication round, the server aggregates the updates from multiple clients to refine the global model. Mathematically, let $\mathcal{D}_i$ denote the local dataset of the i-th client, and θ represent the global model parameters. The process of updating the global model is given by:

$$\tilde{\theta} \in \arg\min \sum_{i=1}^{K} \alpha_i \cdot \mathcal{L}(\mathcal{D}_i; \theta), \tag{4.15}$$

where $\mathcal{L}(\mathcal{D}_i; \theta)$ is the local objective function computed over the dataset $\mathcal{D}_i$ with the model parameters θ. The factor α_i is typically proportional to the size of client i's dataset, i.e., $\alpha_i = \frac{|\mathcal{D}_i|}{\sum_i |\mathcal{D}_i|}$, and K denotes the total number of participating clients.

There exist various methods for aggregating client updates, with FedAvg [97] being the most widely adopted due to its simplicity and effectiveness in a wide range of FL applications. This method aggregates the client updates weighted by their respective dataset sizes, providing a robust way to train models in data-sensitive environments.

4.2.4 Proposed Method

In this section, we present FedSpaLLM, our novel framework for federated pruning of large language models (LLMs). The proposed method is designed to address the challenges of pruning in federated learning settings, specifically targeting com-

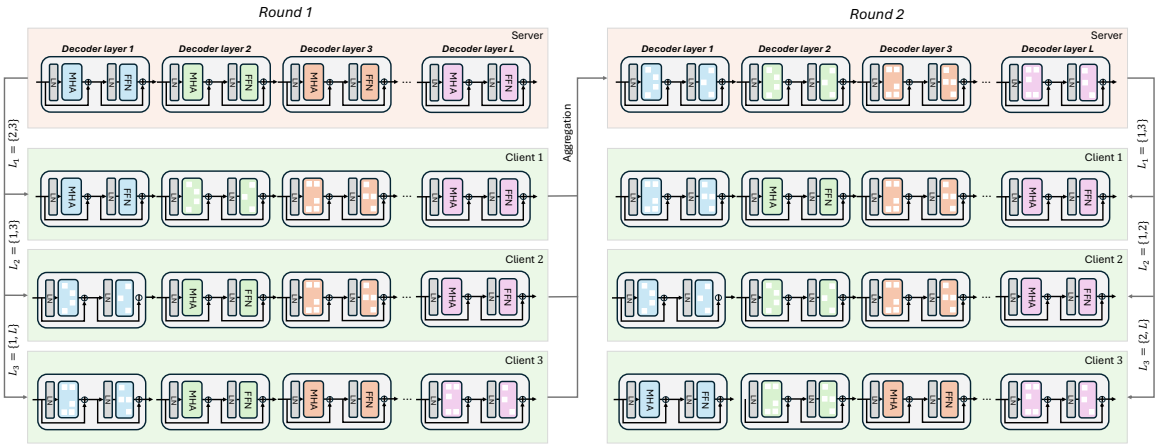


Figure 4.3: Visualization of the proposed **FedSpaLLM** framework. Instead of transmitting the full model at each communication round, the server samples a subset of layers based on each client's computational resources. Clients prune only the sampled layers and retain the rest from their cached pre-trained dense model. After local pruning, clients only send their pruned layers to the server, which aggregates the pruned layers using a novel ℓ_0-norm aggregation function that averages only the non-zero parameters. This approach ensures that important weights are preserved while reducing communication overhead. The layer sampling strategy enables personalized pruning tailored to client heterogeneity, reducing resource usage without compromising overall model performance.

munication efficiency and system heterogeneity across clients. We first formulate the problem of pruning LLMs in an FL setup and introduce a specialized aggregation function based on the ℓ_0-norm to handle sparse model updates. Next, we propose an adaptive mask expansion technique to ensure that the global model meets the target sparsity, even when clients generate diverse pruning masks. Finally, we introduce a layer sampling strategy that allows clients to prune subsets of model layers based on their computational resources, enabling personalized pruning and reducing communication costs.

Figure 4.3 provides a visual overview of the FedSpaLLM framework, illustrating how clients and the central server interact during the pruning process.

Problem Formulation

We present the first formulation of pruning LLMs under the FL setting. In this scenario, multiple clients collaboratively prune a global LLM while ensuring that their local data remains private. Let W_g denote the global model parameters, where $W_g \in \mathbb{R}^d$, and each client i holds its own local dataset $\mathcal{D}_i = \{X_i, Y_i\}$ for training and pruning purposes.

During pruning, each client applies a binary pruning mask, $M_i \in \{0,1\}^d$, to its local model W_i. This mask determines which weights are retained ($M_i = 1$) and which are pruned ($M_i = 0$). The objective is to prune the global model while ensuring that the pruned models on each client still perform effectively on their respective local datasets. Importantly, our formulation allows for model heterogeneity or personalization, meaning that each client can have its own set of model parameters, W_i, which differs from the global model, W_g. This flexibility contrasts with the traditional FL setting, where all clients share the same global model.

The overall objective of federated pruning is to minimize the weighted sum of local losses across clients. Each client i minimizes its local loss function $\mathcal{L}_i$ based on the pruned model $M_i \odot W_i$. Formally, this can be written as:

$$\min_{W_1,\ldots,W_N,M_1,\ldots,M_N} \sum_{i=1}^{N} \alpha_i \cdot \mathcal{L}_i(M_i \odot W_i), \tag{4.16}$$

subject to the global model W_g being an aggregation of the locally pruned models:

$$W_g = \Omega\Big(M_1 \odot W_1, \ldots, M_N \odot W_N\Big), \tag{4.17}$$

where Ω denotes the aggregation function that combines the pruned models from all clients to update the global model.

Our formulation is general and supports model heterogeneity, where each client

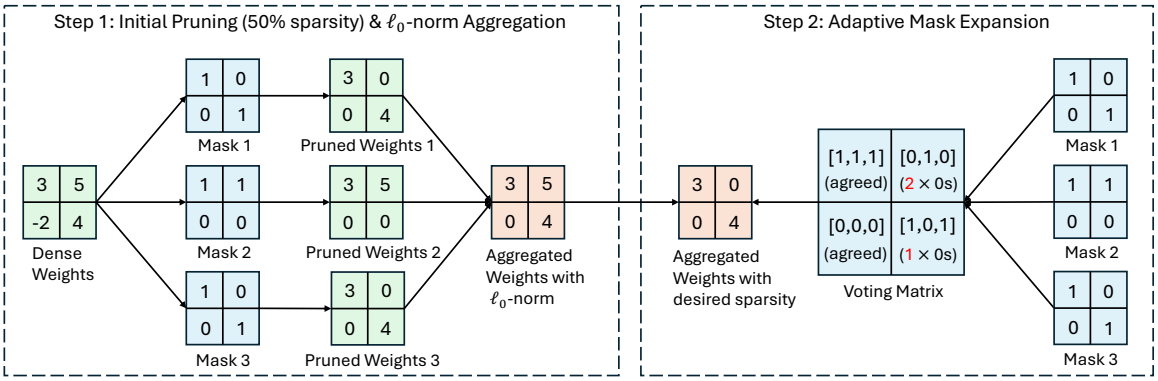


Figure 4.4: Visualization of the proposed **Aggregation Function** of FedSpaLLM to handle heterogeneous sparsified parameters. After clients prune their local models, the server aggregates the pruned layers by using the ℓ_0-norm aggregation function. This method avoids diluting the effect of unpruned weights by excluding zeros from the averaging process, thus preserving important parameters. To achieve the target global sparsity, an adaptive mask expansion is applied: the server counts the number of times each weight has been pruned across clients and uses this information to expand the pruning mask. The mask expansion prioritizes pruning weights that are most commonly pruned across clients, balancing individual client pruning decisions with the global sparsity goal.

may have its own model parameters, W_i, as opposed to the vanilla FL setting sharing a common global model. This flexibility is crucial for accommodating diverse client environments, making our method applicable to a wide range of real-world federated pruning scenarios.

Aggregation Function

ℓ_0-norm Aggregator. In traditional FL, the model weights from different clients are aggregated using a simple mean average or a similar function after each round of local updates. However, in our case, the parameters are sparsified through pruning, resulting in binary pruning masks. Due to the discrete nature of these masks, using a vanilla average function is not suitable. Specifically, when aggregating binary masks, dividing by the total count of clients would incorrectly include zeros (pruned positions) in the calculation, causing the averaged value to approach zero. This leads to an undesirable convergence, where the model parameters tend to vanish as the

denominator grows larger than it should be.

To address this, we propose an aggregation function based on the ℓ_0-norm. Instead of averaging over all elements, we compute the average only over the non-zero elements (i.e., weights that are retained across clients). Mathematically, given the pruning masks $M_1, M_2, \ldots, M_N$ from N clients and the corresponding local weights $W_1, W_2, \ldots, W_N$, the aggregation is defined as:

$$W_g = \frac{\sum_{i=1}^{N} M_i \odot W_i}{\left\| \sum_{i=1}^{N} M_i \right\|_{\ell_0}}. \tag{4.18}$$

In this way, the aggregation only considers the non-zero elements, ensuring that the resulting global weights do not tend toward zero unless they are truly pruned across all clients. This method ensures the correct balance between sparsity and preserving important weights in the global model.

Adaptive Aggregation w/. Mask Expansion. Another challenge arises from the heterogeneity of data across clients. Even with the same target sparsity, different clients may generate different pruning masks based on their local data. If we simply averaged these masks, the result would be equivalent to taking the intersection of all local masks. In other words, only the weights pruned by all clients would be pruned in the global model. This leads to a situation where the global model's sparsity is always smaller than the target sparsity, as fewer weights are pruned overall.

To address this, we propose an *mask expansion* technique. The idea is to first apply the ℓ_0-norm aggregation described above and then expand the global pruning mask to achieve the target sparsity. Specifically, after the initial aggregation, we count the number of zeros (pruned positions) for each weight across clients. If the count is below the target sparsity, we sort the remaining weights and select the ones with the most agreement (i.e., those that are most commonly pruned) to prune further, ensuring the final sparsity matches the desired level.

Let C_j represent the number of zeros for the j-th weight across clients, where:

$$C_j = \sum_{i=1}^{N} \mathbb{I}(M_i^{(j)} = 0), \tag{4.19}$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if $M_i^{(j)} = 0$ (the j-th weight is pruned by client i) and 0 otherwise. After counting, we sort the weights based on their C_j values and expand the pruning mask for weights that are not fully agreed upon by all clients. To achieve the target sparsity s, we select the top-k weights from the sorted list, where:

$$k = \text{ceil}(s \cdot d) - \sum_{j=1}^{d} \mathbb{I}(C_j = N), \tag{4.20}$$

where d is the total number of weights and s is the target sparsity. The first term ensures that the final mask achieves the desired sparsity, and the second term subtracts the number of weights that are already pruned by all clients (i.e., those where $C_j = N$).

By employing this sorting and expansion method, we ensure that the global pruning mask adheres to the desired sparsity while reflecting the commonly agreed-upon pruned positions, balancing individual client decisions and the global sparsity requirement.

Personalization with Layer Sampling

In FL for LLM pruning, communication overhead, and system heterogeneity are key challenges. The pruning of LLMs is typically done in a local manner, where the pruning of each decoder layer is independent, allowing for efficient and flexible layer sampling. This independence enables us to design a novel sampling strategy that maintains both efficiency and pruning accuracy while addressing the diverse computational capacities of clients.

Remark: Unlike traditional FL, where models are typically fine-tuned holistically, pruning in LLMs allows each decoder layer to be pruned independently. We exploit this property by sampling layers, significantly reducing communication costs without compromising the overall pruning outcome.

Layer Sampling Strategy. In each communication round, the server randomly samples a subset of layers from the LLM for each client to prune. Let $L = \{l_1, l_2, \ldots, l_m\}$ denote the set of all layers in the LLM, where m is the total number of layers. For each client i, the server selects a subset of layers $L_i \subseteq L$ to be pruned, where the number of layers sampled, $|L_i| = k_i$, is proportional to the computational capacity of client i, denoted as r_i. Formally,

$$L_i = \text{Sample}(L, k_i), \quad k_i \propto r_i. \tag{4.21}$$

Once client i receives the subset L_i, it prunes the sampled layers while retaining the original weights of the unsampled layers. Let $W_g = \{W_1, W_2, \ldots, W_m\}$ represent the global model weights, and $M_i = \{M_1, M_2, \ldots, M_m\}$ represent the pruning masks for client i. The locally pruned model W_i' is updated as:

$$W_i' = \begin{cases} M_j \odot W_j, & \text{if } j \in L_i, \\ W_j^{\text{dense}}, & \text{if } j \notin L_i. \end{cases} \tag{4.22}$$

For unsampled layers $j \notin L_i$, we retain their original unpruned dense weights W_j^{dense}. This process ensures that communication costs are minimized, as we only need to communicate the sampled layers L_i, only a portion of the model, in each round.

Corollary 4.2.1 (Sparsity Guarantee)**.** *Let $\mathcal{S}_{global}$ denote the target global sparsity, and let $\mathcal{S}_i$ be the sparsity achieved by client i on its local model. If the layer sampling strategy ensures that all layers $\mathcal{L}$ are sampled at least once across all clients in each communication round, the aggregated global pruned model $\hat{W}_g$ will maintain the same*

sparsity as the target sparsity $\mathcal{S}_{global}$. Since all clients share the same target sparsity, and our ℓ_0-norm aggregation guarantees that the aggregated layers have the exact sparsity as the locally pruned layers, the sparsity of the global model will always equal the desired target. Formally:

$$\mathcal{S}_{global} = \mathcal{S}_i, \quad \forall i, \tag{4.23}$$

where N is the total number of clients. Thus, the sparsity of the global model is consistent with the target across all communication rounds.

This corollary ensures that, with our sampling strategy, the global model will meet the desired global sparsity after aggregation.

Theorem 4.2.2 (Unbiased Estimator)**.** *Let $\hat{W}_g$ denote the global model obtained by aggregating pruned models after layer sampling, and W_g^* be the global model obtained if all layers were pruned at every client (without sampling). The model $\hat{W}_g$ is an unbiased estimator of W_g^* under the layer sampling strategy. Formally:*

$$\mathbb{E}[\hat{W}_g] = W_g^*. \tag{4.24}$$

This holds as long as each layer has an equal probability of being selected across clients during each communication round, ensuring that all layers are eventually represented in the final aggregated model.

Theorem 1 ensures that the global model obtained by layer sampling converges to the same result as if every client had pruned all layers. The key here is the randomness of the layer selection process: over multiple communication rounds, the expected contribution of each layer is preserved, resulting in an unbiased estimate of the fully pruned model. This guarantees that our sampling strategy does not introduce systematic errors and that, over time, the sampled model will mirror the performance of the fully pruned model.

OPT-125m												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.45e4	1.36e4	1.17e4	1.86e4	1.74e4	1.62e4	2.16e4	2.09e4	1.96e4	3.22e4	3.22e4	3.03e4
Standalone	37.87	57.38	33.78	62.09	93.14	49.38	237.07	296.61	158.54	1699.62	1907.27	759.18
FedSpaLLM	37.65	57.07	33.75	61.28	92.32	48.96	226.44	287.72	152.96	1414.77	1699.16	739.97
OPT-1.3b												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.90e4	1.92e4	1.75e4	1.77e4	1.73e4	1.67e4	2.14e4	2.23e4	2.01e4	3.00e4	3.10e4	2.97e4
Standalone	18.16	26.78	20.31	23.55	35.93	24.15	62.08	97.46	52.13	1814.44	1756.18	636.56
FedSpaLLM	18.09	26.52	20.13	22.70	34.05	23.72	54.87	82.99	48.05	1653.65	1564.86	598.83
LlaMA-2 7b												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	8.83e4	1.20e5	1.41e5	1.98e5	2.99e5	1.92e5	5.06e4	4.98e4	4.93e4	4.99e4	4.91e4	4.97e4
Standalone	7.13	185.18	9.47	10.71	1255.83	13.52	31.60	6582.79	34.17	124.59	8398.51	110.45
FedSpaLLM	6.71	88.15	9.03	9.02	300.35	11.94	20.14	1371.44	23.28	119.21	3382.99	98.87

Table 4.6: Average perplexity of the client models and perplexity of the global model with 4 clients; the lower the perplexity, the better.

OPT-125m												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.24e4	1.12e4	9.93e3	1.65e4	1.52e4	1.40e4	2.11e4	2.04e4	1.86e4	3.29e4	3.31e4	3.15e4
Standalone	38.40	57.51	33.95	64.52	94.19	50.08	244.90	308.59	158.49	1655.23	1893.9	732.29
FedSpaLLM	38.01	56.66	33.65	63.00	91.50	49.10	217.54	274.06	147.58	1415.78	1611.45	665.36
OPT-1.3b												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.59e4	1.44e4	1.40e4	2.15e4	2.17e4	1.84e4	2.26e4	2.27e4	2.06e4	3.25e4	3.27e4	3.22e4
Standalone	18.51	27.18	20.29	23.99	36.53	24.54	68.70	108.93	56.28	2109.46	2134.22	733.80
FedSpaLLM	18.38	27.89	20.03	23.20	35.60	23.75	63.31	105.37	54.85	1979.77	1989.21	690.86
LlaMA-2 7b												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	2.62e5	1.14e5	2.92e5	6.57e4	6.11e4	6.58e4	6.15e4	6.25e4	6.17e4	4.57e4	4.76e4	4.78e4
Standalone	7.17	193.89	9.43	10.76	1126.63	13.45	30.77	8349.21	33.31	134.08	1.3e4	108.87
FedSpaLLM	6.70	73.67	8.97	9.09	175.98	12.03	21.18	1242.25	24.98	117.73	2819.17	96.44

Table 4.7: Average perplexity of the client models and perplexity of the global model with 8 clients; the lower the perplexity, the better.

4.2.5 Experiments

Experiments Setup. We implement our FedSpaLLM in PyTorch [110] and use the HuggingFace Transformers library [152] for handling models and datasets. All

OPT-125m												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.22e4	1.14e4	9.96e3	1.66e4	1.56e4	1.45e4	2.47e4	2.35e4	2.24e4	3.20e4	3.23e4	2.95e4
Standalone	38.24	57.37	33.91	67.42	98.29	51.92	264.88	326.87	171.83	1624.46	1764.94	777.79
FedSpaLLM	38.00	56.90	33.62	66.41	97.92	51.84	240.18	304.17	167.30	1389.22	1454.49	700.84
OPT-1.3b												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.80e4	1.77e4	1.53e4	1.88e4	1.87e4	1.75e4	2.32e4	2.22e4	2.14e4	2.99e4	3.02e4	2.94e4
Standalone	18.99	27.48	20.76	25.01	37.87	25.49	82.24	138.91	65.53	2472.21	2475.89	873.45
FedSpaLLM	18.20	27.30	20.52	23.93	36.36	22.72	77.51	129.40	63.62	2327.93	2361.86	838.53
LlaMA-2 7b												
Sparsity	50%			60%			70%			80%		
Dataset	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4	WT-2	PTB	C4
Random	1.76e5	2.79e5	2.19e5	8.30e4	7.96e4	7.92e4	9.73e4	2.54e5	1.13e5	5.20e4	4.89e4	4.89e4
Standalone	7.17	188.35	9.41	10.80	969.84	13.44	31.81	6671.71	33.20	137.21	7757.43	106.87
FedSpaLLM	6.75	70.79	9.04	9.33	178.15	12.45	25.09	702.03	27.27	115.91	1791.83	94.87

Table 4.8: Average perplexity of the client models and perplexity of the global model with 16 clients; the lower the perplexity, the better.

pruning experiments are conducted on NVIDIA A100 GPUs. For each client, we utilize SparseGPT [39] to perform pruning. For the calibration data, we follow [39] and use 2048-token segments, randomly chosen from the first shard of the C4 [114] dataset. This represents generic text data crawled from the internet and ensures that our experiments remain zero-shot since no task-specific data is seen during pruning. In addition, we consider a random pruning baseline in which the model is randomly pruned to the target sparsity.

In particular, we perform experiments on the OPT model family [164] and LlaMA-2 model family [141] with 4, 8, and 16 clients. We consider OPT-125m, OPT-1.3b, and LlaMA-2 7b with unstructured sparsity of 50% to 80%. In each communication round, each of the clients receives a copy of the same global model from the server and each client is assumed to utilize 32 calibration samples to perform its own pruning. By evaluating models of varying sizes alongside different number of clients, we can gain a more comprehensive understanding of FedSpaLLM's performances in its scalability and robustness. In terms of metrics, we mainly focus on perplexity, which

is known to be a challenging and stable metric that is well-suited for evaluating the accuracy of compression methods [158, 27], and thus measuring the performances of the compressed models. We consider the test sets of raw-WikiText2 [98] (WT-2) and PTB [93] as well as a subset of the C4 validation data, which are all popular benchmarks in LLM compression literature [158, 109, 39, 133].

Results and analyses

We present the perplexity results in Tables 4.6 to 4.8. In the tables, we report the random pruning baseline, denoted by "Random", the average perplexity of the client models, denoted by "standalone", and the global model, denoted by "FedSpaLLM". From the results, we see that random pruning results in perplexity that are orders of magnitude higher than both standalone and FedSpaLLM. As the model size increases, the performances of random pruning becomes even worse. This suggests that random pruning may have pruned weights that are crucial for maintaining model quality. Comparing standalone and FedSpaLLM, we see that across the models, datasets, and sparsity levels, FedSpaLLM consistently outperforms standalone in achieving lower perplexity. In general, as the target sparsity increases, we see more noticeable improvements in the perplexity of the global model over the client models. This is expected because as the sparsity increases, more information is required to accurately prune the model weights to maintain the model performances. In essence, each of the client models is pruned with the private calibration samples of each client while the global model benefits from the collaborative information from the communicated weights from the clients. As a result, the global model is effectively utilizing a signicantly larger calibration sample set, even though such a centralized calibration sample set is prohibited in FL setting as the client's calibration samples are private. Notably, FedSpaLLM achieves substantially lower perplexity compared to standalone in higher sparsity levels, highlighting the benefits of FedSpaLLM where

the clients collaboratively contribute to the global model with much better performances while the privacy of their own data is well maintained. Furthermore, we can see the improvements in the perplexity of the global model over the client models are particularly significant for the LlaMA-2 model family and the PTB dataset. We observe that, in the case of LlaMA-2 7b, the client models generally struggle with the PTB dataset from sparsity 60% and beyond, regardless of the number of clients. In many of those cases, the global model achieves up to 5 times better perplexity values. This demonstrates the effectiveness of FL in the pruning tasks. In addition, we do not observe there is noticeable trend in perplexity values with varying number of clients and the perplexity values of the global models are comparable regardless of the number of clients participating in FL, when the sparsity level is small.

Chapter 5

Conclusion

5.1 Summary of Research Contributions

This dissertation advances the field of functional-space-guided learning, offering solutions to critical challenges in efficiency, generalization, and scalability for modern machine learning systems. By leveraging insights from functional space, my research has proposed innovative methodologies spanning multi-task learning, continual learning, domain adaptation, and model pruning. This section summarizes the completed works of mine.

Saliency-Regularized Deep Multi-Task Learning (SRDML). Multi-task learning (MTL) provides an effective framework for transferring knowledge across related tasks, yet existing methods often struggle with balancing task-specific and shared representations. SRDML addresses this by employing saliency-based regularization to align functional-space representations. This approach ensures task coherence while preserving individuality, enabling improved generalization and interpretability. Theoretical analyses validate the framework's ability to minimize generalization error, while experiments on synthetic and real-world datasets demonstrate its efficacy over traditional MTL methods.

Saliency-Augmented Memory Completion (SAMC). Continual learning systems often face catastrophic forgetting, where knowledge of earlier tasks is lost as new ones are learned. SAMC introduces a biologically inspired framework that prioritizes task-relevant saliency regions for episodic memory storage. The model employs an adaptive inpainting mechanism to reconstruct missing information during training, enhancing memory efficiency and task retention. Extensive experiments on benchmark datasets showcase SAMC's superior performance in mitigating forgetting while maintaining scalability.

Drift-Aware Dynamic Neural Networks (DRAIN). In dynamic environments, temporal domain shifts pose significant challenges to machine learning models. DRAIN bridges parameter and functional spaces through dynamic adjustment mechanisms, allowing models to adapt to temporal concept drift effectively. This approach employs drift-aware optimization strategies to maintain robust performance across unseen temporal domains. DRAIN's ability to generalize to future domains has been validated through rigorous experimentation on real-world datasets, offering a robust solution for time-sensitive applications.

SparseLLM: Global Pruning for Pre-Trained Language Models. The increasing size of pre-trained language models (LLMs) presents substantial computational and memory challenges. SparseLLM introduces a globally optimized pruning strategy that reduces redundancy in LLMs while maintaining task performance. By leveraging functional-space optimization, the framework decomposes pruning into manageable subproblems, ensuring fast convergence and scalability. Experiments on models such as OPT and LLaMA demonstrate the effectiveness of SparseLLM in achieving high sparsity levels with minimal performance degradation, making it a key contribution to resource-efficient AI.

FedSpaLLM: Federated Pruning of Large Language Models. FedSpaLLM presents a federated sparse language model pruning framework that extends SparseLLM

to address data heterogeneity and resource constraints in FL. It introduces a sparsity-aware aggregation method to integrate heterogeneous client pruning masks, applies structured pruning and gradient compression for improved efficiency, and establishes theoretical guarantees for global sparsity and convergence.

Prompt-Based Domain Discrimination (POND). Adapting models to diverse domains, particularly in time-series data, remains a challenging problem. POND proposes a prompt-tuning approach for multi-source domain adaptation, leveraging functional-space representations to capture domain-specific and invariant features. The method incorporates saliency-guided alignment and domain discrimination loss to enhance fidelity and distinction. POND has achieved state-of-the-art performance on benchmark datasets, establishing itself as a powerful tool for domain adaptation in real-world applications.

5.2 Future Work

5.2.1 Time-aware future generalization with foundation models by function-space modeling

One promising direction is to explore time-aware generalization using foundation models through the lens of function-space modeling. In many real-world scenarios, data distributions evolve over time due to concept drift, seasonality, or long-term shifts. Modeling such temporal dynamics in function space enables us to learn representations that are invariant or adaptable to time-specific patterns. By leveraging large pre-trained foundation models and fine-tuning them in function space, we can better capture temporal dependencies and develop models that generalize across future time periods with improved robustness and interpretability.

This approach also opens new avenues for continual learning, where time is treated

as a fundamental axis along which tasks arrive and evolve. By embedding time-aware priors or temporal regularization in the function space, we can prevent catastrophic forgetting while promoting forward transfer to unseen future domains.

5.2.2 Extending LLM global pruning to structured pruning / MoE pruning

While existing work such as SparseLLM focuses on unstructured global pruning for LLMs, a natural extension is to incorporate structured pruning methods, including block-wise, channel-wise, or layer-wise pruning. Structured sparsity not only improves hardware efficiency during inference but also makes the pruning process more interpretable and manageable. Integrating such structured constraints into the global pruning pipeline in functional space can provide theoretical and empirical gains in performance and deployability.

Another compelling direction is to extend the pruning framework to Mixture-of-Experts (MoE) models. MoE architectures are inherently sparse and modular, but managing expert activation across clients in federated or distributed settings remains challenging. By pruning experts in function space based on their functional contribution to task performance, it is possible to reduce memory and compute costs while preserving the benefits of model specialization. This could also facilitate better routing mechanisms and more scalable expert deployment.

5.2.3 Function-aware evaluations of deep learning models

Traditional evaluation metrics such as accuracy or loss provide limited insight into the underlying functional behavior of models. A future direction is to develop evaluation frameworks that are explicitly aware of the model's function space. For instance, we can measure how sensitive a model's output function is to perturbations

in the input space, or analyze the smoothness, curvature, and generalizability of the function approximated by the network.

These function-aware evaluations can help identify overfitting, detect spurious correlations, or assess model robustness in a principled way. They can also be used to guide training, regularization, and model selection — providing a more holistic understanding of what the model has learned beyond simple predictive performance.

These future directions build upon the principles established in this dissertation while addressing critical gaps in current methodologies. Their exploration will contribute to scalable, resource-efficient, and generalizable AI systems, aligning with the broader vision of deploying AI in diverse and dynamic environments.

5.3 Publications

The research conducted during my doctoral studies has resulted in several publications across top-tier conferences and journals. This section lists the publications grouped by category, including published works and preprints under review.

5.3.1 Published Work

- **FedSpaLLM: Federated Pruning of Large Language Models**
  Guangji Bai, Yijiang Li, Zilinghan Li, Liang Zhao, Kibaek Kim. *NAACL main 2025.*

- **Staleness-Alleviated Distributed GNN Training via Online Dynamic-Embedding Prediction**
  Guangji Bai, Ziyang Yu, Zheng Chai, Yue Cheng, Liang Zhao. *SDM 2025.*

- ***SparseLLM*: Towards Global Pruning for Pre-trained Language Models**

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, Liang Zhao. *NeurIPS 2024*.

- **Prompt-based Domain Discrimination for Multi-source Time Series Domain Adaptation**

  Junxiang Wang*, Guangji Bai*, Wei Cheng, Zhengzhang Chen, Liang Zhao, Haifeng Chen. *KDD 2024*.

- **Continuous Temporal Domain Generalization**

  Zekun Cai, Guangji Bai, Renhe Jiang, Xuan Song, Liang Zhao. *NeurIPS 2024*.

- **Temporal Domain Generalization with Drift-Aware Dynamic Neural Networks**

  Guangji Bai*, Chen Ling*, Liang Zhao. *ICLR 2023* (Oral, top 1%).

- **Saliency-Augmented Memory Completion for Continual Learning**

  Guangji Bai, Chen Ling, Yuyang Gao, Liang Zhao. *SDM 2023*.

- **Sign-Regularized Multi-task Learning**

  Guangji Bai, Johnny Torres, Junxiang Wang, Liang Zhao, Cristina Abad, Carmen Vaca. *SDM 2023*.

- **Saliency-Regularized Deep Multi-Task Learning**

  Guangji Bai, Liang Zhao. *KDD 2022*.

5.3.2 Preprints and Under Review

- **Beyond Efficiency: A Systematic Survey of Resource-Efficient Large Language Models**

  Guangji Bai et al., *Under review at CSUR*.

- **Saliency-Guided Hidden Associative Replay for Continual Learning**

  Guangji Bai, Qilong Zhao, Xiaoyang Jiang, Yifei Zhang, Liang Zhao. *AMHN@NeurIPS 2023*.

- **Distributed Graph Neural Network Training with Periodic Historical Embedding Synchronization**

  Zheng Chai*, Guangji Bai*, Liang Zhao, Yue Cheng. *Preprint.*

5.3.3 Collaborative Work

- **Quantifying Uncertainty in Graph Neural Network Explanations**

  Junji Jiang, Chen Ling, Hongyi Li, Guangji Bai, Xujiang Zhao, Liang Zhao. *Frontiers in Big Data, 2024.*

- **Estimation of Daily PM2.5 Concentration Using a Geographically Weighted Neural Network**

  Yun Li, Guangji Bai, Dazhou Yu, Liang Zhao. *AGU23, 2023.*

- **Res: A Robust Framework for Guiding Visual Explanation**

  Yuyang Gao, Tong Steven Sun, Guangji Bai, Siyi Gu, Sungsoo Ray Hong, Liang Zhao. *KDD 2022.*

*Remark: * denotes co-first author with equal contribution.*

Appendix A

Theory Proof of SRDML

In this section, we provide the formal proof for all the theories presented in Saliency-regularized Deep Multi-task Learning paper.

A.1 Proof of Theorem 2.1.1

Proof. Suppose $\mathcal{X} \subseteq \mathbb{R}^K$ is an open set and $f_1, f_2 : \mathcal{X} \to \mathbb{R}$, where both functions are differentiable and equal to zero at the origin.

"$\Longrightarrow$": This direction is obvious, since two exactly the same functions will have the same gradient at any input point.

"$\Longleftarrow$": Given $\nabla f_1(\mathbf{x}) = \nabla f_2(\mathbf{x})$, we know that

$$\partial f_1/\partial x_k = \partial f_2/\partial x_k,\ k = 1, 2, \cdots, K,\ \forall\, \mathbf{x} \in \mathcal{X} \tag{A.1}$$

For arbitrary k, by $\partial f_1/\partial x_k = \partial f_2/\partial x_k$, we know that

$$\exists\, c_k(x_1, \cdots, x_{k-1}, x_{k+1}, \cdots, x_K),\ s.t.,\ f_1 = f_2 + c_k \tag{A.2}$$

Meanwhile, notice $\forall\, l \neq k, \quad \partial c_k/\partial x_l = 0$ (otherwise, contradiction!) Hence, $dc_k = 0$

and we know c_k is a constant. Also, the value of c_k does not depend on k since for all k, l, we have $f_1 - f_2 = c_k = c_l$, thus there exists a constant c such that $f_1 = f_2 + c$. Finally, by the boundary condition that $f_1(\mathbf{0}) = f_2(\mathbf{0}) = 0$, we know that $c = 0$, i.e., $f_1 = f_2$, which finishes the proof. □

A.2 Proof of Theorem 2.1.5

In this section, we provide the proof of our model's generalization error bound. First, we introduce some definitions and lemmas that will be continuously used, and at the end of this section, we present the proof for Theorem 2.1.5.

In general, we will use γ to denote a generic vector of i.i.d. standard normal variables, whose dimension will be clear in context. In addition, without further specification, we will use K, T, n to denote the (flattened) dimension of the output space from the feature extraction function h, the number of tasks, and the number of training samples, respectively. We denote the representation class for task-specific function f and representation extraction function h as $\mathcal{F}$ and $\mathcal{H}$, respectively. Two hypothesis classes here can be very general, and the only assumption here is that $\forall f \in \mathcal{F}$, f has a Lipschitz constant at most L, for any positive L.

Definition A.2.1. *Given a set $V \subseteq \mathbb{R}^n$, define the Gaussian average of V as*

$$G(V) \coloneqq \mathbb{E} \sup_{v \in V} \langle \gamma, v \rangle = \mathbb{E} \sup_{v \in V} \sum_{i=1}^{n} \gamma_i v_i \tag{A.3}$$

As mentioned in section 3.1 in the main paper, we denote the feature representation learning part as function $h \in \mathcal{H}$. As we will see later, the complexity of the representation class $\mathcal{H}$ is important in our proof for the error bound, so we define a measure of its complexity by the Gaussian average.

Definition A.2.2. *Given observed input data $\mathbf{X} \in \mathcal{X}^{Tn}$, define a random set $\mathcal{H}(\mathbf{X}) \subseteq$*

$\mathbb{R}^{KTn}$ *by*

$$\mathcal{H}(\mathbf{X}) \coloneqq \left\{(h_k(\mathbf{x}_i^t)) : h \in \mathcal{H}\right\}. \tag{A.4}$$

The Gaussian average over $\mathcal{H}(\mathbf{X})$ *can be defined accordingly as*

$$G(\mathcal{H}(\mathbf{X})) = \mathbb{E}[\sup_{h\in\mathcal{H}} \sum_{kti}^{K,T,n} \gamma_{kti} h_k(\mathbf{x}_{ti})|\mathbf{x}_{ti}] \tag{A.5}$$

The following lemmas will be useful in our later proof, and we will introduce them here in advance.

Lemma A.2.3. $\forall\ A, C \in \mathbb{R}^{m\times n}$ *and* $B \in \mathbb{R}^{m\times m}$,

$$tr(A^\intercal BC) = \sum_{i,j}^{m} B_{ij} \sum_{k=1}^{n} A_{ik}C_{jk}. \tag{A.6}$$

Lemma A.2.4. *Suppose* $\mathcal{X} \subseteq \mathbb{R}^K$ *is an open set, and two differentiable functions* $f_1, f_2\ : \mathcal{X} \to \mathbb{R}$. $\forall x \in \mathcal{X}$, *if*

$$\exists\ B > 0,\ \ s.t\ \|\nabla f_1(x) - \nabla f_2(x)\| < B \tag{A.7}$$

then

$$\lim_{\Delta x\to 0} \left| \frac{f_1(x+\Delta x) - f_1(x)}{\|\Delta x\|} - \frac{f_2(x+\Delta x) - f_2(x)}{\|\Delta x\|} \right| < B. \tag{A.8}$$

Now we begin to prove our Theorem 2.1.5.

Proof. We start our proof from the generalization error bound for multi-task learning in Maurer et al. [95], which also considers the shared representation learning part but without the regularization on task-specific part.

Step 1

By Theorem 9 from Maurer et al. [95], $\forall \delta \geq 0$, with probability at least $1-\delta$ in

the samples $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$, for all $h \in \mathcal{H}$ and $\mathbf{f} = (f_1, \cdots, f_T) \in \mathcal{F}_\epsilon$, we have that

$$\mathcal{E}(h, f_1, \cdots, f_T) - \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)}) \leq \frac{\sqrt{2\pi}}{nT}G(S) + \sqrt{\frac{9\ln(2/\delta)}{2nT}}, \quad \text{(A.9)}$$

where $S = \{(\mathcal{L}(f_t(h(\mathbf{x}_i)), \mathbf{y}_i^{(t)})) : \mathbf{f} \in \mathcal{F}_\epsilon, h \in \mathcal{H}\} \subseteq \mathbb{R}^{Tn}$. By our assumption over the Lipschitz of the loss function $\mathcal{L}$ and Corollary 11 from [95], we have $G(S) \leq G(S')$, where $S' = \{(f_t(h(\mathbf{x}_i), \mathbf{y}_i^{(t)})) : \mathbf{f} \in \mathcal{F}_\epsilon, h \in \mathcal{H}\} \subseteq \mathbb{R}^{Tn}$.

By Theorem 12 from Maurer et al. [95], for universal constant c_1' and c_2',

$$G(S') \leq c_1' L(\mathcal{F}_\epsilon) G(\mathcal{H}(\mathbf{X})) + c_2' D(\mathcal{H}(\mathbf{X})) Q(\mathcal{F}_\epsilon) + \min_{v \in V} G(\mathcal{F}(v)), \quad \text{(A.10)}$$

where $L(\mathcal{F}_\epsilon)$ is the Lipschitz of function class $\mathcal{F}_\epsilon$, $D(\mathcal{H}(\mathbf{X}))$ is the diameter of set $\mathcal{H}(\mathbf{X})$, and

$$Q(\mathcal{F}_\epsilon) = \sup_{\mathbf{y} \neq \mathbf{y}'} \mathbb{E}_\gamma \sup_{\mathbf{f} \in \mathcal{F}_\epsilon} \frac{\langle \gamma, \mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{y}') \rangle}{\|\mathbf{y} - \mathbf{y}'\|}. \quad \text{(A.11)}$$

For each term **except** $Q(\mathcal{F}_\epsilon)$ in the right hand side of (A.10), it can be upper-bounded or replaced by another term as shown in Theorem 13 of Maurer et al. [95], and we simply follow their proof for those terms. The main difference or the main contribution of our theoretical analysis comes from bounding the term $Q(\mathcal{F}_\epsilon)$, which you will see later connects our regularization and the generalization error bound. Hence, for the rest of proof we will show how to bound $Q(\mathcal{F}_\epsilon)$ and bridge our regularizer with the error bound.

Step 2

Given multi-sample $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^{KTn}$, where $\mathbf{y} \neq \mathbf{y}'$, $\mathbf{y} = (y_{ti})$ with $y_{ti} \in \mathbb{R}^K$, define

$$\hat{Q}(\mathcal{F}_\epsilon) \coloneqq \mathbb{E}_\gamma \sup_{(f_1, f_2, \cdots, f_T) \in \mathcal{F}_\epsilon} \sum_{t,i}^{T,n} \gamma_{ti} \cdot \frac{f_t(y_{ti}) - f_t(y'_{ti})}{\|y_{ti} - y'_{ti}\|}. \quad \text{(A.12)}$$

Also, define two matrices Γ and Δ as $\Gamma = (\gamma_{ti})$ and $\Delta = ((f_t(y_{ti}) - f_t(y'_{ti}))/\|y_{ti} - y'_{ti}\|)$,

where $\Gamma, \Delta \in \mathbb{R}^{T\times n}$.

We first apply Cauchy-Schwarz Inequality as follows:

$$
\begin{aligned}
&\mathbb{E}_\gamma \sup_{(f_1,f_2,\cdots,f_T)\in\mathcal{F}_\epsilon} \sum_{t,i}^{T,n} \gamma_{ti}\cdot\frac{f_t(y_{ti})-f_t(y'_{ti})}{\|y_{ti}-y'_{ti}\|} \\
=&\mathbb{E}_\gamma \sup_{(f_1,\cdots,f_T)\in\mathcal{F}_\epsilon} \langle vec(\Gamma), vec(\Delta)\rangle \\
=&\mathbb{E}_\gamma \sup_{(f_1,\cdots,f_T)\in\mathcal{F}_\epsilon} tr(\Gamma^\intercal\cdot\Delta) \quad \text{(Lemma Theorem A.2.3)} \\
=&\mathbb{E}_\gamma \sup_{(f_1,\cdots,f_T)\in\mathcal{F}_\epsilon} tr(\Gamma^\intercal\cdot A^{-1/2}A^{1/2}\cdot\Delta) \\
=&\mathbb{E}_\gamma \sup_{(f_1,\cdots,f_T)\in\mathcal{F}_\epsilon} \langle vec(\Gamma^\intercal\cdot A^{-1/2}), vec(A^{1/2}\cdot\Delta)\rangle \\
\leq&\mathbb{E}_\gamma \sup_{(f_1,\cdots,f_T)\in\mathcal{F}_\epsilon} \left\|\Gamma^\intercal\cdot A^{-1/2}\right\|_2\cdot\left\|A^{1/2}\cdot\Delta\right\|_2 \\
&\text{(Cauchy-Schwartz Inequality)} \\
=&\sup_{(f_1,\cdots,f_T)\in\mathcal{F}_\epsilon} \left\|A^{1/2}\cdot\Delta\right\|_2\cdot\mathbb{E}_\gamma\left\|\Gamma^\intercal\cdot A^{-1/2}\right\|_2,
\end{aligned}
\tag{A.13}
$$

where A is any symmetric positive definite matrix with corresponding size. From the second line to the third line we utilize our Lemma Theorem A.2.3 with B equal to identity, A is Γ and C is Δ. The ℓ_2-norm here is basically same as the standard vector norm, i.e., the squared root of the summation over the square of each element within the matrix. We omit the vectorization operator inside the norm for simpler and more compact notation.

By doing so, we separate the random part and "sup" part in $\hat{Q}(\mathcal{F}_\epsilon)$ and our goal now is to bound them respectively.

Step 2.a

Using Jensen's Inequality, we can bound the random part:

$$
\begin{aligned}
&\{\mathbb{E}_\gamma \left\|\Gamma^\intercal \cdot A^{-1/2}\right\|_2\}^2 \\
&\leq \mathbb{E}_\gamma[\left\|\Gamma^\intercal \cdot A^{-1/2}\right\|_2^2] \quad \text{(Jensen's Inequality)} \\
&= \mathbb{E}_\gamma[\langle vec(\Gamma^\intercal \cdot A^{-1/2}), vec(\Gamma^\intercal \cdot A^{-1/2})\rangle] \\
&= \mathbb{E}_\gamma[tr((\Gamma^\intercal \cdot A^{-1/2})(\Gamma^\intercal \cdot A^{-1/2})^\intercal)] \\
&= \mathbb{E}_\gamma[tr(\Gamma^\intercal A^{-1}\Gamma)] \quad \text{(A's symmetry)} \\
&= \sum_{p=1}^{T}\sum_{q=1}^{T} A_{pq}^{-1} \sum_{i=1}^{n} \mathbb{E}[\gamma_{pi} \cdot \gamma_{qi}] \quad \text{(Lemma Theorem A.2.3)} \\
&= \sum_{t=1}^{T} A_{tt}^{-1} \sum_{i=1}^{n} \mathbb{E}\{[\gamma_{ti}]^2\} \quad (\gamma \text{ is i.i.d } \mathcal{N}(0,1)). \\
&= n \cdot tr(A^{-1}).
\end{aligned}
\tag{A.14}
$$

Step 2.b

It is a little bit trickier to bound the "sup" part. Notice the regularization term in (2.3) is actually a special case of graph regularization or laplacian regularization (Evgeniou et al. [36], Maurer [94]). The basic idea of our regularizer is to enforce the distance between similar tasks' input gradient to be small. More specifically, given $y \in \mathbb{R}^{Kn}$ and replace the $dist()$ in (2.3) with standard ℓ_2-norm, we have

$$
\frac{1}{2}\sum_{p,q=1}^{T} \omega_{pq} \left\|\nabla f_p(y) - \nabla f_q(y)\right\|^2 = \sum_{p,q=1}^{T} \mathbb{L}_{pq} \cdot \langle \nabla f_p(y), \nabla f_q(y)\rangle, \tag{A.15}
$$

where $\mathbb{L}$ is the Laplacian matrix of the graph with T vertices and edge-weights $\{\omega\}$ (each weight is between 0 and 1) as given value, and $\nabla f_t(y) = (\nabla f_t(y_1)^\intercal, \nabla f_t(y_2)^\intercal, \cdots, \nabla f_t(y_n)^\intercal)^\intercal$ is the concatenated vector of function f_t's input gradient at each sample $y_i \in \mathbb{R}^K$.

We slightly modify the form in (A.15) by adding an identity matrix, i.e.,

$$\frac{1}{2}\sum_{p,q=1}^{T}\omega_{pq}\left\|\nabla f_p(y)-\nabla f_q(y)\right\|^2+\eta\sum_{p=1}^{T}\left\|\nabla f_p(y)\right\|^2=\sum_{p,q=1}^{T}(\mathbb{L}+\eta\mathbb{I})_{pq}\cdot\langle\nabla f_p(y),\nabla f_q(y)\rangle, \tag{A.16}$$

where $\eta>0$ is any positive real number and $\mathbb{I}$ is the identity matrix. The correctness of (A.15) and (A.16) can be validated by following section 3 in Maurer [94].

Recall the definition of $\mathcal{F}_\epsilon$ in (2.5). $\forall\,(f_1,f_2,\cdots,f_T)\in\mathcal{F}_\epsilon$ we have

$$\frac{1}{2}\sum_{p,q=1}^{T}\left\|\nabla f_p(y)-\nabla f_q(y)\right\|^2=\sum_{p\neq q}\left\|\nabla f_p(y)-\nabla f_q(y)\right\|^2\leq\alpha. \tag{A.17}$$

Since $\{\omega\}$ are between 0 and 1, there exists positive value B that satisfies

$$\sum_{p,q=1}^{T}\omega_{pq}\left\|\nabla f_p(y)-\nabla f_q(y)\right\|^2\leq B^2 \tag{A.18}$$

By Lemma Theorem A.2.4, we know that as $y'\to y$,

$$\sum_{p,q=1}^{T}\omega_{pq}\cdot\left\|\frac{f_p(y)-f_p(y')}{\|y-y'\|}-\frac{f_q(y)-f_q(y')}{\|y-y'\|}\right\|^2\leq B^2 \tag{A.19}$$

Replace the gradient by differential in (A.16), the equation still holds, i.e.,

$$
\begin{aligned}
&\frac{1}{2}\sum_{p,q=1}^{T}\omega_{pq}\left\|\frac{f_p(y)-f_p(y')}{\|y-y'\|}-\frac{f_q(y)-f_q(y')}{\|y-y'\|}\right\|^2 \\
&+\eta\sum_{p=1}^{T}\left\|\frac{f_p(y)-f_p(y')}{\|y-y'\|}\right\|^2 \\
=&\sum_{p,q=1}^{T}(\mathbb{L}+\eta\mathbb{I})_{pq}\cdot\langle\frac{f_p(y)-f_p(y')}{\|y-y'\|},\frac{f_q(y)-f_q(y')}{\|y-y'\|}\rangle \qquad \text{(A.20)}\\
=&tr(\Delta^\intercal\cdot(\mathbb{L}+\eta\mathbb{I})\cdot\Delta) \quad \text{(Lemma Theorem A.2.3)}\\
=&\left\|\Delta^\intercal\cdot(\mathbb{L}+\eta\mathbb{I})^{1/2}\right\|_2^2\\
=&\left\|(\mathbb{L}+\eta\mathbb{I})^{1/2}\cdot\Delta\right\|_2^2 \quad \text{(Symmetry of } \mathbb{L}+\eta\mathbb{I})
\end{aligned}
$$

The second last equation can be derived by following (A.14) inversely. Just clarify a slight difference in notation, where in (A.13) we use $\mathbf{y}$ but in (A.20) we use y without bold. In section 3.1 of our main paper, we already assume the input feature is shared cross different tasks, which corresponds to the definition of y (the last sentence before (A.15)), and this is a special case of $\mathbf{y}$. Hence, this difference in notation has no influence on our proof.

The result in (A.20) indicates that by setting $A=\mathbb{L}+\eta\mathbb{I}$, we can potentially link our regularization as in (A.15) and the "sup" part in (A.13). The remaining proof will show this formally.

By controlling the level of η, we want to let the first term in the first line of (A.20) dominate. Since we assume the Lipschitz of function class $\mathcal{F}$, we know that $\forall t$,

$$
\|f_t(y)-f_t(y')\|^2\le L^2\cdot\sum_{i=1}^{n}\|y_i-y_i'\|^2=L^2\cdot\|y-y'\|^2. \qquad \text{(A.21)}
$$

Hence, for the second term (without η) in the first line of (A.20) it can be upper-bounded by $T\cdot L^2$. We set $\eta=B^2/(2TL^2)$, and the second term (with η) is no larger than $B^2/2$. Combining with (A.19) and notice the $\frac{1}{2}$ before the first term in the first line of (A.20), we have

$$\left\|(\mathbb{L}+\eta\mathbb{I})^{1/2}\cdot\Delta\right\|_2^2 \leq B^2. \tag{A.22}$$

By plugging (A.14) and (A.22) into the last equation of (A.13), and notice these two conclusions hold for arbitrary $\mathbf{f}\in\mathcal{F}_\epsilon$, we have

$$\begin{aligned}
\hat{Q}(\mathcal{F}_\epsilon) &= \sup_{\mathbf{f}\in\mathcal{F}_\epsilon}\left\|A^{1/2}\cdot\Delta\right\|_2\cdot\mathbb{E}_\gamma\left\|\ \Gamma^\intercal\cdot A^{-1/2}\right\|_2 \\
&\leq B\cdot\sqrt{n\cdot tr((\mathbb{L}+\eta\mathbb{I})^{-1})} \\
&= B\sqrt{n}\cdot\sqrt{\sum_{i=2}^{T}\frac{1}{\lambda_i+\eta}+\frac{1}{\eta}} \\
&\leq B\sqrt{n}\cdot\sqrt{\frac{T}{\lambda_{min}}+\frac{1}{\eta}} \\
&\leq c_\eta B\sqrt{n}\cdot\sqrt{\frac{T}{\lambda_{min}}},
\end{aligned} \tag{A.23}$$

where $\lambda_2,\lambda_3,\cdots,\lambda_T$ are the non-zero eigenvalues of $\mathbb{L}$ in ascending order, with $\lambda_1=0$ having multiplicity 1 that we used connectivity of the graph (Maurer [94] section 3), and λ_{min} corresponds to the minimal non-zero eigenvalue of $\mathbb{L}$, i.e. λ_2 here. c_η is a positive constant determined by η.

Step 3

Finally, notice (A.23) holds for arbitrary sample $\mathbf{y}$ and $\mathbf{y}'$, we have

$$Q(\mathcal{F}_\epsilon)=\sup_{\mathbf{y}\neq\mathbf{y}'}\hat{Q}(\mathcal{F}_\epsilon)\leq c_\eta B\sqrt{n}\cdot\sqrt{\frac{T}{\lambda_{min}}} \tag{A.24}$$

Substitute $Q(\mathcal{F}_\epsilon)$ by (A.24) (let c_η be absorbed into universal constant c_2) and plug it into (A.10), then put everything in (A.10) back to (A.9) will give the result of Theorem 2.1.5.

□

A.3 Proof of Section 4.2

Natural generalization of shallow MTL

Proof. Basically, when the feature extraction function h is identity function and each task-specific function f_t, $t = 1, 2, \cdots, T$ are linear functions, we know for any input $x \in \mathcal{X}$,

$$h(x) = x, \quad \nabla f_t(x) = w_t, \ \forall t \tag{A.25}$$

where w_t is the model parameter of linear model f_t. Hence, denote $W = [w_1; w_2; \cdots; w_T]$ and take the $dist()$ function in Eq. 2.3 to be inner product, by Lemma A.2.3 we have

$$\begin{aligned} \sum\nolimits_{i,j} \omega_{ij} \cdot dist(\nabla f_i(x), \nabla f_j(x)) &= \sum\nolimits_{i,j} \omega_{ij} \cdot \langle w_i, w_j \rangle \\ &= tr(W^\intercal \Omega W) \end{aligned} \tag{A.26}$$

where $\Omega = (\omega_{ij})$. Let Ω to be either Θ^{-1} or Σ^{-1} as in section 4.2 can finish the proof.

□

Relation with deep MTL

Proof. First, we define two hyperparameters:

- λ: The coefficient of our regularizer in SRDML
- ℓ: The index of the layer before which the model is shared cross different tasks.

Case 1. If $\lambda = 0$ and $1 < \ell < L$, where L (please differentiate this L with that for Lipschitz constant) denotes the total number of layers, our SRDML has no regularization and is simply equivalent to hard-parameter sharing.

Case 2. If $\lambda > 0$ and $\ell = 1$, each layer in our SRDML is separate for different tasks and the regularization is posed on all the layers, which is equivalent to soft-parameter sharing.

□

Appendix B

Theory Proof of SAMC

In this section, we provide the formal proof of the theories presented in the paper of Saliency-Augmented Memory Completion of Continual Learning.

Proof of Lemma 2.2.2

Proof. Denote $\theta = \theta^{(t-1)} - \alpha \cdot g$, where step size $\alpha > 0$, θ^{t-1} is the state of model parameter of f_θ after task $t-1$.

By Taylor Expansion of $\ell(f_\theta, \tilde{\mathcal{M}}_k)$ at $\theta^{(t-1)}$,

$$\ell(f_\theta, \tilde{\mathcal{M}}_k) = \ell(f_{\theta^{(t-1)}}, \tilde{\mathcal{M}}_k) + \left\langle \frac{\partial}{\partial \theta}\bigg|_{\theta^{(t-1)}} \ell(f_\theta, \tilde{\mathcal{M}}_k), \theta - \theta^{(t-1)} \right\rangle + \mathcal{O}(\|\theta - \theta^{(t-1)}\|^2) \tag{B.1}$$

where we expand the loss function up to the first-order, and the last term on right-hand side is the residual.

By some simple manipulation, we can get:

$$\ell(f_\theta, \tilde{\mathcal{M}}_k) - \ell(f_{\theta^{(t-1)}}, \tilde{\mathcal{M}}_k) = \left\langle \frac{\partial}{\partial \theta}\Big|_{\theta^{(t-1)}} \ell(f_\theta, \tilde{\mathcal{M}}_k), \theta - \theta^{(t-1)} \right\rangle + \mathcal{O}(\|\theta - \theta^{(t-1)}\|^2)$$
$$= -\alpha \cdot \langle g, \tilde{g}_k \rangle + \mathcal{O}(\|\theta - \theta^{(t-1)}\|^2). \tag{B.2}$$

If we let $\alpha \to 0^+$, which means setting α to approach zero from the positive side, the residual will shrink to zero quickly. Hence, we have:

$$\ell(f_\theta, \tilde{\mathcal{M}}_k) - \ell(f_{\theta^{(t-1)}}, \tilde{\mathcal{M}}_k) \approx -\alpha \cdot \langle g, \tilde{g}_k \rangle. \tag{B.3}$$

□

Proof of Lemma 2.2.3

Proof. Given input $x \in \mathbb{R}^d$ and saliency map $M \in \mathbb{R}^d$, we generate $\tilde{x}$ by:

- Drop any x_i such that $M_i < \mu$, $i = 1, 2, \cdots, d$.
- Inpaint each element of x_i if it is dropped in step 1.
- Copy each element of x_i if it is not dropped in step 1.
- Combine step 2 and step 3 gives $\tilde{x}$.

Hence, we can find that

$$\|x - \tilde{x}\|_2^2 = \sum_{i=1}^{d} (x_i - \tilde{x}_i)^2 = (\sum_{i \in \text{dropped}} + \sum_{i \in \text{kept}})(x_i - \tilde{x}_i)^2 = \sum_{i \in \text{dropped}} (x_i - \tilde{x}_i)^2 \tag{B.4}$$

As a result,

$$
\begin{aligned}
err(x, \tilde{x}) &\coloneqq |y_x^c - y_{\tilde{x}}^c| \\
&= |f_\theta^c(x) - f_\theta^c(\tilde{x})| \\
&\approx |\langle \frac{\partial}{\partial \theta} f_\theta^c(x), \tilde{x} - x \rangle| \quad \text{(Taylor Expansion of } f_\theta^c \text{ at } x) \\
&\leq \| \frac{\partial}{\partial \theta} f_\theta^c(x) \| \cdot \| \tilde{x} - x \| \quad \text{(Cauchy-Schwartz Inequality)} \\
&\leq \mu \cdot \Delta x \cdot \sqrt{d}
\end{aligned} \tag{B.5}
$$

The last inequality holds since $\frac{\partial}{\partial \theta} f_\theta^c(x)$ is essentially the saliency of prediction f_θ^c and is bounded by μ for each dropped position i. On the other hand, $\|\tilde{x} - x\|$ is upper bounded by the following Eq. (B.4) and the definition of Δx.

□

Now we present the proof of Theorem 2.2.4.

Proof of Theorem 2.2.4

Proof. First, notice that $\langle g, g_k \rangle = \langle g, g_k + \tilde{g}_k - \tilde{g}_k \rangle = \langle g, g_k - \tilde{g}_k \rangle + \langle g, \tilde{g}_k \rangle$. Since $\langle g, \tilde{g}_k \rangle > 0$ by our assumption, it suffices to show

$$
\forall\, \epsilon > 0, \text{ we can find appropriate } \tilde{x}, \text{ s.t. } |\langle g, g_k - \tilde{g}_k \rangle| < \epsilon \tag{B.6}
$$

By Cauchy-Schwarz Inequality,

$$
|\langle g, g_k - \tilde{g}_k \rangle| \leq \|g\| \cdot \|g_k - \tilde{g}_k\| \tag{B.7}
$$

By Lipschitz smoothness on ℓ, we have $\|g\| \leq L$. In addition, by Lemma 2 and Lipschitz continuity of $\partial \ell / \partial \theta$, we have $\|g_k - \tilde{g}_k\| \leq L_\theta \cdot \mu \cdot \Delta x \cdot \sqrt{d}$. Hence, given $\epsilon > 0$, we can always find small enough μ and Δx such that $|\langle g, g_k - \tilde{g}_k \rangle| < \epsilon$.

□

Appendix C

Theory Proof of DRAIN

In this chapter, we provide the formal proof of theorems in the work of DRAIN.

C.1 Proof for Theorem 3.1.4

Proof. By definition of the predictive distribution,

$$\begin{aligned} P(\hat{y}|\mathbf{x}_{T+1}, \mathcal{D}_{1:T}) &= \int P(\hat{y}|\mathbf{x}_{T+1}, \omega_{T+1}) P(\omega_{T+1}|\mathcal{D}_{1:T}) d\omega_{T+1} \\ &= \int P(\hat{y}|\mathbf{x}_{T+1}, \omega_{T+1}) P(\omega_{T+1}|\omega_{1:T}) P(\omega_{1:T}|\mathcal{D}_{1:T}) d\omega_{1:T+1} \end{aligned} \tag{C.1}$$

Our goal is to prove that the variance of this predictive distribution for our proposed method, online baseline, and offline baseline follows the inequality as in Theorem 3.1.4.

Ours v.s. Online Baseline

Here we prove that $\mathrm{Var}(\mathcal{M}_{\mathrm{ours}}) < \mathrm{Var}(\mathcal{M}_{\mathrm{on}})$.

Notice that the first term on the right hand side of Eq. C.1, namely $P(\hat{y}|\mathbf{x}_{T+1}, \omega_{T+1})$, corresponds to deployment of the model with parameter ω_{T+1} on the future domain $\mathcal{D}_{T+1}$, hence the variance of $P(\hat{y}|\mathbf{x}_{T+1}, \omega_{T+1})$ only depends on the noise or randomness coming from $\mathbf{x}_{T+1}$ as long as ω_{T+1} is given. In other words, the uncertainty coming

from $P(\hat{y}|\mathbf{x}_{T+1}, \omega_{T+1})$ can be cancelled for both methods since we are considering the same set of domains. Now the problem reduces to proving that the variance of the second and third terms on the right-hand side of Eq. C.1 for our model is smaller than those for the online baseline.

Notice that

$$\begin{aligned} & P(\omega_{1:T}|\mathcal{D}_{1:T}) \\ = & \int_{\Theta} P(\omega_1|\mathcal{D}_1) \cdot P(\omega_2|\omega_1, \mathcal{D}_2, \theta_0) \cdot P(\theta_1|\omega_1, \omega_2, \theta_0) \cdot P(\omega_3|\omega_2, \mathcal{D}_3, \theta_1) \cdot P(\theta_2|\omega_2, \omega_3, \theta_1) \\ & \cdots P(\omega_T|\omega_{T-1}, \mathcal{D}_T, \theta_{T-2}) \cdot P(\theta_{T-1}|\omega_{T-1}, \omega_T, \theta_{T-2}) d\theta_{0:T-1}, \end{aligned} \tag{C.2}$$

where θ is the parameter of the parameterized function to approximate the ground-truth drift of ω, as defined in Assumption 3.1.2. For example, $P(\omega_1|\mathcal{D}_1)$ denotes that we train the model on the very first domain and $P(\omega_2|\omega_1, \mathcal{D}_2, \theta_0)$ denotes that we continue to train the model on the second domain but with initialization of ω_2 as $q_{\theta_0}(\omega_1)$ where ω_1 is learned from the previous domain and q_{θ_0} is trying to capture the conditional probability or drift between ω_2 and ω_1, i.e., $P(\omega_2|\omega_1)$. In our Bayesian framework, we treat q_θ as a learnable function (e.g., LSTM unit in our proposed method) and we use the subscript of θ to differentiate the status of θ after the training on each domain. In other words, q_θ will be updated after the training on each domain (at least for our method). Notice that θ_0 always denotes the parameter initialization as in Assumption 3.1.2.

By Bayes' rule, we have:

$$P(\omega_{t+1}|\omega_t, \mathcal{D}_{t+1}, \theta_{t-1}) \propto \underbrace{P(q_{\theta_{t-1}}(\omega_t))}_{\text{prior on } \omega_{t+1}} \cdot \underbrace{P(\mathcal{D}_{t+1}|\omega_{t+1})}_{\text{likelihood}}, \tag{C.3}$$

where $P(q_{\theta_{t-1}}(\omega_t))$ can be regarded as the prior of ω_{t+1} because as we mentioned $q_{\theta_{t-1}}$ denotes the initialization of ω_{t+1} before we train the model on domain $\mathcal{D}_{t+1}$, and

$P(\mathcal{D}_{t+1}|\omega_{t+1})$ corresponds to the likelihood of training ω_{t+1} on $\mathcal{D}_{t+1}$. In addition,

$$\begin{aligned} P(\theta_t|\omega_t, \omega_{t+1}, \theta_{t-1}) \propto & P(\theta_{t-1}) \cdot P(\omega_t, \omega_{t+1}|\theta_t) \\ \propto & P(\theta_{t-2}) \cdot P(\omega_{t-1}, \omega_t|\theta_{t-1}) \cdot P(\omega_t, \omega_{t+1}|\theta_t) \\ & \cdots \\ \propto & \underbrace{P(\theta_0)}_{\text{prior on } \theta} \cdot \underbrace{\prod_{i=1}^{t} P(\omega_i, \omega_{i+1}|\theta_i)}_{\text{likelihood}}, \end{aligned} \tag{C.4}$$

for any $t = 1, 2, 3, \cdots, T-1$. In the equation above, this time the prior is over parameter θ and ω_i, ω_{i+1} can be regarded as the "training data" for θ_i.

For the online baseline, since it only keeps one-step finetuning of the model and does not learn how ω_t evolves, the θ_t for the online baseline is always equal to the prior, i.e., $\theta_t = \theta_0$. In other words, $P(q_{\theta_{t-1}}(\omega_t)) = P(q_{\theta_0}(\omega_t))$ and $P(\theta_t|\omega_t, \omega_{t+1}, \theta_{t-1}) = P(\theta_0)$, $\forall\ t$ for the online baseline.

Since we follow the standard routine and assume all distributions are Gaussian, by the Bayesian Theorem, we know that the posterior distribution always has a variance smaller than the prior distribution under the expectation, i.e.,

$$\mathbb{E}\big[Var(\theta_t|\omega_t, \omega_{t+1}, \theta_{t-1})\big] < Var(\theta_0), \tag{C.5}$$

which proves that our method has smaller variance in terms of Eq. C.4. On the other hand, since the second term on the right-hand side of Eq. C.3 is the same for both methods, and for the first term $P(q_{\theta_{t-1}}(\omega_t))$, by our Assumption 3.1.2 we know that for baseline $\Pr(q_{\theta_{t-1}}(\omega_t)) = \Pr(q_{\theta_0}(\omega_t))$ so the variance is basically σ_{θ_0}. For our method, after each training step across a new domain, our θ will get updated and achieve smaller variance (because of the posterior variance of the Gaussian), so we also prove that our method has smaller variance in terms of Eq. C.3. Two parts combined prove that our method has smaller variance in the third term of Eq. C.1,

namely $P(\omega_{1:T}|\mathcal{D}_{1:T})$.

The last step is to compare the variance from the second term in Eq. C.1, namely $P(\omega_{T+1}|\omega_{1:T})$. For online baseline, basically it uses the parameter from the last training domain, i.e., ω_T as the final model on the future domain, i.e., $P(\omega_{T+1}|\omega_{1:T}) = P(q_{\theta_0}(\omega_T))$.

On the other hand, for our method we have $P(\omega_{T+1}|\omega_{1:T}) = P(q_{\theta_{T-1}}(\omega_T))$ which has a smaller variance due to the posterior variance of the Gaussian.

All together we finish the proof for $\text{Var}(\mathcal{M}_{\text{ours}}) < \text{Var}(\mathcal{M}_{\text{on}})$.

Online Baseline vs. Offline Baseline

This case is simpler to prove. Again, the first term on the right-hand side of Eq C.1, namely $P(\hat{y}|\mathbf{x}_{T+1}, \omega_{T+1})$ can be cancelled in this case. Moreover, the second term, namely $P(\omega_{T+1}|\omega_{1:T})$ has the same variance for both baselines, i.e., $Var(P(\omega_{T+1}|\omega_{1:T})) = Var(P(q_{\theta_0}(\omega_T))) = \sigma_{\theta_0}$. This makes sense since two baselines do not learn the drift and the uncertainty in predicting ω_{T+1} based on ω_T is always the same as the prior distribution of θ_0.

Hence, it suffices to compare the uncertainty of the last term of Eq. C.1, namely $P(\omega_{1:T}|\mathcal{D}_{1:T})$. Recall

$$\begin{aligned} \mathcal{M}_{\text{on}}: &\quad \omega_{t+1} = \text{argmax}_{\omega_{t+1}} P(\omega_{t+1}|\omega_t, \mathcal{D}_{t+1}) \\ \mathcal{M}_{\text{off}}: &\quad \omega_{t+1} = \text{argmax}_{\omega_{t+1}} P(\omega_{t+1}|\mathcal{D}_{1:t+1}) \end{aligned} \tag{C.6}$$

For the offline baseline, we are using all datasets so far, namely $\mathcal{D}_{1:t+1}$ to train the model, while the online baseline only uses $\mathcal{D}_{t+1}$. Since we are considering domain generalization with temporal concept drift, i.e., for each $i \neq j$ we have $\mathcal{D}_i \neq \mathcal{D}_j$ (otherwise we merge them), the randomness of $\bigcup_{i=1}^{t+1} \mathcal{D}_i$ is at least as large as that of $\mathcal{D}_{t+1}$ alone, i.e., $Var(\bigcup_{i=1}^{t+1} \mathcal{D}_i) \geq Var(\mathcal{D}_{t+1})$.

To prove this, let's consider the case of two domains $\mathcal{D}_1$ and $\mathcal{D}_2$ without loss of generality. Also, assume the sample size for both domains are equal. By definition of

variance, we have

$$Var(\mathcal{D}_1) = \frac{\sum_{i=1}^{n}(x_{1,i} - \mu_1)^2}{n}, \quad Var(\mathcal{D}_2) = \frac{\sum_{i=1}^{n}(x_{2,i} - \mu_2)^2}{n}, \tag{C.7}$$

while

$$Var(\mathcal{D}_1 \cup \mathcal{D}_2) = \frac{\sum_{i=1}^{n}(x_{1,i} - \frac{\mu_1+\mu_2}{2})^2 + \sum_{i=1}^{n}(x_{2,i} - \frac{\mu_1+\mu_2}{2})^2}{2n}, \tag{C.8}$$

where μ_1 and μ_2 is the sample mean for each domain, respectively and n denotes the sample size. Hence,

$$\begin{aligned}
&Var(\mathcal{D}_1 \cup \mathcal{D}_2) - \frac{1}{2}\Big(Var(\mathcal{D}_1) + Var(\mathcal{D}_2)\Big) \\
=&\frac{\sum_{i=1}^{n}(x_{1,i} - \frac{\mu_1+\mu_2}{2})^2 + \sum_{i=1}^{n}(x_{2,i} - \frac{\mu_1+\mu_2}{2})^2}{2n} - \frac{\sum_{i=1}^{n}(x_{1,i} - \mu_1)^2 + \sum_{i=1}^{n}(x_{2,i} - \mu_2)^2}{2n} \\
\propto&\sum_{i=1}^{n}(x_{1,i} - \frac{\mu_1+\mu_2}{2})^2 + \sum_{i=1}^{n}(x_{2,i} - \frac{\mu_1+\mu_2}{2})^2 - \Big(\sum_{i=1}^{n}(x_{1,i} - \mu_1)^2 + \sum_{i=1}^{n}(x_{2,i} - \mu_2)^2\Big) \\
=&\sum_{i=1}^{n}\left((x_{1,i} - \frac{\mu_1+\mu_2}{2})^2 + (x_{2,i} - \frac{\mu_1+\mu_2}{2})^2 - \Big[(x_{1,i} - \mu_1)^2 + (x_{2,i} - \mu_2)^2\Big]\right) \\
=&\sum_{i=1}^{n}\left(-(\mu_1+\mu_2)x_{1,i} - (\mu_1+\mu_2)x_{2,i} + 2\mu_1 x_{1,i} + 2\mu_2 x_{2,i} + \frac{1}{2}(\mu_1+\mu_2)^2 - \mu_1^2 - \mu_2^2\right) \\
=&\sum_{i=1}^{n}\left((\mu_1-\mu_2)x_{1,i} - (\mu_1-\mu_2)x_{2,i} - \frac{1}{2}(\mu_1-\mu_2)^2\right) \\
=&\sum_{i=1}^{n}\left((\mu_1-\mu_2)(x_{1,i} - x_{2,i}) - \frac{1}{2}(\mu_1-\mu_2)^2\right) \\
=&\sum_{i=1}^{n}\left((\mu_1-\mu_2)^2 - \frac{1}{2}(\mu_1-\mu_2)^2\right) \\
=&\sum_{i=1}^{n}\frac{(\mu_1-\mu_2)^2}{2} \geq 0,
\end{aligned} \tag{C.9}$$

where the equation from the third last row to the second last row is under expectation as $\mathbb{E}\big[(x_{1,i} - x_{2,i})\big] = \mu_1 - \mu_2$. Since we assume $Var(\mathcal{D}_1) = Var(\mathcal{D}_2)$, we finish the proof that $Var(\mathcal{D}_1 \cup \mathcal{D}_2) \geq Var(\mathcal{D}_2)$. One can generalize this conclusion to three or

more domains.

Finally, combining $Var(\bigcup_{i=1}^{t+1}\mathcal{D}_i)$ is at least as large as that of $Var(\mathcal{D}_{t+1})$ with Bayes' rule, one can finish the proof.

□

C.2 Proof of Theorem 3.1.6

Proof. We finish our proof in **two steps**. First, we prove that the generalization error of our method is smaller than that of the online baseline.

By definition, we know that

$$err \coloneqq \ell\Big(\mathbb{E}\big[P(\hat{y}_{T+1}|\mathbf{x}_{T+1},\mathcal{D}_{1:T})\big], y_{T+1}\Big) = \ell\Big(g_{\omega_{T+1}}(\mathbf{x}_{T+1}), y_{T+1}\Big), \tag{C.10}$$

where g_ω denotes the target neural network with parameter ω, and ω_{T+1} denotes the parameter status on the $(T+1)$-th domain (i.e., future domain).

For online baseline, since it does not consider the temporal information, the parameters on the future domain will be the same as the parameters after the training on the last source domain, i.e, for online baseline we have $\omega_{T+1} = \omega_T$.

For our method, we have $\omega_{T+1} = q_{\theta_T}(\omega_T)$, where q_θ is the recurrent structure and θ_T denotes the parameter status of the recurrent structure after training on the first T domains. In other words, our method utilizes the recurrent structure to generate the model parameters on the next domain. Now it suffices to show that

$$\ell\Big(g_{\omega_{T+1}}(\mathbf{x}_{T+1}), y_{T+1}\Big) < \ell\Big(g_{\omega_T}(\mathbf{x}_{T+1}), y_{T+1}\Big), \tag{C.11}$$

where $\omega_{T+1} = q_{\theta_T}(\omega_T)$. Here, the LHS and RHS above correspond to the generalization error of our method and the online baseline, respectively.

Recall that q_θ represents the LSTM unit in our case, and we train the LSTM

unit to approximate the transition probability $P(\omega_{t+1}|\omega_t)$, i.e., how neural network g's parameter distribution changes over time. From a probabilistic point of view, during training of the LSTM unit q_θ, we basically minimize the empirical loss, which is equivalent to

$$\min_{\theta} KL\Big(q_\theta\Big\|P(\omega_{t+1}|\omega_t)\Big), \quad t=1,2,\cdots,T-1. \tag{C.12}$$

As mentioned in Assumption 3.1.2, we denote θ_0 as the initialization of q_θ. On the other hand, after $T-1$ times of training over the LSTM unit on the T source domains, θ will converge to the optimum denoted as θ^*. Hence, the model parameter ω_{T+1} generated by the converged LSTM unit for sure will be closer to the ground truth than that generated by the random initialized LSTM unit, i.e.,

$$\big\|q_{\theta_T}(\omega_T) - q_{\theta^*}(\omega_T)\big\| < \big\|q_{\theta_0}(\omega_T) - q_{\theta^*}(\omega_T)\big\|. \tag{C.13}$$

By Lipschitz continuity of g_ω over the parameter ω, we have

$$L_{lower}\cdot\big\|\omega-\omega'\big\| < \big\|g_\omega(x)-g_{\omega'}(x)\big\| < L_{upper}\cdot\big\|\omega-\omega'\big\|, \ \ \forall\, x\in\mathcal{X}, \tag{C.14}$$

where $\mathcal{X}$ is defined as the input space of neural network g_ω. [19] proved that the Lipschitz constant actually can have a lower bound for a neural network.

Denote $\omega^* = q_{\theta^*}(\omega_T)$, i.e., the optimal parameter for the target neural network g on the future domain. Then, it directly follows Eq. C.14 that

$$\begin{aligned}\big\|g_{\omega_T}(x)-g_{\omega^*}(x)\big\| &> L_{lower}\cdot\big\|\omega_T-\omega^*\big\|,\\ \big\|g_{\omega_{T+1}}(x)-g_{\omega^*}(x)\big\| &< L_{upper}\cdot\big\|\omega_{T+1}-\omega^*\big\|.\end{aligned} \tag{C.15}$$

Denote $r = \|\omega_T-\omega^*\|/\|\omega_{T+1}-\omega^*\|$. Since neural network g_ω is a continuous function of ω, there always exists a constant $\delta>0$ such that, within the sphere centering at ω^*

with radius δ, namely $\mathcal{S}(\omega^*, \delta)$, the local lower and upper bound for the Lipschitz constant of g_ω could satisfy $L_{upper}/L_{lower} < r$. The reason behind this is, as δ approaches 0, due to the continuity of g_ω, the upper bound and lower bound of Lipschitz constant within $\mathcal{S}(\omega^*, \delta)$ will become closer and finally identical, i.e., $\lim_{\delta \to 0^+} L_{upper}/L_{lower} = 1$. On the other hand, by Eq. C.13 we know that r is always greater than 1, so it is always possible to find a δ to satisfy the above condition. As a result,

$$\frac{L_{upper}}{L_{lower}} \cdot \frac{\|\omega_{T+1} - \omega^*\|}{\|\omega_T - \omega^*\|} < 1 \iff L_{upper} \cdot \left\|\omega_{T+1} - \omega^*\right\| < L_{lower} \cdot \left\|\omega_T - \omega^*\right\|. \quad \text{(C.16)}$$

Hence, $\left\|g_{\omega_{T+1}}(x) - g_{\omega^*}(x)\right\| < \left\|g_{\omega_T}(x) - g_{\omega^*}(x)\right\|$. Since $g_{\omega^*}(x)$ is the optimal neural network on the future domain, $g_{\omega^*}(\mathbf{x}_{T+1})$ should achieve the lowest loss as defined in Eq. C.11. Combining everything above together finishes the first step of our proof.

The second step of our proof is for the comparison between two baselines. We consider the case where the drift of ω_t is monotonic, but our proof can be generalized to other cases easily.

As can be shown,

$$\begin{aligned} &\text{Online baseline:} \quad \mathbb{E}[P(\omega_{T+1}|\omega_{1:T})] = \mathbb{E}[q_{\theta_0}(\omega_T)] = \omega_T, \\ &\text{Offline baseline:} \quad \mathbb{E}[P(\omega_{T+1}|\omega_{1:T})] = \mathbb{E}[q_{\theta_0}(\bar{\omega})] = \mathbb{E}[P(\omega|\mathcal{D}_{1:T})]. \end{aligned} \quad \text{(C.17)}$$

If we denote a distance function over the domains, as d, we assume that $d(\mathcal{D}_{t+1}, \mathcal{D}_{T+1}) < d(\mathcal{D}_t, \mathcal{D}_{T+1})$. By the monotonic assumption, the distribution of each $\mathcal{D}_{1:T}$ is changing along a certain direction. Hence, among them $\mathcal{D}_T$ has the distribution most close to that of $\mathcal{D}_{T+1}$. In other words, the online baseline finetunes the model so its ω_T is leaning towards the last domain, while the offline baseline is using the averaged domains to train the model, which finishes the proof.

□

Appendix D

Appendix of SparseLLM

This chapter includes supplemental materials for the work SparseLLM.

D.1 Two-layer Demo on the Details behind our Global Pruning

Figure D.1 illustrates the *SparseLLM* pruning method compared to conventional global pruning and local pruning, using a two-layer neural network as an abstraction for simplicity. The figure is divided into three main parts:

On the left, conventional global pruning is depicted. This method applies a global mask to the entire network, resulting in significant memory costs due to poor scalability. Both functions f_1 and f_2 are pruned using the same mask across all layers, leading to high memory usage.

In the middle, local pruning is shown, where each layer is pruned independently. This approach reduces memory costs by applying separate masks to each layer. However, it inevitably sacrifices performance because it ignores global supervision, which can lead to suboptimal pruning decisions that do not consider the network as a whole.

On the right, the adaptive global pruning method of *SparseLLM* is presented.

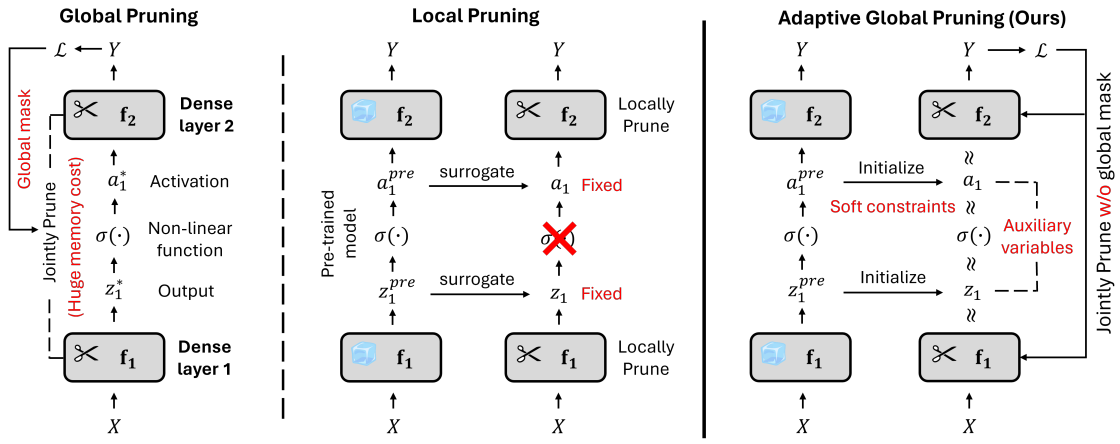


Figure D.1: **Illustration of *SparseLLM* pruning method** compared to *conventional global pruning* and *local pruning*. We consider a *two-layer* neural network as an abstraction for simplicity. *Global pruning* (**left**) is memory prohibitive due to poor scalability. *Local pruning* (**mid**) considers pruning each layer independently, while inevitably sacrificing performance due to the ignorance of global supervision. Our adaptive global pruning (**right**) achieves global pruning with low memory cost by leveraging auxiliary variables and soft constraints.

This method achieves global pruning with low memory cost by leveraging auxiliary variables and soft constraints. It combines the benefits of global pruning—considering the entire network structure—with efficient memory usage. The introduction of auxiliary variables allows for flexible and adaptive pruning, ensuring that the overall performance of the network is maintained while keeping memory costs low.

Thus, the figure highlights the trade-offs between different pruning strategies. Conventional global pruning incurs high memory costs, local pruning reduces memory usage at the expense of performance, and the adaptive global pruning of *SparseLLM* strikes a balance by maintaining performance with lower memory requirements through the use of auxiliary variables and soft constraints.

D.2 Calibration Samples

Figure D.2 and Figure D.3 present how perplexity changes with the calibration sample sizes on the datasets PTB and C4 for OPT-2.7b and LlaMA-2 7b, respectively. In both figures, as the number of calibration samples increases, the perplexity decreases for both SparseGPT and *SparseLLM*. This indicates that having more calibration samples can be beneficial in the pruning process. Perplexity decreases more rapidly from 8 samples to 32 samples. Beyond 32 samples, the rate at which perplexity decreases starts to slow down. In addition, increasing the number of calibration samples requires more computational resources, e.g., memory and computation time, in the overall pruning process. This suggests that the calibration sample sizes should be between 32 and 64 to ensure good performance while maintaining computational efficiency. Lastly, the figures show that *SparseLLM* achieves better perplexity than SparseGPT does with 32 or larger sample sizes for both OPT and LlaMA-2 models.

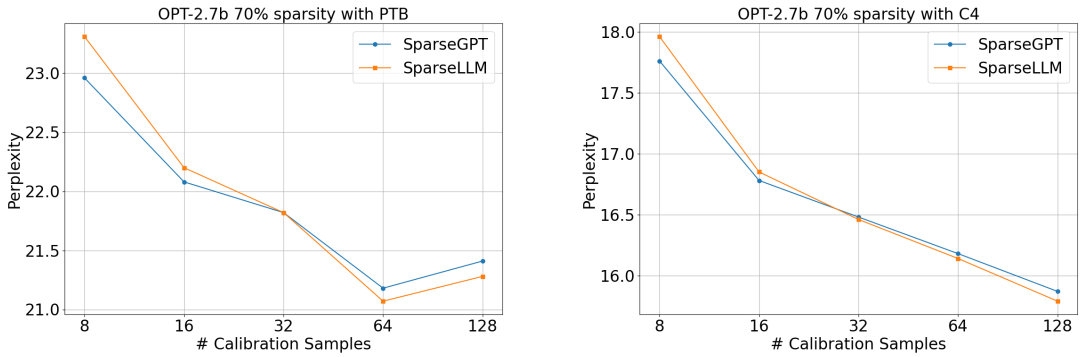


Figure D.2: Sensitivity of OPT-2.7b on the calibration sample sizes for datasets PTB and C4.

D.3 Computation Time vs. Model Sizes

We study how the computation time per layer of SparseGPT and *SparseLLM* varies with different model sizes, as illustrated in Table D.1 and Table D.2 for OPT

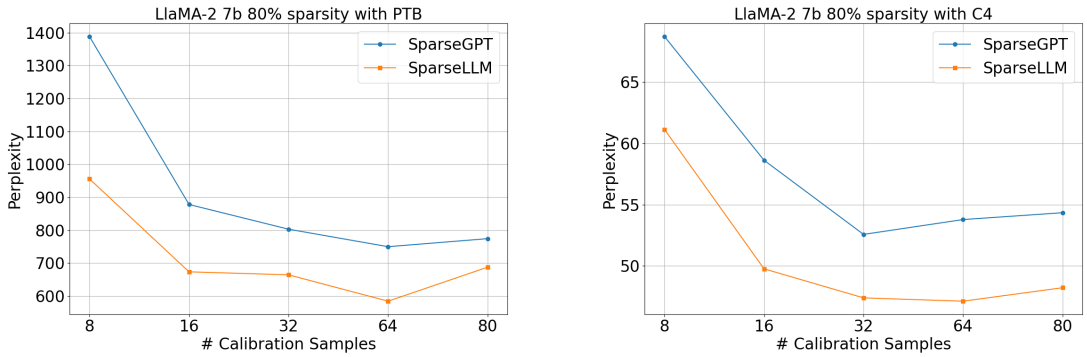


Figure D.3: Sensitivity of LlaMA-2 7b models on the calibration sample sizes for datasets PTB and C4.

models and LlaMA-2 models. The rate at which the time taken increases is comparable for SparseGPT and *SparseLLM* as the model size increases. Additionally, computation time for *SparseLLM* are reported for a configuration of 4 to 10 epochs. As we have reported in Section 4.1.5, *SparseLLM* can reduce the training loss in as few as 2 to 3 epochs. This suggests that the proposed *SparseLLM* remains computationally efficient.

Table D.1: Computation time in seconds of OPT models.

Method	OPT-125m	OPT-1.3b	OPT-2.7b	OPT-6.7b	OPT-13b	OPT-30b	OPT-66b
SparseGPT	2.30	10.18	18.35	24.40	28.65	48.91	103.19
SparseLLM	4.34	22.79	42.86	174.08	85.62	174.07	284.59

Table D.2: Computation time in seconds of LlaMA-2 models.

Method	Llama-2 7b	Llama-2 13b
SparseGPT	17.94	16.58
SparseLLM	46.80	252.48

D.4 Experiment Results for Additional Models

Detailed results on perplexity and zero-shot task accuracy for additional models are reported in Table D.3 and Table D.4. Similar to other models, we report the

perplexity results for the dense model next to the name of the model in the table. In particular, we see that SparseGPT and *SparseLLM* outperform Magnitude and Wanda with a significant margin across different sparsity. *SparseLLM* shares similar perplexity with SparseGPT for smaller sparsity but demonstrates much better perplexity for larger sparsity. Similar perplexity trends are observed across all three datasets, although, PTB, having the highest perplexity for each sparsity and method, is likely the most challenging dataset among the three. For the zero-shot taks accuracy, we see that *SparseLLM* achieves comparable results to SparseGPT for smaller perplexity and the performance improvements are more obvious and significant with higher sparsity.

Table D.3: Perplexity in high sparsity regimes ($\geq 70\%$); the lower the perplexity, the better.

OPT-125m (WikiText2 (WT2): 27.66; PTB: 38.99; C4: 26.56)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	3806.96	3429.35	2263.37	4890.96	4121.43	3213.85	5613.18	5380.80	4475.29	-	-	-
Wanda	351.83	412.52	248.94	1912.45	2512.93	1066.86	4940.89	4337.27	3126.02	-	-	-
SparseGPT	239.26	265.83	156.33	2072.12	1952.85	1050.85	6131.57	6963.27	2443.33	1482.61	2215.44	657.26
SparseLLM	208.46	255.75	137.72	1358.10	1418.06	654.54	5291.64	5067.41	2003.09	914.87	1210.84	450.01
OPT-6.7b (WikiText2 (WT2): 10.86; PTB: 15.77; C4: 12.71)												
Sparsity	70%			80%			90%			3:4		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
Magnitude	7054.25	5437.44	4850.25	7937.49	5971.86	6031.54	2.4e4	2.5e4	2.1e4	-	-	-
Wanda	54.95	129.73	116.67	1493.58	1196.93	996.00	2.1e4	2.0e4	1.8e4	-	-	-
SparseGPT	21.27	18.90	15.28	31.04	51.26	29.42	8871.24	5713.57	3797.26	570.08	361.81	328.18
SparseLLM	19.16	18.39	14.93	23.96	39.32	26.97	2095.85	1842.48	953.44	83.36	128.99	62.11

D.4.1 Hyperparameter Selection

Hyperparameters α and β are used in Eq. 4.5. We select α and β from the set $\{0.01, 0.1, 1, 5, 10, 100\}$ and perform a study on models to understand the impact of the hyperparameters. Results for OPT-1.3b with 70% sparsity are shown in Table D.5.

Table D.4: Accuracy (%) of zero-shot tasks; the higher the accuracy, the better.

OPT-6.7b									
Sparsity	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
Dense		66.12	56.03	50.49	65.27	65.72	30.63	27.60	51.69
70%	SparseGPT	61.74	54.87	48.46	63.85	64.31	29.27	25.40	49.70
	SparseLLM	60.61	54.51	48.8	62.9	64.14	30.03	26.60	49.66
80%	SparseGPT	55.08	48.38	42.22	59.43	57.79	25.85	21.40	44.31
	SparseLLM	58.69	51.26	43.78	59.67	58.38	26.88	22.00	45.81
90%	SparseGPT	38.53	53.07	26.00	48.07	26.81	21.67	14.40	32.65
	SparseLLM	46.48	52.71	26.21	51.70	27.44	19.71	13.40	33.95
3:4	SparseGPT	46.70	54.15	28.82	51.07	32.45	18.17	15.40	35.25
	SparseLLM	53.49	53.42	36.24	53.51	43.94	22.61	17.40	40.09

Table D.5: Ablations of the hyperparameters α and β on OPT-1.3b with 70% sparsity (in perplexity)

α / β	0.01	0.1	1	5	10	100
0.01	18.01	17.97	17.97	-	-	-
0.1	18.04	**17.82**	17.96	18.04	18.40	-
1	18.20	18.02	18.11	17.87	17.96	18.22
5	18.06	18.02	18.03	17.92	17.96	18.04
10	18.03	18.01	17.96	17.96	17.96	18.03
100	18.04	18.04	17.98	18.01	18.01	18.03

Appendix E

Theory Proof of FedSpaLLM

E.1 Proof of Corollary 4.2.1 (Sparsity Guarantee)

Proof. 1. **Client Sparsity Consistency:** Each client enforces the same target sparsity $\mathcal{S}_{global}$. This implies that for each client i, the sparsity of the pruned layers $\mathcal{L}_i$ matches the global sparsity target $\mathcal{S}_{global}$. Formally, we have:

$$\mathcal{S}_i = \mathcal{S}_{global}, \quad \forall i = 1, 2, \ldots, N.$$

Since all clients prune their local models independently but according to the same target sparsity, each pruned local model achieves the same sparsity.

2. **Layer Sampling Strategy:** The layer sampling strategy ensures that all layers of the model $\mathcal{L}$ are eventually sampled across all clients during each communication round. Therefore, every layer in the global model $\hat{W}_g$ has been pruned by each client according to the same sparsity criterion $\mathcal{S}_{global}$.

3. **Aggregation with ℓ_0-norm:** The aggregation function using ℓ_0-norm averages only the non-zero elements (i.e., the pruned weights) from the client models. Since all clients enforce the same sparsity, and the aggregation only involves the non-zero weights from these pruned models, the sparsity of the aggregated global model

will match that of the local models. Specifically:

$$\mathcal{S}_{global} = \mathcal{S}_i, \quad \forall i.$$

Thus, the sparsity of the global model after aggregation is equivalent to the sparsity of each client model.

4. **Conclusion:** Therefore, the global model $\hat{W}_g$ will maintain the target sparsity $\mathcal{S}_{global}$ after aggregation in each communication round. The aggregation process ensures that the global sparsity is consistent with the target sparsity across rounds.

$$\boxed{\mathcal{S}_{global} = \mathcal{S}_i, \quad \forall i.}$$

□

E.2 Proof of Theorem 4.2.2 (Unbiased Estimator)

Proof. 1. **Layer Sampling Strategy:** Let $\mathcal{L} = \{L_1, L_2, \ldots, L_m\}$ denote the set of all layers in the model, where m is the total number of layers. In each communication round, the server randomly samples a subset of layers $\mathcal{L}_i \subseteq \mathcal{L}$ for each client i. Each layer $L_j \in \mathcal{L}$ has an equal probability p_j of being selected across clients.

The expectation of the sampled weights for layer L_j across all clients can be expressed as:

$$\mathbb{E}[W_i[L_j]] = p_j W_g^*[L_j],$$

where $W_g^*[L_j]$ is the weight of layer L_j in the fully pruned global model W_g^*.

2. **Unbiasedness of Layer Sampling:** Since each layer has an equal probability of being sampled across clients, the expected contribution of each layer is proportional to its selection probability. Over multiple communication rounds, all layers will be sampled enough times to represent the fully pruned model.

Therefore, the expected value of the global model $\hat{W}_g$ is the same as the fully pruned model W_g^*. For any layer L_j, we have:

$$\begin{aligned} \mathbb{E}[\hat{W}_g[L_j]] &= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} W_i[L_j]\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[W_i[L_j]] = W_g^*[L_j]. \end{aligned} \tag{E.1}$$

Thus, the global model $\hat{W}_g$ is an unbiased estimator of W_g^*, as the expected value of the pruned weights matches the fully pruned model.

3. **Conclusion:** Therefore, the global model $\hat{W}_g$ obtained by aggregating pruned models after layer sampling is an unbiased estimator of the fully pruned model W_g^*. Formally, we can conclude that:

$$\boxed{\mathbb{E}[\hat{W}_g] = W_g^*}.$$

This unbiased property holds as long as each layer has an equal probability of being selected across clients during each communication round. □

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

[2] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[4] Andreas Argyriou, Charles A Micchelli, Massimiliano Pontil, and Yiming Ying. A spectral regularization framework for multi-task structure learning. In *NIPS*, volume 1290, page 1296. Citeseer, 2007.

[5] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[6] Ali Ayub and Alan R Wagner. Eec: Learning to encode and regenerate images for continual learning. *arXiv preprint arXiv:2101.04904*, 2021.

[7] Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with

drift-aware dynamic neural networks. In *The Eleventh International Conference on Learning Representations*, 2022.

[8] Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*, 2024.

[9] Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. Sparsellm: Towards global pruning of pre-trained language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[10] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018.

[11] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[12] Nathan Bell and Michael Garland. Efficient sparse matrix-vector multiplication on cuda. Technical report, Citeseer, 2008.

[13] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[15] Marcelo Bertalmio, Andrea L Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001*

IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, pages I–I. IEEE, 2001.

[16] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14:629–654, 2008.

[17] Michael Bommarito II and Daniel Martin Katz. Gpt takes the bar exam. *arXiv preprint arXiv:2212.14402*, 2022.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[19] Sébastien Bubeck, Yuanzhi Li, and Dheeraj M Nagaraj. A law of robustness for two-layers neural networks. In *Conference on Learning Theory*, pages 804–820. PMLR, 2021.

[20] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[21] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Fedadapter: Efficient federated learning for modern nlp. In *MobiCom*, 2023.

[22] Zekun Cai, Guangji Bai, Renhe Jiang, Xuan Song, and Liang Zhao. Continuous temporal domain generalization. *arXiv preprint arXiv:2405.16075*, 2024.

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

[24] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[25] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 42–50, 2011.

[26] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *International conference on machine learning*, pages 1779–1788. PMLR, 2020.

[27] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR, 2023.

[28] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[29] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *ICLR*, 2021.

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[31] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pages 845–850, 2015.

[33] Sayna Ebrahimi, Suzanne Petryk, Akash Gokul, William Gan, Joseph E Gonzalez, Marcus Rohrbach, and Trevor Darrell. Remembering for the right reasons: Explanations reduce catastrophic forgetting. *arXiv preprint arXiv:2010.01528*, 2020.

[34] An Evgeniou and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

[35] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

[36] Theodoros Evgeniou, Charles A Micchelli, Massimiliano Pontil, and John Shawe-Taylor. Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(4), 2005.

[37] Zihan Fang, Zheng Lin, Zhe Chen, Xianhao Chen, Yue Gao, and Yuguang Fang. Automated federated pipeline for parameter-efficient fine-tuning of large language models. *Arxiv*, 2024.

[38] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning

for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[39] Elias Frantar and Dan Alistarh. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

[40] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[41] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

[42] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[43] Zishan Gu, Ke Zhang, Guangji Bai, Liang Chen, Liang Zhao, and Carl Yang. Dynamic activation of clients and parameters for federated learning over heterogeneous graphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1597–1610. IEEE, 2023.

[44] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing

deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[45] Fengxiang He and Dacheng Tao. Recent advances in deep learning theory. *arXiv preprint arXiv:2012.10931*, 2020.

[46] Huan He, Owen Queen, Teddy Koker, Consuelo Cuevas, Theodoros Tsiligkaridis, and Marinka Zitnik. Domain adaptation for time series under feature and label shifts. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12746–12774. PMLR, 23–29 Jul 2023.

[47] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1): 10882–11005, 2021.

[48] Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 867–874, 2014.

[49] Charlie Hou, Akshat Shrivastava, Hongyuan Zhan, Rylan Conway, Trang Le, Adithya Sagar, Giulia Fanti, and Daniel Lazar. Pre-text: Training language models on private federated data in the age of llms. *arXiv preprint arXiv:2406.02958*, 2024.

[50] Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Distributed pruning towards tiny neural networks in federated learning. *ICDCS*, 2023.

[51] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34:21099–21111, 2021.

[52] Xiaopeng Jiang and Cristian Borcea. Complement sparsification: Low-overhead model pruning for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8087–8095, 2023.

[53] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Prunefl: Model pruning enables efficient federated learning on edge devices. *TNNLS*, 2022.

[54] Xiaoyong Jin, Youngsuk Park, Danielle Maddix, Hao Wang, and Yuyang Wang. Domain adaptation for time series forecasting via attention sharing. In *International Conference on Machine Learning*, pages 10280–10297. PMLR, 2022.

[55] Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. Client-customized adaptation for parameter-efficient federated learning. In *ACL Findings*, 2023.

[56] Jason Kingdon and Jason Kingdon. Hypothesising neural nets. *Intelligent Systems and Financial Forecasting*, pages 81–106, 1997.

[57] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[58] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[59] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is np-hard. In *International Conference on Machine Learning*, pages 5327–5337. PMLR, 2020.

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

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[62] Eldar Kurtic, Elias Frantar, and Dan Alistarh. Ziplm: Hardware-aware structured pruning of language models. *arXiv preprint arXiv:2302.04089*, 2023.

[63] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2): 74–82, 2011.

[64] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116, 2022.

[65] Kwei-Herng Lai, Lan Wang, Huiyuan Chen, Kaixiong Zhou, Fei Wang, Hao Yang, and Xia Hu. Context-aware domain adaptation for time series anomaly detection. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 676–684. SIAM, 2023.

[66] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[67] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998.

[68] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

[69] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In *PHM Society European Conference*, volume 3, 2016.

[70] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[71] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Sequential learning for domain generalization. In *European Conference on Computer Vision*, pages 603–619. Springer, 2020.

[72] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[73] Lin Li, Jianping Gou, Baosheng Yu, Lan Du, and Zhang Yiand Dacheng Tao. Federated distillation: A survey. *arXiv preprint arXiv:2404.08564*, 2024.

[74] Shuang Li, Yilun Du, Gido M van de Ven, and Igor Mordatch. Energy-based models for continual learning. *arXiv preprint arXiv:2011.12216*, 2020.

[75] Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models

based on low-rank and sparse approximation. *arXiv preprint arXiv:2306.11222*, 2023.

[76] Yuening Li, Zhengzhang Chen, Daochen Zha, Mengnan Du, Jingchao Ni, Denghui Zhang, Haifeng Chen, and Xia Hu. Towards learning disentangled representations for time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3270–3278, 2022.

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

[78] Zilinghan Li, Shilan He, Pranshu Chaturvedi, Volodymyr Kindratenko, Eliu A Huerta, Kibaek Kim, and Ravi Madduri. Secure federated learning across heterogeneous cloud and high-performance computing resources-a case study on federated fine-tuning of llama 2. *Computing in Science & Engineering*, 2024.

[79] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. In *NAACL Findings*, 2022.

[80] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[81] Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Tianjiao Zhao, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*, 2305, 2023.

[82] Qiao Liu and Hui Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. In *IJCAI*, pages 2744–2750, 2021.

[83] Zirui Liu, Qingquan Song, Qiang Charles Xiao, Sathiya Keerthi Selvaraj, Rahul Mazumder, Aman Gupta, and Xia Hu. Ffsplit: Split feed-forward network for optimizing accuracy-efficiency trade-off in language model inference. *arXiv preprint arXiv:2401.04044*, 2024.

[84] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[85] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and S Yu Philip. Learning multiple tasks with multilinear relationship networks. In *NIPS*, 2017.

[86] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[87] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5334–5343, 2017.

[88] Chen Luo, Zhengzhang Chen, Lu-An Tang, Anshumali Shrivastava, Zhichun Li, Haifeng Chen, and Jieping Ye. Tinet: learning invariant networks via knowledge transfer. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1890–1899, 2018.

[89] Dongsheng Luo, Wei Cheng, Yingheng Wang, Dongkuan Xu, Jingchao Ni, Wenchao Yu, Xuchao Zhang, Yanchi Liu, Yuncong Chen, Haifeng Chen, et al. Time

series contrastive learning with information-aware augmentations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4534–4542, 2023.

[90] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1930–1939, 2018.

[91] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023.

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

[93] Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[94] Andreas Maurer. The rademacher complexity of linear transformation classes. In *International Conference on Computational Learning Theory*, pages 65–78. Springer, 2006.

[95] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

[96] Michael McCloskey and Neal J Cohen. Catastrophic interference in connection-

ist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

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

[98] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[99] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[100] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5715–5725, 2017.

[101] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013.

[102] Muhammad Tahir Munir, Muhammad Mustansar Saeed, Mahad Ali, Zafar Ayyub Qazi, and Ihsan Ayyub Qazi. Fedprune: Towards inclusive federated learning. *Arxiv*, 2021.

[103] Anshul Nasery, Soumyadeep Thakur, Vihari Piratla, Abir De, and Sunita Sarawagi. Training for the future: A simple gradient interpolation loss to generalize along time. *Advances in Neural Information Processing Systems*, 34, 2021.

[104] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[105] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[106] R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2:13, 2023.

[107] Guillermo Ortiz-Jimenez, Mireille El Gheche, Effrosyni Simou, Hermina Petric Maretic, and Pascal Frossard. Cdot: Continuous domain adaptation using optimal transport. *arXiv preprint arXiv:1909.11448*, 2019.

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

[109] Gunho Park, Baeseong Park, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022.

[110] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[111] Jialun Peng, Dong Liu, Songcen Xu, and Houqiang Li. Generating diverse structure for image inpainting with hierarchical vq-vae. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10775–10784, 2021.

[112] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In *International Conference on Machine Learning*, pages 5102–5112. PMLR, 2019.

[113] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[114] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[115] Mohamed Ragab, Emadeldeen Eldele, Wee Ling Tan, Chuan-Sheng Foo, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Adatime: A benchmarking suite for domain adaptation on time series data. *ACM Transactions on Knowledge Discovery from Data*, 17(8):1–18, 2023.

[116] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, pages 81–94, 2020.

[117] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. 2019.

[118] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[119] Chao Ren, Han Yu, Hongyi Peng, Xiaoli Tang, Anran Li, Yulan Gao, Alysa Ziying Tan, Bo Zhao, Xiaoxiao Li, Zengxiang Li, et al. Advances and open

challenges in federated learning with foundation models. *arXiv preprint arXiv:2404.15381*, 2024.

[120] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

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

[122] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

[123] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[124] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*, 2018.

[125] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[126] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[127] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[128] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[129] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140, 2015.

[130] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Learning task relatedness in multi-task learning for images in context. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 78–86, 2019.

[131] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 443–450. Springer, 2016.

[132] Jingwei Sun, Ziyue Xu, Hongxu Yin, Dong Yang, Daguang Xu, Yiran Chen, and Holger R. Roth. Fedbpt: Efficient federated black-box prompt tuning for large language models. *Arxiv*, 2024.

[133] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[134] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

[135] Yi-Lin Sung, Jaehong Yoon, and Mohit Bansal. Ecoflap: Efficient coarse-to-fine layer-wise pruning for vision-language models. *arXiv preprint arXiv:2310.02998*, 2023.

[136] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*, pages 269–278, 2020.

[137] Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10880–10895, 2023.

[138] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[139] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[140] Johnny Torres, Guangji Bai, Junxiang Wang, Liang Zhao, Carmen Vaca, and Cristina Abad. Sign-regularized multi-task learning. *arXiv preprint arXiv:2102.11191*, 2021.

[141] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[142] Shikhar Tuli and Niraj K Jha. Acceltran: A sparsity-aware accelerator for dynamic inference with transformers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[143] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[144] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[145] Dongjie Wang, Zhengzhang Chen, Yanjie Fu, Yanchi Liu, and Haifeng Chen. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2269–2278, 2023.

[146] Dongjie Wang, Zhengzhang Chen, Jingchao Ni, Liang Tong, Zheng Wang, Yanjie Fu, and Haifeng Chen. Interdependent causal networks for root cause localization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5051–5060, 2023.

[147] Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. *arXiv preprint arXiv:2007.01807*, 2020.

[148] Haozhao Wang, Yichen Li, Wenchao Xu, Ruixuan Li, Yufeng Zhan, and Zhigang Zeng. Dafkd: Domain-aware federated knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20412–20421, 2023.

[149] Junxiang Wang and Liang Zhao. Multi-instance domain adaptation for vaccine adverse event detection. In *Proceedings of the 2018 World Wide Web Conference*, pages 97–106, 2018.

[150] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Met-

zler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[151] Garrett Wilson, Janardhan Rao Doppa, and Diane J Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1768–1778, 2020.

[152] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[153] Feijie Wu, Zitao Li, Yaliang Li, Bolin Ding, and Jing Gao. Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3345–3355, 2024.

[154] Sen Wu, Hongyang R Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. *arXiv preprint arXiv:2005.00944*, 2020.

[155] Canwen Xu and Julian McAuley. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10566–10575, 2023.

[156] Yuxuan Yan, Shunpu Tang, Zhiguo Shi, and Qianqian Yang. Federa: Efficient fine-tuning of language models in federated learning leveraging weight decomposition. *Arxiv*, 2024.

[157] Ling Yang and Shenda Hong. Unsupervised time-series representation learning

with iterative bilinear temporal-spectral fusion. In *International Conference on Machine Learning*, pages 25038–25054. PMLR, 2022.

[158] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

[159] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5485–5493, 2017.

[160] Kai Yi, Nidham Gazagnadou, Peter Richtarik, and Lingjuan Lyu. Fedp3: Federated personalized and privacy-friendly network pruning under model heterogeneity. *ICLR*, 2024.

[161] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[162] Jiaxuan You, Jure Leskovec, Kaiming He, and Saining Xie. Graph structure of neural networks. In *International Conference on Machine Learning*, pages 10881–10891. PMLR, 2020.

[163] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[164] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin,

et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[165] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[166] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer, 2014.

[167] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy preserving prompt tuning in federated learning. In *ICASSP*, 2023.

[168] Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. Learning sleep stages from radio signals: A conditional adversarial architecture. In *International Conference on Machine Learning*, pages 4100–4109. PMLR, 2017.

[169] Sicheng Zhao, Bo Li, Pengfei Xu, and Kurt Keutzer. Multi-source domain adaptation in the deep learning era: A systematic survey. *arXiv preprint arXiv:2002.12169*, 2020.

[170] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–416, 2018.

[171] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Malsar: Multi-task learning via structural regularization. *Arizona State University*, 21, 2011.

[172] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[173] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

[174] Yongchun Zhu, Fuzhen Zhuang, Jindong Wang, Guolin Ke, Jingwu Chen, Jiang Bian, Hui Xiong, and Qing He. Deep subdomain adaptation network for image classification. *IEEE transactions on neural networks and learning systems*, 32 (4):1713–1722, 2020.